

ОГЛАВЛЕНИЕ

ПРЕДИСЛОВИЕ	10
СПИСОК СОКРАЩЕНИЙ И ОБОЗНАЧЕНИЙ	11
ВВЕДЕНИЕ.....	13
ГЛАВА 1. ТЕОРЕТИЧЕСКИЕ ОСНОВЫ УПРАВЛЕНИЯ ГЕТЕРОГЕННЫМИ РАСПРЕДЕЛЁННЫМИ ВЫЧИСЛИТЕЛЬНЫМИ СИСТЕМАМИ	17
1.1. Понятие и свойства гетерогенных распределённых вычислительных систем	17
1.2. Управление как информационный процесс	19
1.3. Архитектуры управления распределёнными системами	20
1.3.1. Централизованное управление	20
1.3.2. Децентрализованное управление	21
1.3.3. Распределённое управление	22
1.3.4. Гибридные архитектуры	23
1.4. Основные задачи управления	23
1.4.1. Распределение ресурсов	24
1.4.2. Балансировка нагрузки	24
1.4.3. Управление потоками данных	25
1.4.4. Обеспечение устойчивости	25
1.5. Ограничения классических подходов	26
1.5.1. Ограничения вычислительных ресурсов	27
1.5.2. Ограничения пропускной способности каналов связи	28
1.5.3. Ограничения времени выполнения	28
1.5.4. Ограничения размещения	28
1.5.5. Ограничения совместимости	28
1.5.6. Энергетические ограничения	28
1.5.7. Ограничения надёжности	29
1.5.8. Ограничения согласованности данных	29
Выводы по главе 1	30

ГЛАВА 2. МОДЕЛИ И ФОРМАЛЬНОЕ ПРЕДСТАВЛЕНИЕ ГЕТЕРОГЕННЫХ РАСПРЕДЕЛЁННЫХ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ.....32

2.1. ОБЩАЯ ФОРМАЛИЗАЦИЯ СИСТЕМЫ	32
2.2. МОДЕЛЬ ВЫЧИСЛИТЕЛЬНЫХ УЗЛОВ	34
2.3. ГРАФОВАЯ МОДЕЛЬ СИСТЕМЫ.....	35
2.4. МОДЕЛЬ ЗАДАЧ	37
2.5. ОТОБРАЖЕНИЕ РАСПРЕДЕЛЕНИЯ ЗАДАЧ	38
2.6. МОДЕЛЬ ПОТОКОВ ДАННЫХ	39
2.7. СОСТОЯНИЕ СИСТЕМЫ	40
2.8. ФОРМАЛИЗАЦИЯ ОГРАНИЧЕНИЙ	42
2.8.1. Ресурсные ограничения	42
2.8.2. Сетевые ограничения.....	43
2.8.3. Временные ограничения.....	43
2.8.4. Обобщённая форма ограничений.....	43
2.9. ЦЕЛЕВАЯ ФУНКЦИЯ УПРАВЛЕНИЯ.....	44
2.9.1. Минимизация времени выполнения.....	44
2.9.2. Минимизация загрузки ресурсов	45
2.9.3. Многокритериальная постановка задачи управления	45
2.10. ДИНАМИЧЕСКАЯ МОДЕЛЬ СИСТЕМЫ	46
2.11. ОБОБЩЕНИЕ ФОРМАЛЬНОЙ МОДЕЛИ СИСТЕМЫ	48
Выводы по главе 2.....	49

ГЛАВА 3. ПРИНЦИПЫ И МЕТОДЫ АДАПТИВНОГО УПРАВЛЕНИЯ ГЕТЕРОГЕННЫМИ РАСПРЕДЕЛЁННЫМИ ВЫЧИСЛИТЕЛЬНЫМИ СИСТЕМАМИ 51

3.1. ПОНЯТИЕ АДАПТИВНОГО УПРАВЛЕНИЯ	51
3.2. КОНТУР АДАПТИВНОГО УПРАВЛЕНИЯ.....	53
3.3. КЛАССИФИКАЦИЯ МЕТОДОВ АДАПТАЦИИ	55
3.3.1. Реактивные методы.....	56
3.3.2. Проактивные методы	56
3.3.3. Обучающиеся методы	57
3.4. МЕТОДЫ ПРИНЯТИЯ РЕШЕНИЙ В УСЛОВИЯХ НЕОПРЕДЕЛЁННОСТИ	57
3.4.1. Эвристические методы.....	58
3.4.2. Оптимизационные методы.....	58
3.4.3. Вероятностные методы	59
3.5. ПРЕДИКТИВНОЕ УПРАВЛЕНИЕ	60
3.6. САМООРГАНИЗАЦИЯ И САМОНАСТРОЙКА	62
3.7. ИНТЕГРАЦИЯ АДАПТИВНОГО УПРАВЛЕНИЯ С ФОРМАЛЬНОЙ МОДЕЛЬЮ ...	64
3.8. КЛЮЧЕВЫЕ СВОЙСТВА АДАПТИВНОГО УПРАВЛЕНИЯ	66
3.9. ОГРАНИЧЕНИЯ АДАПТИВНОГО УПРАВЛЕНИЯ.....	67
Выводы по главе 3.....	68

ГЛАВА 4. МУЛЬТИАГЕНТНЫЙ ПОДХОД К АДАПТИВНОМУ УПРАВЛЕНИЮ ГЕТЕРОГЕННЫМИ РАСПРЕДЕЛЁННЫМИ ВЫЧИСЛИТЕЛЬНЫМИ СИСТЕМАМИ 70

4.1. Необходимость мультиагентного подхода.....	70
4.2. Формальная модель мультиагентной системы управления	72
4.3. Декомпозиция задачи управления	75
4.4. Классы агентов в системе управления	77
4.4.1. Агент мониторинга	77
4.4.2. Агент вычислительных ресурсов.....	79
4.4.3. Агент сети.....	80
4.4.4. Агент времени и качества обслуживания.....	81
4.4.5. Агент размещения.....	82
4.4.6. Агент оптимизации	83
4.4.7. Агент прогнозирования.....	84
4.4.8. Агент согласования.....	85
4.5. Взаимодействие агентов.....	87
4.6. Распределённое принятие решений	89
4.7. Архитектуры мультиагентной системы	91
4.8. Связь с адаптивным управлением	93
4.9. Интегральная модель управления	95
4.10. Ключевые преимущества мультиагентного подхода.....	97
Выводы по главе 4.....	99

ГЛАВА 5. МЕТОДЫ РАСПРЕДЕЛЕНИЯ И ОПТИМИЗАЦИИ РЕСУРСОВ В ГЕТЕРОГЕННЫХ РАСПРЕДЕЛЁННЫХ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМАХ 102

5.1. Постановка задачи распределения ресурсов	102
5.2. Декомпозиция задачи между агентами	104
5.3. Статические и простые динамические методы распределения.....	106
5.3.1. Метод round-robin	106
5.3.2. Метод минимальной загрузки.....	107
5.4. Динамические методы распределения	108
5.4.1. Жадные алгоритмы	108
5.4.2. Алгоритмы на основе очередей	109
5.5. Методы оптимизации.....	110
5.5.1. Многокритериальная оптимизация	110
5.5.2. Генетические алгоритмы.....	112
5.5.3. Эволюционные методы.....	112
5.5.4. Локальный поиск.....	113
5.6. Управление потоками данных	114
5.6.1. Маршрутизация.....	114
5.6.2. Балансировка потоков.....	115

5.6.3. Разделение потоков	115
5.7. РАСПАРАЛЛЕЛИВАНИЕ ЗАДАЧ	116
5.8. АЛГОРИТМЫ В МУЛЬТИАГЕНТНОЙ СИСТЕМЕ	118
5.9. АДАПТИВНАЯ ОПТИМИЗАЦИЯ	121
5.10. ГИБРИДНЫЕ МЕТОДЫ.....	123
5.11. СВЯЗЬ С ИССЛЕДОВАНИЯМИ АВТОРА	126
Выводы по главе 5.....	128
ГЛАВА 6. НАБЛЮДАЕМОСТЬ И ДАННЫЕ В КОНТУРЕ АДАПТИВНОГО УПРАВЛЕНИЯ	131
6.1. Роль данных в управлении распределёнными системами	131
6.2. ФОРМАЛИЗАЦИЯ ПРОЦЕССА НАБЛЮДЕНИЯ	133
6.3. МЕТРИКИ СОСТОЯНИЯ СИСТЕМЫ	135
6.3.1. Ресурсные метрики.....	136
6.3.2. Сетевые метрики.....	136
6.3.3. Метрики задач.....	137
6.3.4. Системные метрики.....	137
6.4. НАБЛЮДАЕМОСТЬ (OBSERVABILITY).....	138
6.5. ОБРАБОТКА ПОТОКОВ ДАННЫХ.....	140
6.5.1. Потокковая обработка.....	141
6.5.2. Event-driven модель	142
6.6. ОЦЕНКА СОСТОЯНИЯ СИСТЕМЫ	143
6.7. ВЫЯВЛЕНИЕ АНОМАЛИЙ	145
6.8. ЗАМЫКАНИЕ КОНТУРА УПРАВЛЕНИЯ	147
6.9. ЗАДЕРЖКИ И ИХ ВЛИЯНИЕ	149
6.10. НЕОПРЕДЕЛЁННОСТЬ ДАННЫХ	151
6.11. СВЯЗЬ НАБЛЮДАЕМОСТИ С МУЛЬТИАГЕНТНОЙ СИСТЕМОЙ.....	153
6.12. Ключевой вывод.....	155
Выводы по главе 6.....	156
ГЛАВА 7. ИНТЕЛЛЕКТУАЛЬНЫЕ МЕТОДЫ УПРАВЛЕНИЯ ГЕТЕРОГЕННЫМИ РАСПРЕДЕЛЁННЫМИ ВЫЧИСЛИТЕЛЬНЫМИ СИСТЕМАМИ	159
7.1. НЕОБХОДИМОСТЬ ИНТЕЛЛЕКТУАЛИЗАЦИИ УПРАВЛЕНИЯ.....	159
7.2. ОБЩАЯ МОДЕЛЬ ИНТЕЛЛЕКТУАЛЬНОГО УПРАВЛЕНИЯ	162
7.3. МАШИННОЕ ОБУЧЕНИЕ В УПРАВЛЕНИИ.....	165
7.3.1. Постановка задачи машинного обучения в контуре управления	165
7.3.2. Регрессионные методы	166
7.3.3. Классификационные методы.....	167
7.3.4. Кластеризация состояний, ресурсов и нагрузки.....	167
7.3.5. Машинное обучение в мультиагентном контуре управления...	168
7.4. ОБУЧЕНИЕ С ПОДКРЕПЛЕНИЕМ (REINFORCEMENT LEARNING)	169
7.4.1. Формализация.....	170

7.4.2. Цель обучения	171
7.4.3. Применение в управлении ГРВС.....	171
7.4.4. Ограничения и практическое использование.....	172
7.5. НЕЙРОСЕТЕВЫЕ МОДЕЛИ.....	173
7.5.1. Прогнозирование состояния вычислительной среды.....	174
7.5.2. Аппроксимация функций управления	175
7.5.3. Нейросетевые архитектуры для временных и потоковых данных.....	175
7.5.4. Роль нейросетевых моделей в гибридном управлении.....	177
7.6. ГИБРИДНЫЕ МЕТОДЫ.....	178
7.7. ИНТЕЛЛЕКТУАЛИЗАЦИЯ АГЕНТОВ.....	181
7.7.1. Интеллектуализация <i>Compute Agent</i>	182
7.7.2. Интеллектуализация <i>Network Agent</i>	183
7.7.3. Интеллектуализация <i>Optimization Agent</i>	183
7.7.4. Интеллектуализация <i>Prediction Agent</i>	184
7.7.5. Коллективная интеллектуализация мультиагентной системы.....	185
7.8. АДАПТАЦИЯ МОДЕЛЕЙ.....	185
7.8.1. <i>Online learning</i>	186
7.8.2. <i>Incremental learning</i>	187
7.8.3. <i>Transfer learning</i>	188
7.8.4. Контроль качества адаптации.....	188
7.9. ОГРАНИЧЕНИЯ ИНТЕЛЛЕКТУАЛЬНЫХ МЕТОДОВ	189
7.10. СВЯЗЬ С АДАПТИВНЫМ УПРАВЛЕНИЕМ.....	191
7.11. КЛЮЧЕВОЙ ВЫВОД.....	192
Выводы по главе 7.....	193

ГЛАВА 8. LLM-АГЕНТЫ В АДАПТИВНОМ УПРАВЛЕНИИ И РАЗРАБОТКЕ РАСПРЕДЕЛЁННЫХ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ..... 196

8.1. ЭВОЛЮЦИЯ ИНТЕЛЛЕКТУАЛЬНЫХ МЕТОДОВ УПРАВЛЕНИЯ.....	196
8.2. ПОНЯТИЕ LLM-АГЕНТА	198
8.3. АРХИТЕКТУРА LLM-АГЕНТА	200
8.4. ИНТЕГРАЦИЯ LLM-АГЕНТОВ В СИСТЕМУ УПРАВЛЕНИЯ.....	203
8.4.1. Интеграция с <i>Prediction Agent</i>	204
8.4.2. Интеграция с <i>Optimization Agent</i>	205
8.4.3. Интеграция с <i>Coordination Agent</i>	206
8.4.4. Роль LLM-агентов в интеллектуальной архитектуре управления	207
8.5. РОЛИ LLM-АГЕНТОВ В УПРАВЛЕНИИ	207
8.5.1. Агент анализа состояния.....	208
8.5.2. Агент поддержки принятия решений.....	209
8.5.3. Агент координации	210

8.5.4. Агент объяснения.....	210
8.5.5. Агент взаимодействия с оператором	211
8.6. Мультиагентные LLM-системы	212
8.6.1. Распределение ролей в LLM-MAS.....	212
8.6.2. Цикл взаимодействия агентов.....	214
8.6.3. Особенности применения LLM-MAS в ГРВС	215
8.7. LLM в разработке информационных систем.....	216
8.8. LLM в адаптивном управлении.....	219
8.9. Ограничения LLM-агентов.....	222
8.10. Ключевой научный результат	224
Выводы по главе 8.....	226
ГЛАВА 9. АРХИТЕКТУРА АДАПТИВНОЙ СИСТЕМЫ УПРАВЛЕНИЯ ГЕТЕРОГЕННЫМИ РАСПРЕДЕЛЁННЫМИ ВЫЧИСЛИТЕЛЬНЫМИ СИСТЕМАМИ	229
9.1. Общая концепция архитектуры	229
9.2. Уровневая структура системы	232
9.2.1. Уровень инфраструктуры	232
9.2.2. Уровень данных и наблюдаемости.....	233
9.2.3. Уровень адаптивного управления.....	234
9.2.4. Агентный уровень.....	234
9.2.5. Интеллектуальный и когнитивный уровень	235
9.2.6. Уровень принятия решений.....	236
9.3. Контур управления в архитектуре	236
9.4. Агентная структура системы	240
9.5. Распределение функций управления.....	242
9.6. Интеграция LLM-агентов	245
9.7. Механизмы адаптации	247
9.8. Масштабируемость системы.....	250
9.9. Отказоустойчивость	252
9.10. Самоорганизация системы.....	254
9.11. Реализация в современных платформах	256
9.12. Ключевой результат.....	259
Выводы по главе 9.....	260
ГЛАВА 10. ЭКСПЕРИМЕНТАЛЬНАЯ АПРОБАЦИЯ И ОЦЕНКА ЭФФЕКТИВНОСТИ АДАПТИВНОГО УПРАВЛЕНИЯ	263
10.1. Цели и научная постановка эксперимента	263
10.2. Экспериментальная база и методология	266
10.3. Экспериментальные сценарии.....	268
10.4. Метрики эффективности (расширенные).....	271
10.5. Исследуемые методы	273
10.6. Результаты экспериментальных исследований	275

10.6.1. Эффективность адаптивного управления.....	275
10.6.2. Преимущества мультиагентного подхода	277
10.6.3. Эффективность методов на основе ZNN.....	277
10.6.4. Эффективность гибридных методов.....	278
10.6.5. Роль имитационного моделирования	279
10.7. СРАВНИТЕЛЬНЫЙ АНАЛИЗ МЕТОДОВ	280
10.8. Ключевой научный результат главы.....	282
10.9. Связь с предыдущими главами	283
Выводы по главе 10.....	285
ЗАКЛЮЧЕНИЕ	287
БИБЛИОГРАФИЧЕСКИЙ СПИСОК	291
I. РАБОТЫ АВТОРА И НАУЧНОЙ ШКОЛЫ	291
II. ФУНДАМЕНТАЛЬНЫЕ РАБОТЫ ПО РАСПРЕДЕЛЁННЫМ СИСТЕМАМ.....	292
III. АДАПТИВНОЕ УПРАВЛЕНИЕ И ОПТИМИЗАЦИЯ	293
IV. МУЛЬТИАГЕНТНЫЕ СИСТЕМЫ.....	294
V. МАШИННОЕ ОБУЧЕНИЕ И ИНТЕЛЛЕКТУАЛЬНОЕ УПРАВЛЕНИЕ.....	295
VI. ZEROING NEURAL NETWORKS И НЕЙРОСЕТЕВЫЕ МЕТОДЫ АДАПТИВНОГО УПРАВЛЕНИЯ	296
VII. БОЛЬШИЕ ЯЗЫКОВЫЕ МОДЕЛИ, LLM-АГЕНТЫ И КОГНИТИВНЫЕ СИСТЕМЫ УПРАВЛЕНИЯ	298
VIII. СОВРЕМЕННЫЕ ИНТЕЛЛЕКТУАЛЬНЫЕ РАСПРЕДЕЛЕННЫЕ И CLOUD-NATIVE СИСТЕМЫ.....	300

Предисловие

Развитие распределённых вычислительных систем, облачных платформ, интеллектуальных инфраструктур и систем искусственного интеллекта привело к существенному усложнению процессов управления вычислительными ресурсами. Современные вычислительные среды характеризуются высокой степенью распределённости, гетерогенности и динамичности, а также необходимостью функционирования в условиях неполной информации и изменяющихся внешних воздействий.

В течение последних десятилетий методы управления распределёнными вычислительными системами эволюционировали от статических алгоритмических подходов к интеллектуальным адаптивным системам, способным изменять своё поведение в зависимости от состояния среды и накопленного опыта. Существенную роль в этом процессе сыграли развитие мультиагентных технологий, методов машинного обучения и интеллектуального анализа данных.

В последние годы наблюдается новый этап развития интеллектуальных систем управления, связанный с появлением больших языковых моделей и LLM-агентов, обладающих способностью к анализу, рассуждению и генерации решений. Это создаёт предпосылки для перехода к когнитивным системам управления распределёнными вычислительными инфраструктурами.

Настоящая монография является результатом многолетних исследований автора и научной школы в области:

- распределённых вычислительных систем;
- адаптивного управления;
- мультиагентных технологий;
- интеллектуальных методов оптимизации;
- нейросетевых моделей;
- zeroing neural network;
- интеллектуальных агентов.

В работе предпринята попытка сформировать целостный научно-методологический подход к построению адаптивных систем управления гетерогенными распределёнными вычислительными системами, объединяющий классические методы системного анализа, мультиагентные технологии, интеллектуальные методы и современные LLM-агенты.

Автор выражает признательность коллегам, аспирантам и соавторам, принимавшим участие в исследованиях, результаты которых легли в основу данной монографии. Отдельную признательность автор выражает той, чья поддержка, вера и человеческое тепло стали важным источником внутренней опоры и вдохновения в период работы над этой монографией.

Список сокращений и обозначений

Обозначение	Расшифровка
E	управляемая вычислительная среда
N	множество вычислительных узлов
L	множество каналов связи
$Gnet = (N, L)$	граф вычислительной инфраструктуры
$S(t)$	фактическое состояние системы в момент времени t
$\hat{S}(t)$	оценка состояния системы
$Y(t)$	поток наблюдаемых данных
O	функция наблюдения и обработки данных
$X(t)$	управляющее воздействие или распределение ресурсов
$X^*(t)$	согласованное или оптимальное управляющее воздействие
A	алгоритм или совокупность алгоритмов управления
$\mathcal{A}$	множество агентов
a_i	отдельный агент
MAS	мультиагентная система управления
AI	интеллектуальный слой моделей и алгоритмов
LLM	когнитивный слой больших языковых моделей
$\theta(t)$	параметры интеллектуальной модели управления
Φ	функция обучения или адаптации параметров
$H(t)$	история наблюдений и управляющих воздействий
G	система ограничений

Обозначение	Расшифровка
<i>Goals</i>	цели управления
<i>Policies</i>	политики управления и безопасности
<i>ARCH</i>	архитектура адаптивной интеллектуальной системы управления

В настоящей работе прямое и курсивное начертания одноимённых букв рассматриваются как различные обозначения. В частности, *A* в курсивном математическом начертании используется для множества агентов, тогда как прямое *A* — для обозначения алгоритма или совокупности алгоритмов управления; аналогично различаются обозначения для графа, ограничений и функций адаптации.

Введение

Современный этап развития информационных технологий характеризуется стремительным усложнением вычислительных систем, переходом к распределённым архитектурам и повсеместным использованием гетерогенных вычислительных ресурсов. Такие системы включают в себя разнообразные аппаратные платформы, программные среды, коммуникационные протоколы и источники данных, что существенно усложняет процессы их проектирования и управления.

Гетерогенные распределённые вычислительные системы (ГРВС) являются основой функционирования облачных платформ, промышленных информационных систем, систем обработки больших данных, цифровых платформ и интеллектуальных сервисов. Однако увеличение масштабов и сложности таких систем приводит к возникновению принципиально новых задач управления, связанных с необходимостью обеспечения согласованного функционирования разнородных компонентов, оптимального распределения ресурсов и адаптации к динамически изменяющимся условиям эксплуатации.

Классические методы управления, основанные на статических моделях и заранее заданных алгоритмах, оказываются недостаточно эффективными в условиях высокой динамики нагрузки, неопределённости среды и изменчивости структуры системы. Это обуславливает необходимость перехода к адаптивным методам управления, способным обеспечивать самоорганизацию, самонастройку и интеллектуальное принятие решений в реальном времени.

Фундаментальные основы управления гетерогенными системами обработки информации были заложены в ряде работ, посвящённых моделированию и оптимизации распределённых вычислительных систем, в том числе в работах автора [1–3]. В этих исследованиях сформулированы базовые принципы управления ресурсами, предложены методы их оптимизации и обоснована целесообразность использования мультиагентного подхода для построения систем управления.

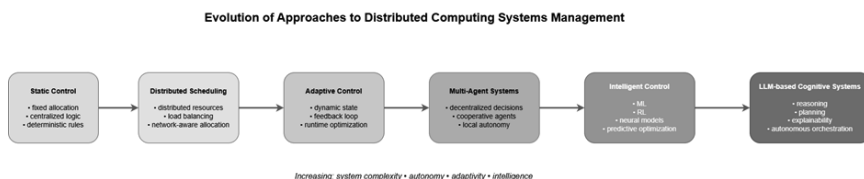


Рисунок 1 – Эволюция подходов к управлению распределёнными вычислительными системами

Так, в работе [1] предложена технология управления гетерогенными системами обработки информации, основанная на декомпозиции системы

на совокупность взаимодействующих компонентов и использовании алгоритмов оптимизации загрузки ресурсов. В дальнейшем данные подходы были развиты в направлении построения мультиагентных систем управления [2], где управление реализуется как результат взаимодействия автономных агентов, отвечающих за различные аспекты функционирования системы. В работе [3] предложены модели и методы оптимизации гетерогенных систем, учитывающие особенности распределённой среды и многокритериальный характер задач управления.

Развитие данных идей привело к формированию научного направления, связанного с адаптивным управлением распределёнными вычислительными системами. В частности, в работах последних лет [4–6] рассматриваются методы адаптивного управления ресурсами, основанные на использовании нейросетевых моделей, методов машинного обучения и интеллектуальных алгоритмов принятия решений. Особое внимание уделяется моделированию динамики системы и учёту изменяющихся условий функционирования.

Существенным этапом эволюции подходов к управлению распределёнными системами стало внедрение мультиагентных технологий. Мультиагентный подход позволяет представить систему управления как совокупность взаимодействующих интеллектуальных агентов, каждый из которых обладает локальными знаниями и целями, но при этом участвует в решении общей задачи управления системой. В работах автора [2, 7] показано, что применение мультиагентного подхода позволяет повысить гибкость, масштабируемость и устойчивость систем управления.

В последние годы наблюдается новый виток развития интеллектуальных методов управления, связанный с использованием больших языковых моделей (LLM) и агентных систем на их основе. LLM-агенты способны выполнять функции анализа, планирования, генерации решений и даже разработки программного обеспечения, что открывает новые возможности для автоматизации процессов управления и проектирования информационных систем.

Интеграция методов адаптивного управления, мультиагентных технологий и интеллектуальных моделей, включая LLM, формирует основу нового поколения систем управления гетерогенными распределёнными вычислительными системами. Такие системы обладают способностью к самообучению, самоорганизации и адаптации к изменяющимся условиям среды, что является ключевым требованием в современных условиях.

Несмотря на значительное количество исследований в области распределённых систем, адаптивного управления и мультиагентных технологий, остаётся нерешённым ряд научных задач, связанных с:

- формализацией гетерогенности как управляемого свойства системы;
- построением универсальных моделей адаптивного управления;
- интеграцией интеллектуальных методов и агентных подходов;
- разработкой архитектур, обеспечивающих масштабируемость и устойчивость управления.

Это определяет актуальность настоящего исследования.

Объект и предмет исследования

Объектом исследования являются гетерогенные распределённые вычислительные системы.

Предметом исследования являются методы, модели и алгоритмы адаптивного управления ресурсами и процессами в гетерогенных распределённых вычислительных системах.

Цель и задачи исследования

Целью монографии является разработка теоретических основ, моделей и методов адаптивного управления гетерогенными распределёнными вычислительными системами, обеспечивающих повышение эффективности их функционирования в условиях динамической и неопределённой среды.

Для достижения поставленной цели необходимо решить следующие задачи:

1. Проанализировать класс гетерогенных распределённых вычислительных систем и особенности их функционирования.
2. Разработать формальные модели представления ресурсов, нагрузки и состояния системы.
3. Исследовать методы адаптивного управления и определить их применимость к рассматриваемому классу систем.
4. Обосновать использование мультиагентного подхода для построения систем управления.
5. Разработать методы распределения и оптимизации ресурсов в условиях динамической нагрузки.
6. Исследовать возможности применения интеллектуальных методов, включая машинное обучение и нейросетевые модели.
7. Разработать архитектуру адаптивной системы управления с использованием мультиагентного подхода.
8. Исследовать возможности применения LLM-агентов в задачах управления и разработки информационных систем.
9. Провести экспериментальную апробацию предложенных методов и оценить их эффективность.

Научная новизна работы заключается в следующем:

- предложена обобщённая модель адаптивного управления гетерогенными распределёнными вычислительными системами;

- разработан мультиагентный подход к реализации адаптивного управления;
- предложены методы интеграции интеллектуальных алгоритмов и нейросетевых моделей в контур управления;
- обосновано использование LLM-агентов как элемента системы управления и разработки;
- разработана архитектура адаптивной системы управления, обеспечивающая масштабируемость и устойчивость.

Практическая значимость

Практическая значимость работы заключается в возможности применения разработанных методов и моделей для:

- управления облачными и распределёнными вычислительными системами;
- оптимизации использования вычислительных ресурсов;
- построения интеллектуальных систем поддержки принятия решений;
- разработки современных информационных систем с использованием агентных технологий.

Структура монографии

Монография состоит из введения, десяти глав и заключения.

ГЛАВА 1. ТЕОРЕТИЧЕСКИЕ ОСНОВЫ УПРАВЛЕНИЯ ГЕТЕРОГЕННЫМИ РАСПРЕДЕЛЁННЫМИ ВЫЧИСЛИТЕЛЬНЫМИ СИСТЕМАМИ

1.1. Понятие и свойства гетерогенных распределённых вычислительных систем

Современные вычислительные системы характеризуются высокой степенью распределённости вычислительных ресурсов, усложнением сетевой инфраструктуры и использованием разнородных аппаратно-программных платформ. Развитие облачных вычислений, периферийных (edge) вычислений, высокопроизводительных кластеров, контейнеризированных сред и интеллектуальных сервисов приводит к формированию вычислительных инфраструктур, обладающих сложной многокомпонентной структурой и высокой динамикой функционирования. В этих условиях особую актуальность приобретает задача формализации и исследования гетерогенных распределённых вычислительных систем как самостоятельного класса сложных технических систем.

Под гетерогенной распределённой вычислительной системой (ГРВС) далее будем понимать совокупность территориально распределённых вычислительных узлов, объединённых коммуникационной сетью и взаимодействующих посредством программных, информационных и сетевых интерфейсов, обладающих различными характеристиками вычислительных, коммуникационных и программных ресурсов.

Характерной особенностью ГРВС является отсутствие однородности как на уровне аппаратного обеспечения, так и на уровне программной среды и механизмов взаимодействия. В отличие от централизованных вычислительных систем, в ГРВС управление осуществляется в условиях неполной наблюдаемости состояния, переменной структуры системы и ограниченной согласованности данных.

К числу основных свойств гетерогенных распределённых вычислительных систем относятся:

- территориальная распределённость, при которой компоненты системы могут быть расположены в различных географических точках и взаимодействовать через каналы связи с различными характеристиками;
- гетерогенность ресурсов, проявляющаяся в различии аппаратных архитектур, вычислительной производительности, операционных систем, программных платформ и протоколов обмена;
- асинхронность взаимодействия, обусловленная отсутствием единого временного контекста и наличием задержек передачи данных;
- динамичность структуры, связанная с изменением состава вычислительных узлов, топологии сети и доступных ресурсов во времени;

– неопределённость состояния системы, возникающая вследствие задержек мониторинга, неполноты информации и стохастического характера вычислительной нагрузки.

Гетерогенность является фундаментальным свойством рассматриваемого класса систем и оказывает непосредственное влияние на процессы распределения ресурсов, балансировки нагрузки, планирования вычислений и обеспечения качества обслуживания. В общем случае она проявляется на нескольких уровнях.

Аппаратная гетерогенность определяется использованием различных типов вычислительных устройств, включая центральные процессоры, графические ускорители, программируемые логические интегральные схемы и специализированные вычислительные модули. Различия в архитектуре и производительности вычислительных устройств требуют применения адаптивных методов распределения задач и учёта особенностей конкретных аппаратных платформ.

Программная гетерогенность связана с использованием различных операционных систем, библиотек, контейнерных сред, систем виртуализации и платформ исполнения. В распределённых средах данное свойство существенно усложняет вопросы совместимости программных компонентов и управления вычислительными сервисами.

Сетевая гетерогенность обусловлена различием характеристик каналов связи, включая пропускную способность, задержки передачи, уровень надёжности и устойчивость сетевой инфраструктуры. В условиях динамической сетевой среды параметры взаимодействия между узлами могут существенно изменяться во времени, что требует адаптивной маршрутизации и интеллектуального управления потоками данных.

Информационная гетерогенность проявляется в различии форматов, структур и моделей представления данных, используемых различными компонентами системы. Наличие неоднородных информационных потоков требует применения механизмов согласования данных, унификации интерфейсов и интеллектуальной обработки информации.

Практика построения современных распределённых вычислительных инфраструктур показывает, что рост гетерогенности приводит к существенному усложнению процессов управления системой. В этих условиях традиционные централизованные методы управления оказываются недостаточно эффективными, поскольку не обеспечивают необходимого уровня адаптивности, масштабируемости и устойчивости к изменениям внешней среды. Это обуславливает необходимость разработки новых подходов к организации управления, основанных на принципах адаптивности, децентрализации и интеллектуальной координации вычислительных ресурсов.

1.2. Управление как информационный процесс

Управление в гетерогенных распределённых вычислительных системах целесообразно рассматривать как информационный процесс, основанный на непрерывном сборе, передаче, обработке и интерпретации данных о состоянии системы и параметрах внешней среды. В условиях распределённости и гетерогенности эффективность функционирования системы в значительной степени определяется качеством информационного обеспечения процессов принятия решений [1–3].

Под управлением далее будем понимать процесс целенаправленного воздействия на систему, осуществляемый на основе информации о её текущем состоянии и ориентированный на достижение заданных характеристик функционирования системы.

В рамках рассматриваемого подхода информация выступает одновременно в нескольких функциональных ролях. Во-первых, информация является ресурсом управления, поскольку именно на её основе формируются управляющие воздействия и осуществляется координация компонентов системы. Во-вторых, информация служит средством взаимодействия между распределёнными узлами и агентами управления, обеспечивая согласование действий в условиях отсутствия централизованного глобального состояния. В-третьих, информация представляет собой результат обработки данных, формируемый в процессе анализа параметров функционирования системы и используемый при выработке решений.

Особенностью распределённых вычислительных систем является высокий уровень неопределённости, сопровождающий процессы управления. Неопределённость возникает вследствие неполноты наблюдаемой информации, задержек передачи данных, динамического изменения состояния системы и воздействия внешней среды [4, 5]. В распределённых средах управляющее решение, как правило, принимается в условиях ограниченной наблюдаемости, когда информация о состоянии отдельных компонентов системы может быть неполной, устаревшей или противоречивой.

С позиций теории управления и теории информации одной из основных задач управления является снижение степени неопределённости состояния системы. В этом контексте ценность информации определяется её способностью уменьшать неопределённость при выборе управляющего воздействия [1, 6]. Следовательно, эффективность управления непосредственно зависит не только от объёма доступных данных, но и от их качества, актуальности и пригодности для принятия решений.

Качество информации в распределённых вычислительных системах определяется совокупностью характеристик, оказывающих влияние на

процессы управления и координации ресурсов. К числу наиболее значимых характеристик относятся:

- своевременность, определяющая соответствие момента получения информации моменту её практического использования;
- полнота, характеризующая достаточность данных для формирования корректного управляющего решения;
- достоверность, отражающая степень соответствия информации реальному состоянию системы;
- полезность, определяющая влияние информации на качество и эффективность принимаемых решений.

В условиях высокодинамичных распределённых систем указанные характеристики приобретают принципиальное значение. Например, даже достоверная информация может оказаться непригодной для управления при значительных задержках её получения. Аналогично неполная или противоречивая информация может приводить к формированию неэффективных стратегий распределения ресурсов и ухудшению качества обслуживания.

Таким образом, управление в гетерогенных распределённых вычислительных системах представляет собой сложный информационно-аналитический процесс, эффективность которого определяется качеством наблюдаемой информации, скоростью её обработки и способностью системы адаптироваться к изменяющимся условиям функционирования [2, 7]. Именно поэтому современные подходы к управлению распределёнными вычислительными средами всё в большей степени опираются на методы интеллектуального анализа данных, адаптивного управления и мультиагентной координации.

1.3. Архитектуры управления распределёнными системами

Организация процессов управления в распределённых вычислительных системах определяется архитектурой взаимодействия управляющих компонентов, механизмами принятия решений и способами координации вычислительных ресурсов. Выбор архитектуры управления оказывает непосредственное влияние на масштабируемость системы, устойчивость к отказам, качество распределения ресурсов и эффективность адаптации к изменениям внешней среды [3, 7, 8].

В современных распределённых вычислительных средах выделяют несколько базовых архитектур управления, различающихся степенью централизации, характером информационного взаимодействия и уровнем автономности управляющих компонентов.

1.3.1. Централизованное управление

При централизованной архитектуре управление системой осуществляется единым управляющим центром, обладающим

информацией о состоянии всех компонентов системы и принимающим решения о распределении ресурсов, маршрутизации задач и координации вычислительных процессов.

Централизованный подход исторически являлся одним из первых методов организации управления вычислительными системами и широко применялся в относительно однородных инфраструктурах с ограниченным числом вычислительных узлов [8, 9]. Основным преимуществом данного подхода является возможность формирования глобально согласованных решений и централизованной оптимизации распределения ресурсов.

К преимуществам централизованного управления относятся:

- относительная простота реализации и сопровождения системы управления;

- возможность использования глобальных критериев оптимизации;

- наличие единой точки принятия решений и контроля состояния системы.

Несмотря на указанные преимущества, централизованные архитектуры обладают рядом существенных ограничений. По мере роста числа вычислительных узлов и увеличения объёма информационных потоков центральный управляющий узел становится потенциальным узким местом системы. Кроме того, отказ центрального компонента управления способен привести к полной потере управляемости распределённой системы.

К основным недостаткам централизованного подхода относятся:

- ограниченная масштабируемость;

- высокая вычислительная и коммуникационная нагрузка на управляющий центр;

- снижение устойчивости системы вследствие наличия единой точки отказа;

- ухудшение адаптивности в условиях динамически изменяющейся среды.

В условиях современных высокодинамичных гетерогенных распределённых вычислительных систем применение полностью централизованных архитектур оказывается ограниченным.

1.3.2. Децентрализованное управление

В децентрализованных архитектурах управление осуществляется множеством независимых компонентов, принимающих решения на основе локальной информации без наличия единого глобального управляющего центра [10, 11].

Данный подход ориентирован на обеспечение высокой автономности компонентов системы и устойчивости к отказам отдельных узлов. Каждый управляющий компонент функционирует независимо, используя собственные механизмы анализа состояния и формирования управляющих воздействий.

К преимуществам децентрализованного управления относятся:

- высокая отказоустойчивость;
- хорошая масштабируемость;
- снижение нагрузки на отдельные компоненты управления;
- возможность локальной адаптации к изменениям среды.

Вместе с тем отсутствие единого координационного механизма приводит к возникновению проблем согласованности решений и потенциальным конфликтам при распределении ресурсов. В ряде случаев локально оптимальные решения могут приводить к ухудшению глобальных характеристик функционирования системы.

Основными недостатками децентрализованного подхода являются:

- сложность координации управляющих воздействий;
- отсутствие гарантии достижения глобального оптимума;
- возможность возникновения конфликтующих решений;
- повышенные требования к механизмам взаимодействия между компонентами системы.

Несмотря на указанные ограничения, децентрализованные архитектуры широко применяются в высокомасштабируемых распределённых средах, где критически важны автономность и устойчивость системы.

1.3.3. Распределённое управление

Распределённая архитектура управления представляет собой компромисс между централизованным и полностью децентрализованным подходами. В таких системах управление осуществляется множеством взаимосвязанных управляющих узлов, обменивающихся информацией и координирующих принимаемые решения [12–14].

В отличие от полностью децентрализованных систем, распределённое управление предполагает наличие механизмов согласования действий и коллективного принятия решений. При этом отдельные управляющие компоненты сохраняют определённую степень автономности.

Характерными особенностями распределённого управления являются:

- наличие множества взаимодействующих управляющих узлов;
- распределённая обработка информации;
- обмен данными между компонентами системы;
- согласование локальных и глобальных целей управления;
- адаптивное перераспределение функций управления.

Распределённые архитектуры обладают значительно более высокой масштабируемостью по сравнению с централизованными системами и обеспечивают лучшую согласованность решений по сравнению с полностью децентрализованными подходами.

Именно распределённое управление в настоящее время рассматривается как наиболее перспективная архитектурная основа для построения гетерогенных распределённых вычислительных систем, функционирующих в условиях высокой динамики, неопределённости и изменяющейся вычислительной нагрузки [3, 12].

1.3.4. Гибридные архитектуры

Развитие облачных платформ, интеллектуальных вычислительных сред и мультиагентных систем привело к появлению гибридных архитектур управления, сочетающих свойства централизованных, распределённых и децентрализованных подходов [14–16].

В гибридных системах управление организуется на нескольких уровнях, различающихся степенью автономности и масштабом принимаемых решений. Как правило, стратегические функции координации реализуются централизованно или иерархически, тогда как локальное управление выполняется автономными компонентами или агентами.

Современные гибридные архитектуры обычно включают:

- иерархические уровни управления;
- локальные механизмы автономного принятия решений;
- глобальные координирующие подсистемы;
- распределённые механизмы обмена информацией;
- интеллектуальные компоненты адаптивного управления.

Использование гибридных архитектур позволяет обеспечить баланс между глобальной согласованностью решений и локальной адаптивностью компонентов системы. Подобный подход особенно эффективен в условиях гетерогенных распределённых вычислительных систем, где необходимо одновременно учитывать ограничения вычислительных ресурсов, динамику сетевой среды и изменяющиеся требования прикладных задач.

В современных исследованиях именно гибридные и мультиагентные архитектуры рассматриваются как наиболее перспективное направление развития интеллектуальных систем управления распределёнными вычислительными инфраструктурами [15–17].

1.4. Основные задачи управления

Управление гетерогенными распределёнными вычислительными системами представляет собой комплекс взаимосвязанных задач, направленных на обеспечение эффективного, устойчивого и адаптивного функционирования вычислительной инфраструктуры в условиях ограниченности ресурсов, изменяющейся нагрузки и неопределённости состояния системы [3, 7, 12].

Специфика ГРВС обуславливает необходимость одновременного учёта вычислительных, коммуникационных и информационных

ограничений, а также динамического характера среды функционирования. В связи с этим процессы управления включают задачи распределения ресурсов, балансировки нагрузки, управления потоками данных и обеспечения устойчивости системы.

1.4.1. Распределение ресурсов

Одной из центральных задач управления является распределение вычислительных ресурсов между задачами и компонентами системы. Содержательно данная задача заключается в определении такого способа назначения вычислительных задач вычислительным узлам, при котором обеспечивается выполнение заданных критериев качества функционирования системы [8, 13].

При распределении ресурсов необходимо учитывать:

- доступные вычислительные мощности узлов;
- ограничения памяти и хранения данных;
- характеристики сетевых каналов связи;
- требования задач к времени выполнения и качеству обслуживания;
- энергетические и эксплуатационные ограничения.

В общем случае задача распределения ресурсов может быть представлена как задача многокритериальной оптимизации:

$$\min F(X), g_i(X) \leq b_i$$

где X — распределение задач по вычислительным ресурсам системы, $F(X)$ — целевая функция управления, а $g_i(X)$ — ограничения, определяющие допустимые режимы функционирования системы.

В качестве критериев оптимизации могут использоваться время выполнения задач, степень загрузки ресурсов, энергопотребление, стоимость вычислений, уровень качества обслуживания и другие показатели эффективности [14, 15].

Особенностью распределения ресурсов в ГРВС является необходимость принятия решений в условиях неполной информации и динамически изменяющейся вычислительной нагрузки. Это требует применения адаптивных методов управления и интеллектуальных алгоритмов оптимизации.

1.4.2. Балансировка нагрузки

Балансировка нагрузки направлена на обеспечение равномерного распределения вычислительной активности между узлами системы и предотвращение локальных перегрузок [9, 16].

Неравномерное распределение нагрузки приводит к снижению общей производительности системы, увеличению времени отклика и ухудшению использования вычислительных ресурсов. В распределённых средах данная задача приобретает особую сложность вследствие различия характеристик вычислительных узлов и динамического изменения интенсивности потоков задач.

В формализованном виде задача балансировки нагрузки может быть представлена следующим образом:

$$\min \max_i L_i$$

где L_i характеризует уровень загрузки i -го вычислительного узла.

Практическая реализация балансировки нагрузки может включать:

- миграцию задач между узлами;
- динамическое перераспределение вычислительных потоков;
- адаптивное масштабирование вычислительных ресурсов;
- изменение маршрутов передачи данных;
- перераспределение контейнеризированных сервисов.

В современных распределённых системах задачи балансировки всё чаще решаются с использованием методов машинного обучения, прогнозирования нагрузки и мультиагентной координации [15–17].

1.4.3. Управление потоками данных

Эффективность функционирования распределённой вычислительной системы в значительной степени определяется характеристиками передачи данных между её компонентами. В связи с этим важной задачей управления является организация и оптимизация информационных потоков [10, 12].

Управление потоками данных включает:

- маршрутизацию информационных потоков;
- управление очередями передачи данных;
- регулирование сетевой нагрузки;
- оптимизацию использования каналов связи;
- обеспечение требуемых параметров качества обслуживания.

В условиях высокодинамичных распределённых сред параметры сетевого взаимодействия могут существенно изменяться во времени. Это требует адаптивного выбора маршрутов передачи данных и интеллектуального управления сетевыми ресурсами.

Особую сложность данная задача приобретает в гетерогенных средах, где каналы связи обладают различной пропускной способностью, задержками передачи и уровнем надёжности. В таких условиях методы статической маршрутизации оказываются недостаточно эффективными, что обуславливает необходимость использования динамических и адаптивных алгоритмов управления сетевыми потоками.

1.4.4. Обеспечение устойчивости

Неотъемлемым свойством современных распределённых вычислительных систем является необходимость функционирования в условиях отказов отдельных компонентов, перегрузок и внешних дестабилизирующих воздействий [11, 18].

Под устойчивостью функционирования далее будем понимать способность системы сохранять работоспособность и обеспечивать

выполнение целевых функций при возникновении внутренних и внешних нарушений.

Обеспечение устойчивости включает решение следующих задач:

- обнаружение отказов и аномальных состояний;
- обеспечение отказоустойчивости компонентов системы;
- резервирование вычислительных и коммуникационных ресурсов;
- восстановление работоспособности после сбоев;
- адаптация структуры системы к изменяющимся условиям функционирования.

В распределённых вычислительных системах устойчивость достигается за счёт сочетания архитектурных, алгоритмических и организационных механизмов. В частности, используются методы репликации данных, резервирования сервисов, распределённого хранения информации и интеллектуального мониторинга состояния компонентов системы.

Развитие адаптивных и мультиагентных подходов позволяет перейти от реактивного восстановления системы к проактивному управлению устойчивостью, основанному на прогнозировании возможных отказов и заблаговременной перестройке структуры вычислительной среды [15–18].

1.5. Ограничения классических подходов

Традиционные методы управления распределёнными вычислительными системами формировались преимущественно для относительно однородных и слабо изменяющихся вычислительных сред. Однако развитие облачных платформ, edge computing, мультиоблачных инфраструктур и интеллектуальных распределённых сервисов привело к существенному усложнению архитектуры вычислительных систем и повышению требований к адаптивности управления [7, 12, 15].

В этих условиях классические подходы к управлению начинают демонстрировать ряд фундаментальных ограничений.

Одним из основных недостатков традиционных методов является жёсткость алгоритмов управления. Во многих классических системах управляющие решения определяются заранее заданными правилами и фиксированными сценариями функционирования. Подобные методы оказываются недостаточно эффективными в условиях изменяющейся вычислительной нагрузки, динамической топологии сети и непредсказуемого поведения распределённой среды.

Другим существенным ограничением является низкая адаптивность. Традиционные алгоритмы, как правило, не обладают механизмами самонастройки и не способны изменять стратегию управления в зависимости от состояния системы и внешних факторов. В результате эффективность управления существенно снижается при возникновении нестандартных режимов функционирования.

Серьёзной проблемой также является ограниченная масштабируемость. По мере увеличения числа вычислительных узлов и объёма информационных потоков сложность централизованного управления возрастает нелинейно. Это приводит к росту вычислительных затрат, увеличению задержек принятия решений и ухудшению согласованности управляющих воздействий [9, 13].

Дополнительные трудности связаны с обработкой неопределённости. Классические методы управления ориентированы преимущественно на работу в условиях полной или квазиполной информации, тогда как реальные распределённые вычислительные системы функционируют в условиях задержек мониторинга, неполной наблюдаемости и стохастического изменения параметров среды [5, 11].

Проведённые исследования показывают, что преодоление указанных ограничений требует перехода к адаптивным, распределённым и интеллектуальным методам управления, основанным на принципах мультиагентной координации, динамической оптимизации и непрерывного анализа состояния системы [2–4, 15].

Одной из ключевых задач управления ГРВС является задача распределения вычислительных ресурсов. В общем виде она может быть представлена как задача оптимизации с системой ограничений:

$$\min F(X), g_i(X) \leq b_i$$

где X представляет распределение задач по вычислительным ресурсам системы, $F(X)$ — целевую функцию управления, $g_i(X)$ — ограничения, определяющие допустимые режимы функционирования системы, а b_i — допустимые значения соответствующих ограничений.

В распределённых вычислительных системах ограничения имеют многокомпонентный характер и отражают физические, логические, сетевые и организационные аспекты функционирования инфраструктуры.

1.5.1. Ограничения вычислительных ресурсов

Наиболее очевидный класс ограничений связан с ограниченностью вычислительных ресурсов отдельных узлов системы. Для каждого вычислительного узла суммарное потребление ресурсов назначенными задачами не должно превышать доступную производительность оборудования.

Для вычислительных ресурсов центрального процессора ограничение может быть представлено следующим образом:

$$g_j^{CPU}(X) = \sum_{k \in T_j} c_k \leq C_j$$

где T_j — множество задач, назначенных узлу j , c_k — потребность задачи k в вычислительных ресурсах, а C_j — доступная вычислительная мощность узла.

Аналогичные ограничения формируются для оперативной памяти, графических ускорителей, дисковых подсистем и других типов вычислительных ресурсов.

1.5.2. Ограничения пропускной способности каналов связи

В распределённых вычислительных системах существенное влияние на производительность оказывают параметры сетевого взаимодействия. Ограничения данного типа определяются пропускной способностью каналов передачи данных:

$$g_{ij}^{net}(X) = \sum_{k \in D_{ij}} d_k \leq B_{ij}$$

где D_{ij} обозначает поток данных между узлами i и j , d_k — объём передаваемых данных, а B_{ij} — пропускная способность канала связи.

Сетевые ограничения особенно критичны для распределённых сред с интенсивным межузловым обменом данными, поскольку коммуникационные задержки могут существенно снижать эффективность распределённых вычислений.

1.5.3. Ограничения времени выполнения

Во многих прикладных задачах необходимо обеспечивать выполнение требований качества обслуживания (QoS), связанных с ограничением времени обработки задач:

$$g_k^{time}(X) = t_k(X) \leq T_k^{max}$$

где $t_k(X)$ — время выполнения задачи, а T_k^{max} — максимально допустимое время обработки.

Подобные ограничения характерны для систем реального времени, облачных сервисов и критически важных вычислительных приложений.

1.5.4. Ограничения размещения

В ряде случаев выполнение задач допускается только на определённых вычислительных узлах. Это может быть связано с локализацией данных, требованиями безопасности или географическими ограничениями.

Соответствующее ограничение может быть представлено в виде:

$$g_k^{loc}(X) \in S_k$$

где S_k определяет допустимое множество узлов для выполнения задачи k .

1.5.5. Ограничения совместимости

Гетерогенность вычислительной среды приводит к необходимости учитывать совместимость задач с архитектурой конкретных вычислительных узлов, операционными системами и средами исполнения.

Формальное ограничение совместимости может быть задано следующим образом:

$$g_k^{comp}(X) = \begin{cases} 1, & \text{если узел поддерживает среду выполнения} \\ 0, & \text{иначе} \end{cases}$$

Подобные ограничения особенно важны в мультиархитектурных и контейнеризированных вычислительных средах.

1.5.6. Энергетические ограничения

Для современных вычислительных инфраструктур существенное значение приобретают ограничения, связанные с энергопотреблением системы:

$$g^{energy}(X) = \sum_j E_j(X) \leq E^{max}$$

где $E_j(X)$ характеризует энергопотребление отдельных компонентов системы.

Энергетические ограничения являются критически важными для edge computing, мобильных вычислительных платформ и энергоэффективных центров обработки данных [14, 16].

1.5.7. Ограничения надёжности

Распределённые вычислительные системы должны обеспечивать требуемый уровень надёжности и устойчивости функционирования:

$$g^{rel}(X) \geq R^{min}$$

где $R(X)$ характеризует надёжность конфигурации системы, а R^{min} — минимально допустимый уровень надёжности.

Данный класс ограничений связан с резервированием ресурсов, репликацией данных и обеспечением отказоустойчивости вычислительной среды.

1.5.8. Ограничения согласованности данных

В распределённых системах важное значение имеют ограничения, связанные с согласованностью информации между компонентами системы:

$$g^{cons}(X) \leq \delta$$

где δ определяет допустимый уровень рассогласованности данных.

Подобные ограничения возникают в распределённых транзакционных системах, облачных хранилищах и сервисах коллективной обработки информации.

Таким образом, совокупность ограничений распределённой вычислительной системы может быть представлена в следующем виде:

$$G(X) = \{g_1(X), g_2(X), \dots, g_n(X)\}$$

При этом ограничения относятся к различным классам: ресурсным, сетевым, временным, логическим и организационным.

Принципиально важной особенностью гетерогенных распределённых вычислительных систем является динамический характер ограничений. В отличие от классических моделей, в современных распределённых средах ограничения зависят не только от распределения ресурсов, но и от текущего состояния системы:

$$g_i = g_i(X, S(t))$$

где $S(t)$ характеризует состояние системы во времени.

Данное обстоятельство имеет фундаментальное значение для построения адаптивных систем управления. Если в традиционных моделях ограничения считаются фиксированными, то в современных распределённых вычислительных средах они изменяются динамически под воздействием вычислительной нагрузки, состояния сети, отказов оборудования и внешних факторов [12, 15, 17].

Следовательно, задача управления ГРВС должна рассматриваться как задача адаптивной оптимизации в условиях неопределённости и изменяющейся структуры ограничений. Это требует применения интеллектуальных методов управления, способных непрерывно

анализировать состояние системы и корректировать стратегию распределения ресурсов в процессе функционирования вычислительной среды.

Выводы по главе 1

Проведённый анализ показал, что современные гетерогенные распределённые вычислительные системы представляют собой сложные динамические среды, характеризующиеся высокой степенью распределённости, неоднородностью вычислительных и коммуникационных ресурсов, изменчивостью структуры и наличием неопределённости состояния [7, 12, 15]. Развитие облачных, периферийных и интеллектуальных вычислительных платформ приводит к дальнейшему усложнению процессов управления распределённой инфраструктурой и повышению требований к адаптивности управляющих механизмов.

Установлено, что управление гетерогенными распределёнными вычислительными системами следует рассматривать как информационный процесс, основанный на непрерывном сборе, обработке и интерпретации данных о состоянии системы и внешней среды. Эффективность управления в значительной степени определяется качеством информации, степенью наблюдаемости системы и способностью механизмов управления функционировать в условиях неполной и изменяющейся информации [1, 5].

Показано, что классические централизованные и жёстко детерминированные методы управления обладают рядом фундаментальных ограничений, связанных с недостаточной масштабируемостью, низкой адаптивностью и ограниченной устойчивостью к неопределённости и динамическим изменениям вычислительной среды [8, 11]. По мере роста сложности распределённых вычислительных инфраструктур эффективность подобных подходов существенно снижается.

Выполненная формализация ограничений задачи распределения ресурсов позволила показать, что ограничения в ГРВС носят многомерный и динамический характер и охватывают вычислительные, сетевые, временные, энергетические, логические и организационные аспекты функционирования системы. Принципиально важным является то обстоятельство, что ограничения зависят не только от распределения ресурсов, но и от текущего состояния системы во времени, что требует перехода от статических моделей управления к адаптивным методам оптимизации.

Таким образом, проведённый анализ подтверждает необходимость разработки новых подходов к управлению гетерогенными распределёнными вычислительными системами, основанных на принципах

адаптивности, распределённого принятия решений, интеллектуальной обработки информации и мультиагентной координации ресурсов [3, 14–17]. Именно данные подходы рассматриваются в последующих главах монографии как основа построения современных интеллектуальных систем управления распределёнными вычислительными средами.

ГЛАВА 2. МОДЕЛИ И ФОРМАЛЬНОЕ ПРЕДСТАВЛЕНИЕ ГЕТЕРОГЕННЫХ РАСПРЕДЕЛЁННЫХ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ

2.1. Общая формализация системы

Для исследования процессов адаптивного управления гетерогенными распределёнными вычислительными системами необходимо формальное описание структуры системы, её компонентов и взаимодействий между ними. Формализация позволяет перейти от описательного представления вычислительной среды к построению математических моделей, пригодных для анализа, оптимизации и синтеза управляющих воздействий [12–14].

Гетерогенная распределённая вычислительная система далее рассматривается как совокупность взаимосвязанных вычислительных, коммуникационных и информационных компонентов, функционирующих в условиях изменяющейся нагрузки и динамической структуры. В общем виде система может быть представлена следующим кортежем:

$$S = \langle N, L, R, T, D, S(t) \rangle$$

где:

- N — множество вычислительных узлов системы;
- L — множество каналов связи между узлами;
- R — множество вычислительных и коммуникационных ресурсов;
- T — множество выполняемых задач;
- D — множество потоков данных;
- $S(t)$ — состояние системы во времени.

Множество вычислительных узлов N определяет аппаратную основу распределённой системы и включает серверы, виртуальные машины, контейнеризированные сервисы, edge-устройства, облачные ресурсы и специализированные вычислительные модули. Каждый узел характеризуется набором параметров, определяющих его вычислительные возможности, архитектурные особенности и доступные ресурсы [8, 15].

Множество каналов связи L описывает коммуникационную структуру системы и определяет топологию взаимодействия между узлами. Каналы связи могут различаться по пропускной способности, задержкам передачи, надёжности и устойчивости к внешним воздействиям. В условиях распределённой среды характеристики сетевых соединений оказывают существенное влияние на производительность системы и эффективность распределения задач.

Множество ресурсов R включает вычислительные, сетевые, энергетические и информационные ресурсы системы. В рамках рассматриваемой модели ресурсы могут быть как локальными, принадлежащими отдельным узлам, так и распределёнными между несколькими компонентами системы. Гетерогенность ресурсов является

одной из ключевых особенностей рассматриваемого класса систем и требует использования адаптивных механизмов управления.

Множество задач T определяет совокупность вычислительных процессов, выполняемых в системе. Каждая задача характеризуется набором параметров, включая вычислительную сложность, объём используемых данных, требования к времени выполнения и ограничения на размещение. В реальных распределённых средах состав множества задач изменяется во времени, что приводит к динамическому изменению вычислительной нагрузки.

Множество потоков данных D описывает процессы обмена информацией между компонентами системы. Потоки данных определяют интенсивность межузловое взаимодействия и оказывают непосредственное влияние на загрузку сетевой инфраструктуры и производительность распределённых вычислений.

Особое значение имеет состояние системы $S(t)$, характеризующее совокупность параметров функционирования системы в момент времени t . Вектор состояния может включать:

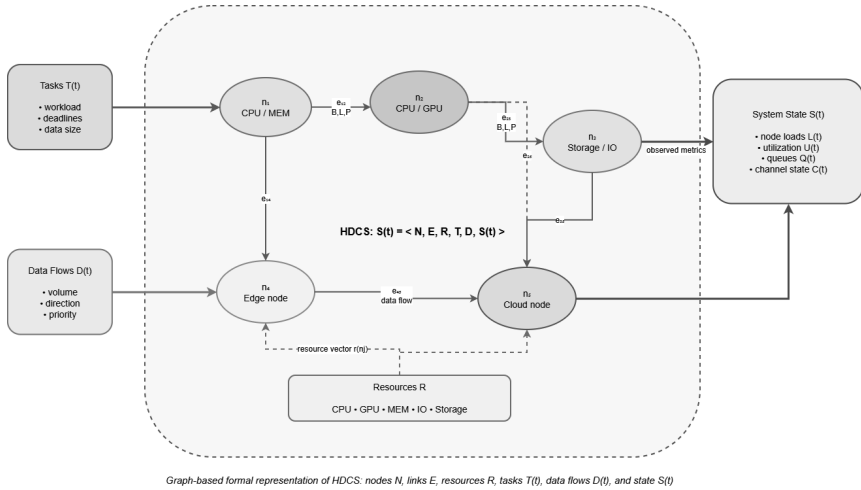
- уровень загрузки вычислительных узлов;
- состояние сетевых каналов;
- распределение задач;
- параметры энергопотребления;
- характеристики информационных потоков;
- показатели качества обслуживания.

В отличие от статических моделей, состояние распределённой вычислительной системы изменяется во времени под воздействием внешней нагрузки, внутренних процессов и управляющих воздействий. Следовательно, задача управления должна рассматриваться как задача управления динамической системой с изменяющейся структурой и неполной наблюдаемостью состояния [5, 11].

Представленная формализация является базовой моделью, используемой в последующих разделах для описания процессов распределения ресурсов, построения графовых моделей системы, формализации ограничений и разработки адаптивных методов управления.

Рисунок 2 иллюстрирует формальную структуру гетерогенной распределённой вычислительной системы и взаимосвязи между её основными компонентами.

Formal Model of a Heterogeneous Distributed Computing System



Graph-based formal representation of HDCS: nodes N , links E , resources R , tasks $T(t)$, data flows $D(t)$, and state $S(t)$

Рисунок 2 – Формальная модель ГРВС

2.2. Модель вычислительных узлов

Вычислительные узлы являются базовыми элементами гетерогенной распределённой вычислительной системы, на которых выполняются прикладные задачи, размещаются сервисы и обрабатываются потоки данных. Множество вычислительных узлов системы обозначим следующим образом:

$$N = \{n_1, n_2, \dots, n_m\},$$

где m — число узлов, входящих в состав распределённой вычислительной системы.

Каждый узел $n_j \in N$ характеризуется набором аппаратных, программных и коммуникационных параметров. В упрощённом виде ресурсное состояние узла может быть задано вектором:

$$r(n_j) = (CPU_j, MEM_j, GPU_j, IO_j),$$

где CPU_j — вычислительная мощность узла, MEM_j — объём доступной оперативной памяти, GPU_j — наличие и производительность графических или иных специализированных ускорителей, IO_j — пропускная способность подсистемы ввода-вывода.

В зависимости от решаемой задачи данный вектор может быть расширен дополнительными характеристиками: объёмом дискового хранилища, энергопотреблением, текущей загрузкой, стоимостью

использования ресурса, надёжностью узла, архитектурой процессора, типом операционной системы и доступными средами исполнения. Такое расширение особенно важно при моделировании облачных, периферийных и контейнеризированных вычислительных сред [8, 12, 15].

Гетерогенность системы проявляется в том, что различные узлы обладают несовпадающими ресурсными характеристиками:

$$r(n_i) \neq r(n_j), i \neq j.$$

Данное соотношение отражает принципиальное отличие ГРВС от однородных вычислительных систем. В таких средах один и тот же набор задач может выполняться с различной производительностью на разных узлах, а отдельные задачи могут быть допустимы к выполнению только на части вычислительных ресурсов вследствие требований к архитектуре, памяти, ускорителям или программной среде.

Для задач адаптивного управления важно учитывать не только статические характеристики узлов, но и их текущее состояние. Поэтому ресурсный вектор узла может рассматриваться как функция времени:

$$r(n_j, t) = (CPU_j(t), MEM_j(t), GPU_j(t), IO_j(t)).$$

В этом случае параметры узла отражают не только номинальные возможности оборудования, но и фактическую доступность ресурсов в момент времени t . Например, вычислительный узел с высокой номинальной производительностью может быть неэффективен для назначения новой задачи при высокой текущей загрузке или ограниченной сетевой доступности.

Таким образом, модель вычислительного узла должна учитывать как структурную гетерогенность ресурсов, так и динамику их использования. Это позволяет более корректно формализовать задачи распределения вычислительной нагрузки, балансировки ресурсов и адаптивного управления в условиях изменяющейся среды [3, 13, 16].

2.3. Графовая модель системы

Коммуникационная структура гетерогенной распределённой вычислительной системы может быть представлена в виде ориентированного графа, вершины которого соответствуют вычислительным узлам, а рёбра — каналам связи между ними [12–14]. Такое представление позволяет формализовать сетевое взаимодействие, анализировать маршруты передачи данных, учитывать ограничения пропускной способности и строить алгоритмы распределённого управления.

Графовая модель системы задаётся следующим образом:

$$G_{net} = (N, L),$$

где $N = \{n_1, \dots, n_m\}$ — множество вершин графа, соответствующих вычислительным узлам системы, а $L \subseteq N \times N$ — множество каналов связи между узлами.

Ориентированность графа позволяет учитывать асимметрию сетевого взаимодействия, характерную для реальных распределённых сред. В частности, пропускная способность, задержка передачи и надёжность канала от узла n_i к узлу n_j могут отличаться от аналогичных характеристик канала в обратном направлении. Это особенно важно для облачных, периферийных и гибридных вычислительных инфраструктур, в которых сетевые соединения могут иметь различную природу и качество [8, 15].

Каждому ребру $l_{ij} \in L$, соединяющему узлы n_i и n_j , сопоставляется весовая характеристика:

$$w(l_{ij}) = (B_{ij}, S_{ij}, P_{ij}),$$

где B_{ij} — пропускная способность канала связи, S_{ij} — задержка передачи данных, P_{ij} — вероятность потери данных или показатель ненадёжности канала.

В зависимости от целей моделирования вектор $w(l_{ij})$ может быть расширен дополнительными параметрами: стоимостью передачи данных, уровнем загрузки канала, доступностью соединения, энергетическими затратами передачи и приоритетами сетевого трафика. Такое расширение позволяет использовать графовую модель не только для описания топологии, но и для решения задач оптимизации потоков данных и адаптивной маршрутизации.

В динамических распределённых системах характеристики каналов связи изменяются во времени. Поэтому вес ребра целесообразно рассматривать как функцию времени:

$$w(l_{ij}, t) = (B_{ij}(t), S_{ij}(t), P_{ij}(t)).$$

Данная запись отражает тот факт, что фактическое состояние сетевой инфраструктуры зависит от текущей нагрузки, отказов, перегрузок, изменений маршрутов и внешних воздействий. Следовательно, управление потоками данных в ГРВС должно учитывать не только статическую топологию сети, но и динамику изменения сетевых характеристик.

Графовая модель также позволяет описать связность системы, наличие альтернативных маршрутов, критические каналы связи и потенциальные точки отказа. Это имеет важное значение для задач балансировки нагрузки, обеспечения отказоустойчивости и построения адаптивных стратегий распределения вычислений [10, 16].

Таким образом, представление ГРВС в виде ориентированного взвешенного графа является удобной формальной основой для анализа сетевого взаимодействия, моделирования потоков данных и разработки

методов адаптивного управления распределённой вычислительной инфраструктурой.

2.4. Модель задач

В гетерогенной распределённой вычислительной системе прикладная нагрузка представляется множеством задач, подлежащих выполнению на доступных вычислительных узлах. Множество задач обозначим следующим образом:

$$T = \{t_1, t_2, \dots, t_k\},$$

где k — количество задач, поступивших в систему или находящихся в обработке в рассматриваемый момент времени.

Каждая задача $t_i \in T$ характеризуется набором параметров, определяющих требования к вычислительным, информационным и временным ресурсам. В базовой форме задача может быть представлена вектором:

$$t_i = (c_i, m_i, d_i, \tau_i),$$

где c_i — вычислительная сложность задачи, m_i — потребность в оперативной памяти, d_i — объём входных и/или передаваемых данных, τ_i — требуемое или максимально допустимое время выполнения задачи.

Параметр c_i отражает объём вычислений, необходимый для завершения задачи, и может задаваться в условных вычислительных операциях, процессорном времени, количестве инструкций или иных единицах, используемых в конкретной модели. Параметр m_i определяет минимальный объём памяти, необходимый для корректного выполнения задачи. Параметр d_i характеризует объём данных, который должен быть загружен, передан между узлами или сохранён в процессе выполнения. Параметр τ_i связан с требованиями к времени выполнения и может интерпретироваться как дедлайн, ограничение качества обслуживания или допустимый интервал обработки [8, 12, 15].

В более общем случае модель задачи может быть расширена за счёт включения дополнительных характеристик: приоритета, стоимости выполнения, требований к специализированным ускорителям, ограничений размещения, требований безопасности, зависимости от других задач и параметров энергопотребления. Такое расширение особенно важно для распределённых сред, в которых задачи различаются не только объёмом вычислений, но и требованиями к данным, сетевому взаимодействию и среде исполнения [13, 16].

Для динамических систем множество задач следует рассматривать как изменяющееся во времени:

$$T = T(t).$$

Это означает, что в систему непрерывно поступают новые задачи, часть задач завершается, а часть может изменять свой статус в процессе

выполнения. В результате вычислительная нагрузка становится нестационарной, что существенно усложняет задачу управления ресурсами.

Важной особенностью распределённых вычислительных систем является наличие зависимостей между задачами. В этом случае задачи могут образовывать ориентированный граф зависимостей, в котором вершины соответствуют задачам, а рёбра отражают необходимость передачи данных или соблюдения порядка выполнения. Наличие таких зависимостей приводит к необходимости учитывать не только индивидуальные характеристики задач, но и структуру вычислительного процесса в целом.

Таким образом, модель задачи должна учитывать вычислительную сложность, требования к памяти, объём данных и временные ограничения, а также при необходимости расширяться характеристиками совместимости, приоритета и взаимозависимости. Такое представление создаёт основу для формализации задачи распределения вычислительной нагрузки и построения адаптивных алгоритмов управления ГРВС.

2.5. Отображение распределения задач

Распределение вычислительных задач между узлами ГРВС задаёт соответствие между множеством задач T и множеством вычислительных узлов N . В простейшем случае каждая задача назначается одному вычислительному узлу, на котором она должна быть выполнена. Такое распределение можно представить в виде отображения:

$$X: T \rightarrow N.$$

Данное отображение определяет, какому узлу $n_j \in N$ соответствует задача $t_i \in T$. В контексте управления ГРВС отображение X является одной из ключевых управляемых переменных, поскольку именно от выбора распределения задач зависят загрузка узлов, время выполнения, объём сетевого обмена и качество обслуживания [8, 13, 15].

Для задач оптимизации распределение удобно задавать в матричной форме:

$$X = \{x_{ij}\},$$

$$\text{где } x_{ij} = \begin{cases} 1, & \text{если задача } t_i \text{ назначена узлу } n_j, \\ 0, & \text{иначе.} \end{cases}$$

В этом случае каждая строка матрицы X соответствует отдельной задаче, а каждый столбец — вычислительному узлу. Значение $x_{ij} = 1$ означает, что задача t_i размещена на узле n_j , а значение $x_{ij} = 0$ означает отсутствие такого назначения.

Если предполагается, что каждая задача выполняется только на одном узле, то для каждой задачи должно выполняться условие единственности назначения:

$$\sum_j x_{ij} = 1.$$

Это условие гарантирует, что задача t_i будет назначена ровно одному вычислительному узлу. В более сложных моделях, связанных с параллельным выполнением задач или декомпозицией вычислительного процесса, данное ограничение может быть ослаблено или заменено условиями распределённого выполнения, когда одна задача разбивается на несколько подзадач и размещается на нескольких узлах.

Матричное представление распределения задач удобно тем, что позволяет непосредственно включать X в целевые функции и системы ограничений. Например, через элементы матрицы x_{ij} могут быть выражены загрузка узлов, использование памяти, объём межузлового обмена, стоимость выполнения задач и показатели качества обслуживания.

В динамической среде распределение задач также изменяется во времени:

$$X = X(t).$$

Это означает, что управляющая система должна не только формировать начальное назначение задач, но и корректировать его в процессе функционирования ГРВС с учётом изменения нагрузки, доступности ресурсов, состояния сети и требований к качеству обслуживания. Такая постановка переводит задачу распределения задач из статической оптимизационной задачи в задачу адаптивного управления ресурсами распределённой вычислительной системы [3, 12, 16].

2.6. Модель потоков данных

В распределённых вычислительных системах выполнение задач, как правило, сопровождается передачей данных между вычислительными узлами. Это связано с необходимостью загрузки входных данных, обмена промежуточными результатами, передачи управляющих сообщений и сохранения результатов обработки. Поэтому при формализации ГРВС наряду с моделью задач и моделью вычислительных ресурсов необходимо учитывать модель потоков данных [8, 12, 15].

Множество потоков данных обозначим следующим образом:

$$D = \{d_{pq}\},$$

где поток d_{pq} соответствует передаче данных между задачами t_p и t_q . Такой поток возникает в случае, если выполнение задачи t_q зависит от данных, формируемых или используемых задачей t_p .

Каждому потоку d_{pq} сопоставляется объём передаваемых данных:

$$V_{pq},$$

который характеризует интенсивность информационного взаимодействия между соответствующими задачами. В зависимости от прикладной области величина V_{pq} может измеряться в байтах, пакетах, сообщениях или условных единицах информационного обмена.

Если задачи t_p и t_q размещены на различных вычислительных узлах, то поток d_{pq} создаёт нагрузку на один или несколько каналов связи графа $G_{net} = (N, L)$. В этом случае необходимо учитывать маршрут передачи данных между узлами и влияние данного потока на состояние сетевой инфраструктуры.

Нагрузка на канал l_{ij} , соединяющий узлы n_i и n_j , может быть представлена следующим образом:

$$Load_{ij} = \sum_{(p,q)} V_{pq} \cdot \delta_{ij}^{pq},$$

где δ_{ij}^{pq} — индикатор использования канала l_{ij} при передаче потока данных между задачами t_p и t_q . Если соответствующий поток проходит через канал l_{ij} , то $\delta_{ij}^{pq} = 1$, в противном случае $\delta_{ij}^{pq} = 0$.

Данная модель позволяет связать распределение задач X с сетевой нагрузкой. При изменении размещения задач меняются маршруты передачи данных, а следовательно, изменяется нагрузка на отдельные каналы связи. Это обстоятельство особенно важно для гетерогенных распределённых систем, в которых пропускная способность каналов, задержки и надёжность соединений могут существенно различаться [10, 13, 16].

В более общем случае индикатор δ_{ij}^{pq} может зависеть от выбранного маршрута, состояния сети и политики передачи данных:

$$\delta_{ij}^{pq} = \delta_{ij}^{pq}(X, S(t)).$$

Это означает, что сетевые потоки зависят не только от структуры зависимостей между задачами, но и от текущего распределения задач по узлам, состояния сетевой инфраструктуры и управляющих решений системы.

Модель потоков данных используется при формировании сетевых ограничений, оценке задержек выполнения задач, выборе маршрутов передачи данных и построении адаптивных алгоритмов распределения ресурсов. Включение потоков данных в общую модель ГРВС позволяет учитывать не только вычислительную, но и коммуникационную составляющую производительности распределённой системы.

2.7. Состояние системы

Для построения адаптивных методов управления необходимо ввести формальное представление состояния гетерогенной распределённой вычислительной системы. Состояние системы отражает совокупность

параметров, определяющих текущий режим её функционирования, доступность ресурсов, характер вычислительной нагрузки и условия сетевого взаимодействия [5, 12, 15].

В общем виде состояние ГРВС в момент времени t можно представить в виде вектора:

$$S(t) = (L(t), U(t), Q(t), C(t)),$$

где $L(t)$ — вектор нагрузки вычислительных узлов, $U(t)$ — вектор использования ресурсов, $Q(t)$ — состояние очередей задач, $C(t)$ — состояние каналов связи.

Компонента $L(t)$ характеризует распределение вычислительной нагрузки между узлами системы. Она может включать показатели загрузки процессоров, графических ускорителей, подсистем хранения и иных вычислительных компонентов. Анализ $L(t)$ позволяет выявлять перегруженные и недоиспользуемые узлы, а также принимать решения о перераспределении задач.

Компонента $U(t)$ описывает фактическое использование ресурсов системы. В отличие от номинальных характеристик узлов, заданных в ресурсных векторах $r(n_j)$, величина $U(t)$ отражает текущее состояние ресурсов с учётом уже выполняемых задач, фоновой нагрузки, резервирования и ограничений доступности. Данная компонента особенно важна для оценки возможности назначения новых задач и предотвращения превышения ресурсных ограничений.

Компонента $Q(t)$ определяет состояние очередей задач. Она может включать длину очередей, приоритеты задач, время ожидания, число задач с нарушением временных ограничений и интенсивность поступления новых заявок. Состояние очередей является одним из основных индикаторов перегрузки системы и используется при выборе стратегий балансировки нагрузки и планирования выполнения задач.

Компонента $C(t)$ характеризует состояние коммуникационной подсистемы. Она включает текущую загрузку каналов связи, задержки передачи данных, доступность соединений, вероятность потерь и параметры маршрутизации. Учёт $C(t)$ позволяет принимать решения, учитывающие не только вычислительные ресурсы узлов, но и возможности передачи данных между ними.

В расширенной модели вектор состояния может включать дополнительные параметры: энергопотребление, стоимость использования ресурсов, уровень надёжности компонентов, показатели качества обслуживания, состояние сервисов и результаты мониторинга аномалий. Состав вектора состояния определяется целями моделирования и уровнем детализации системы управления [13, 16].

Важной особенностью состояния ГРВС является его динамический характер. Под воздействием новых задач, завершения вычислений,

изменения сетевых условий и управляющих воздействий состояние системы непрерывно изменяется:

$$S(t) \rightarrow S(t + 1).$$

Поэтому задача управления заключается не только в выборе оптимального распределения ресурсов в отдельный момент времени, но и в формировании последовательности управляющих воздействий, обеспечивающих требуемые характеристики функционирования системы на временном интервале.

Следует также учитывать, что состояние системы в реальных условиях наблюдается не полностью. Управляющая подсистема оперирует не истинным состоянием $S(t)$, а его оценкой $\hat{S}(t)$, сформированной на основе данных мониторинга, логов, метрик и трассировок:

$$\hat{S}(t) = O(Y(t)),$$

где $Y(t)$ — множество наблюдаемых данных, а O — функция обработки и агрегации информации.

Таким образом, модель состояния системы является центральным элементом формального описания ГРВС. Она связывает вычислительные ресурсы, задачи, сетевые потоки и наблюдаемые данные в единую основу для построения адаптивного управления, прогнозирования нагрузки и принятия решений в условиях неопределённости.

2.8. Формализация ограничений

В главе 1 была рассмотрена общая структура ограничений, возникающих при управлении гетерогенными распределёнными вычислительными системами. После введения формальных моделей вычислительных узлов, задач, потоков данных и состояния системы становится возможным представить данные ограничения в строгой математической форме и связать их с переменными распределения задач X и состоянием системы $S(t)$.

Ограничения определяют множество допустимых состояний системы и допустимых управляющих воздействий. Именно они задают область существования решений задачи распределения ресурсов и обеспечивают корректность функционирования распределённой вычислительной среды [12–15].

2.8.1. Ресурсные ограничения

Наиболее важный класс ограничений связан с ограниченностью вычислительных ресурсов отдельных узлов системы. Для каждого узла суммарное потребление ресурсов назначенными задачами не должно превышать доступные возможности оборудования.

Ограничение на использование вычислительных ресурсов центрального процессора имеет вид:

$$\sum_i x_{ij} \cdot c_i \leq CPU_j,$$

где x_{ij} определяет назначение задачи t_i узлу n_j , c_i — вычислительная сложность задачи, а CPU_j — доступная вычислительная мощность узла.

Аналогично ограничение на использование оперативной памяти может быть представлено следующим образом:

$$\sum_i x_{ij} \cdot m_i \leq MEM_j,$$

где m_i характеризует потребность задачи в памяти, а MEM_j — доступный объём оперативной памяти вычислительного узла.

Подобные ограничения могут быть сформированы также для графических ускорителей, подсистем хранения данных, сетевых интерфейсов и иных типов ресурсов. В условиях гетерогенности параметры доступных ресурсов различаются для разных узлов системы, что приводит к необходимости учитывать индивидуальные характеристики каждого вычислительного элемента [8, 13].

2.8.2. Сетевые ограничения

Поскольку выполнение распределённых задач сопровождается межузловым обменом данными, важное значение приобретают ограничения, связанные с пропускной способностью каналов связи.

Для канала e_{ij} соответствующее ограничение может быть представлено в виде:

$$Load_{ij} \leq B_{ij},$$

где $Load_{ij}$ — текущая нагрузка на канал связи, а B_{ij} — его пропускная способность.

Нарушение данного ограничения приводит к перегрузке сетевой инфраструктуры, росту задержек передачи данных и ухудшению качества обслуживания. В распределённых вычислительных средах сетевые ограничения зачастую оказываются не менее значимыми, чем ограничения вычислительных ресурсов.

2.8.3. Временные ограничения

Для многих прикладных задач необходимо выполнение требований к времени обработки и качеству обслуживания. Соответствующее ограничение имеет вид:

$$t_i(X) \leq \tau_i,$$

где $t_i(X)$ — фактическое время выполнения задачи при заданном распределении ресурсов, а τ_i — максимально допустимое время выполнения.

Временные ограничения особенно важны для систем реального времени, потоковой обработки данных, облачных сервисов и критически важных вычислительных приложений [10, 16].

2.8.4. Обобщённая форма ограничений

С учётом введённых моделей совокупность ограничений может быть представлена в обобщённой форме:

$$g_i(X, S(t)) \leq b_i,$$

где g_i — функция ограничения, зависящая от распределения задач X и состояния системы $S(t)$, а b_i — допустимое значение ограничения.

Принципиально важной особенностью данной записи является зависимость ограничений не только от самого распределения ресурсов, но и от текущего состояния системы. В традиционных моделях ограничения обычно рассматриваются как фиксированные параметры, однако в реальных гетерогенных распределённых средах они изменяются во времени под воздействием вычислительной нагрузки, сетевой активности, отказов компонентов и внешних факторов [5, 12, 15].

Например, доступная вычислительная мощность узла может уменьшаться вследствие фоновой нагрузки, пропускная способность канала — снижаться из-за перегрузки сети, а допустимое время выполнения задачи — изменяться в зависимости от требований качества обслуживания. Следовательно, множество допустимых решений задачи управления также становится динамическим.

Таким образом, задача распределения ресурсов в ГРВС должна рассматриваться как задача адаптивной оптимизации в условиях изменяющейся системы ограничений. Это обстоятельство является одной из ключевых причин перехода от статических методов управления к адаптивным, распределённым и интеллектуальным подходам, рассматриваемым в последующих главах монографии.

2.9. Целевая функция управления

Одним из центральных элементов задачи управления гетерогенной распределённой вычислительной системой является формирование целевой функции, определяющей критерии эффективности функционирования системы и направление оптимизации управляющих воздействий [12–15].

Целевая функция отражает требования к производительности, качеству обслуживания, использованию ресурсов и устойчивости вычислительной среды. В зависимости от прикладной области и особенностей функционирования системы критерии оптимизации могут существенно различаться.

В общем случае задача управления заключается в поиске такого распределения задач X , при котором достигается экстремальное значение целевой функции:

$$X^* = \operatorname{argmin}_X F(X),$$

где $F(X)$ — критерий эффективности распределения ресурсов.

2.9.1. Минимизация времени выполнения

Одной из наиболее распространённых постановок является минимизация времени выполнения вычислительных задач. Данный

критерий особенно важен для высокопроизводительных вычислений, облачных сервисов и систем реального времени.

В такой постановке целевая функция может быть представлена следующим образом:

$$F(X) = \max_i t_i(X)$$

где $t_i(X)$ характеризует время выполнения задачи t_i при заданном распределении ресурсов.

Минимизация максимального времени выполнения позволяет уменьшить общее время завершения набора задач и повысить производительность системы в целом. В ряде случаев вместо максимального времени выполнения могут использоваться среднее время обработки, медианное время отклика или взвешенные показатели качества обслуживания.

2.9.2. Минимизация загрузки ресурсов

Другим важным критерием является минимизация неравномерности загрузки вычислительных узлов. В распределённых системах перегрузка отдельных компонентов приводит к ухудшению производительности и снижению устойчивости функционирования.

Соответствующая целевая функция имеет вид:

$$F(X) = \max_j L_j$$

где L_j характеризует уровень загрузки вычислительного узла n_j .

Минимизация максимальной загрузки способствует более равномерному распределению вычислительной активности и снижает вероятность возникновения локальных перегрузок. Подобные критерии широко применяются в задачах балансировки нагрузки и управления облачными инфраструктурами [9, 16].

2.9.3. Многокритериальная постановка задачи управления

На практике управление распределённой вычислительной системой редко сводится к оптимизации только одного показателя. Как правило, необходимо одновременно учитывать время выполнения задач, степень загрузки ресурсов, стоимость вычислений, энергопотребление, надёжность и другие характеристики функционирования системы.

В этом случае задача управления приобретает многокритериальный характер. Обобщённая целевая функция может быть представлена в виде линейной комбинации частных критериев:

$$F(X) = \alpha_1 F_{time} + \alpha_2 F_{load} + \alpha_3 F_{cost}$$

где F_{time} характеризует временные показатели системы, F_{load} — степень загрузки ресурсов, F_{cost} — стоимость использования вычислительной инфраструктуры, а коэффициенты α_i определяют относительную важность соответствующих критериев.

В более сложных моделях целевая функция может включать дополнительные компоненты, связанные с энергопотреблением, устойчивостью, качеством обслуживания, надёжностью и безопасностью функционирования системы [14, 17]. При этом коэффициенты α_i могут

изменяться динамически в зависимости от состояния системы и текущих требований к её функционированию.

Следует отметить, что критерии оптимизации в распределённых вычислительных системах зачастую являются конфликтующими. Например, минимизация времени выполнения может приводить к увеличению энергопотребления или стоимости вычислений. В связи с этим задача управления ГРВС обычно рассматривается как задача поиска компромиссного решения в условиях множества ограничений и противоречивых критериев эффективности.

Таким образом, целевая функция управления представляет собой формализованное описание целей функционирования распределённой вычислительной системы и служит основой для построения адаптивных алгоритмов распределения ресурсов, рассматриваемых в последующих главах монографии.

2.10. Динамическая модель системы

Одной из ключевых особенностей гетерогенных распределённых вычислительных систем является их динамический характер. В процессе функционирования изменяются вычислительная нагрузка, состав выполняемых задач, состояние сетевой инфраструктуры, доступность ресурсов и параметры внешней среды. Вследствие этого система не может рассматриваться как статическая структура с фиксированными характеристиками [5, 12, 15].

Для описания процессов функционирования ГРВС во времени используется динамическая модель системы:

$$S(t) = \langle N, E, R(t), T(t), D(t), S(t) \rangle,$$

где N и E определяют структуру вычислительных узлов и каналов связи, а множества $R(t)$, $T(t)$, $D(t)$ и состояние системы $S(t)$ рассматриваются как изменяющиеся во времени параметры.

Динамика вычислительной системы обусловлена несколькими взаимосвязанными факторами.

Прежде всего, изменяется вычислительная нагрузка. Интенсивность поступления задач, объём обрабатываемых данных и характер вычислительных процессов в распределённых системах являются нестационарными величинами. В различные моменты времени система может функционировать как в режиме низкой загрузки, так и в условиях перегрузки отдельных компонентов инфраструктуры.

Изменяется также состав задач:

$$T = T(t).$$

В процессе функционирования системы новые задачи поступают в очередь обработки, ранее назначенные задачи завершаются, а некоторые вычислительные процессы могут изменять своё состояние или

перераспределяться между узлами. В результате структура вычислительной нагрузки непрерывно эволюционирует.

Существенное влияние на состояние системы оказывает динамика потоков данных:

$$D = D(t).$$

Интенсивность межузлового обмена информацией зависит от текущего распределения задач, характера вычислительных процессов и состояния коммуникационной инфраструктуры. При изменении маршрутов передачи данных или перераспределении вычислений изменяется нагрузка на сетевые каналы и параметры взаимодействия между компонентами системы.

Динамический характер имеют и вычислительные ресурсы:

$$R = R(t).$$

Фактическая доступность ресурсов изменяется вследствие текущей загрузки узлов, отказов оборудования, ограничений энергопотребления, миграции сервисов и внешних воздействий. В гетерогенных распределённых средах доступность ресурсов может существенно различаться для различных узлов системы и изменяться во времени.

Особую роль играет изменение состояния каналов связи. Пропускная способность, задержки передачи данных и уровень потерь зависят от сетевой нагрузки, маршрутизации, качества соединений и внешних факторов. Вследствие этого коммуникационная подсистема также должна рассматриваться как динамическая среда [10, 16].

С учётом указанных факторов задача управления распределённой вычислительной системой приобретает динамический характер. Управляющие воздействия должны формироваться не только на основе текущего состояния системы, но и с учётом прогнозируемых изменений вычислительной нагрузки и параметров среды. В результате управление ГРВС становится задачей адаптивного управления динамической системой в условиях неопределённости.

Переход к динамической модели имеет принципиальное значение для последующего рассмотрения интеллектуальных методов управления. В отличие от статических подходов, динамическая модель позволяет учитывать эволюцию состояния системы, реализовывать механизмы самонастройки и использовать прогнозирование для формирования адаптивных стратегий распределения ресурсов [14, 17].

Таким образом, гетерогенная распределённая вычислительная система должна рассматриваться как сложная динамическая среда, параметры которой непрерывно изменяются во времени. Именно это обстоятельство обуславливает необходимость применения адаптивных, распределённых и интеллектуальных методов управления, рассматриваемых в последующих главах монографии.

2.11. Обобщение формальной модели системы

Построенная формальная модель гетерогенной распределённой вычислительной системы позволяет перейти от описательного представления распределённой вычислительной среды к строгому математическому описанию её структуры, ресурсов, вычислительной нагрузки и процессов взаимодействия между компонентами системы [12–15].

В результате проведённой формализации установлено, что гетерогенная распределённая вычислительная система представляет собой сложную динамическую систему, состояние которой непрерывно изменяется под воздействием вычислительной нагрузки, сетевого взаимодействия, изменения доступности ресурсов и внешних факторов. Динамический характер системы проявляется как на уровне вычислительных узлов и потоков данных, так и на уровне структуры ограничений и параметров среды функционирования.

Показано, что задача управления распределённой вычислительной системой является многокритериальной задачей оптимизации. При формировании управляющих воздействий необходимо одновременно учитывать время выполнения задач, загрузку вычислительных ресурсов, сетевые ограничения, стоимость вычислений, энергопотребление, требования качества обслуживания и показатели устойчивости функционирования системы [8, 13, 16].

Существенным результатом формализации является установление зависимости ограничений не только от распределения ресурсов, но и от текущего состояния системы:

$$g_i = g_i(X, S(t)).$$

Данное обстоятельство отражает фундаментальную особенность современных распределённых вычислительных сред: допустимость и эффективность управляющих решений определяются изменяющимся состоянием вычислительной и коммуникационной инфраструктуры. Вследствие этого множество допустимых решений задачи управления также становится динамическим.

Полученные результаты позволяют сделать принципиально важный вывод: оптимальное решение задачи управления не может рассматриваться как фиксированное или заранее заданное. Решения, являющиеся эффективными в одном состоянии системы, могут становиться неэффективными или недопустимыми при изменении вычислительной нагрузки, структуры потоков данных или параметров сетевой среды.

Следовательно, управление гетерогенными распределёнными вычислительными системами должно носить адаптивный характер и обеспечивать возможность непрерывной корректировки управляющих воздействий в зависимости от изменения состояния системы во времени [3, 5, 15]. Это требует перехода от статических методов распределения ресурсов к интеллектуальным механизмам адаптивного управления,

способным учитывать динамику вычислительной среды, неопределённость наблюдаемой информации и изменяющуюся структуру ограничений.

Именно данные вопросы рассматриваются в следующей главе, посвящённой принципам и методам адаптивного управления гетерогенными распределёнными вычислительными системами.

Выводы по главе 2

В главе разработана формальная модель гетерогенной распределённой вычислительной системы, позволяющая описывать структуру вычислительной среды, процессы распределения ресурсов и взаимодействие компонентов системы в условиях динамически изменяющейся нагрузки [12–15]. В качестве основы моделирования использованы теоретико-множественное и графовое представления, обеспечивающие возможность дальнейшей формализации задач управления и оптимизации.

Предложена обобщённая модель ГРВС в виде совокупности вычислительных узлов, коммуникационных каналов, ресурсов, задач, потоков данных и состояния системы во времени. Выполнена формализация вычислительных узлов с учётом гетерогенности ресурсов, включающей различия вычислительных, коммуникационных и программных характеристик компонентов системы.

Построена графовая модель распределённой вычислительной среды, позволяющая учитывать структуру сетевого взаимодействия, параметры каналов связи и динамику потоков данных. Введены формальные модели вычислительных задач и потоков данных, описывающие вычислительную сложность задач, требования к памяти, объёмы передаваемой информации и временные ограничения [8, 13].

Выполнена формализация задачи распределения ресурсов в виде задачи оптимизации с системой ограничений. Показано, что ограничения в гетерогенных распределённых вычислительных системах носят многомерный характер и охватывают вычислительные, сетевые, временные, энергетические и организационные аспекты функционирования системы.

Принципиально важным результатом является установление зависимости ограничений от текущего состояния системы:

$$g_i = g_i(X, S(t)).$$

Данная зависимость отражает динамический характер распределённой вычислительной среды и показывает, что множество допустимых решений изменяется во времени под воздействием вычислительной нагрузки, состояния каналов связи, доступности ресурсов и внешних факторов.

Показано, что задача управления ГРВС является многокритериальной задачей адаптивной оптимизации, в которой оптимальное распределение ресурсов не может рассматриваться как

статическое решение. Эффективное управление требует непрерывного анализа состояния системы, корректировки управляющих воздействий и учёта изменяющихся ограничений и критериев функционирования [5, 15, 17].

Таким образом, проведённая формализация создаёт теоретическую основу для последующего рассмотрения методов адаптивного управления, интеллектуальной координации ресурсов и мультиагентных подходов, рассматриваемых в следующих главах монографии.

ГЛАВА 3. ПРИНЦИПЫ И МЕТОДЫ АДАПТИВНОГО УПРАВЛЕНИЯ ГЕТЕРОГЕННЫМИ РАСПРЕДЕЛЁННЫМИ ВЫЧИСЛИТЕЛЬНЫМИ СИСТЕМАМИ

3.1. Понятие адаптивного управления

Рассмотренная в главе 2 формальная модель гетерогенной распределённой вычислительной системы показывает, что современные распределённые вычислительные среды характеризуются высокой динамичностью, неопределённостью параметров и изменяющейся структурой ограничений [12–15]. Состояние системы непрерывно изменяется под воздействием вычислительной нагрузки, сетевой активности, доступности ресурсов и внешних факторов, а эффективность управляющих решений зависит от текущего режима функционирования вычислительной инфраструктуры.

Для ГРВС характерны:

- динамическое изменение состояния системы $S(t)$;
- неопределённость параметров среды и вычислительной нагрузки;
- зависимость ограничений от текущего состояния системы;
- многокритериальный характер задачи управления;
- изменяющаяся структура вычислительных и коммуникационных ресурсов.

В этих условиях классические методы управления, основанные на фиксированных алгоритмах и заранее заданных правилах принятия решений, оказываются недостаточно эффективными [5, 8, 11]. Статические стратегии распределения ресурсов не способны адекватно реагировать на изменение вычислительной нагрузки, перегрузку отдельных компонентов системы и изменение характеристик сетевой инфраструктуры.

Данное обстоятельство обуславливает необходимость перехода к адаптивным методам управления, ориентированным на непрерывный анализ состояния системы и динамическую корректировку управляющих воздействий.

Под адаптивным управлением гетерогенной распределённой вычислительной системой далее будем понимать процесс формирования и корректировки управляющих воздействий на основе анализа текущего состояния системы и её динамики с целью достижения заданных характеристик функционирования.

В отличие от статических методов, адаптивное управление предполагает возможность изменения параметров и структуры управляющего алгоритма в процессе функционирования системы. Управляющая подсистема должна не только реагировать на изменения состояния среды, но и модифицировать собственную стратегию

управления с учётом накопленной информации и наблюдаемой динамики процессов [14, 16].

Формально адаптивное управление может быть представлено следующим отображением:

$$X(t) = A(\hat{S}(t), \theta(t)).$$

где $X(t)$ — управляющее воздействие, $S(t)$ — текущее состояние системы, $\theta(t)$ — параметры модели управления, а A — адаптивный алгоритм формирования управляющих решений.

В контексте распределённых вычислительных систем управляющее воздействие $X(t)$ может включать:

- перераспределение задач между вычислительными узлами;
- изменение маршрутов передачи данных;
- масштабирование вычислительных ресурсов;
- миграцию контейнеров и сервисов;
- изменение приоритетов задач;
- перестройку структуры взаимодействия компонентов системы.

Ключевой особенностью адаптивного управления является динамическое изменение параметров модели управления:

$$\theta(t + 1) = \Phi(\theta(t), \hat{S}(t), X(t), S(t + 1)).$$

где Φ определяет механизм корректировки параметров управления на основе текущего состояния системы и результатов ранее принятых решений.

Данная зависимость отражает способность системы изменять собственную стратегию управления в процессе функционирования. В отличие от классических подходов, параметры управления здесь не являются фиксированными, а эволюционируют во времени под воздействием наблюдаемой динамики среды.

Следует отметить, что в распределённых вычислительных системах адаптивность может реализовываться на различных уровнях:

- на уровне распределения вычислительных ресурсов;
- на уровне сетевого взаимодействия;
- на уровне балансировки нагрузки;
- на уровне управления потоками данных;
- на уровне координации распределённых компонентов и агентов.

В современных интеллектуальных вычислительных средах адаптивное управление всё чаще сочетается с методами машинного обучения, мультиагентной координации и интеллектуального прогнозирования [15–17]. Это позволяет системе не только реагировать на уже произошедшие изменения, но и формировать проактивные стратегии управления на основе анализа тенденций изменения состояния вычислительной среды.

Таким образом, адаптивное управление представляет собой необходимую основу построения современных систем управления гетерогенными распределёнными вычислительными инфраструктурами, функционирующими в условиях высокой динамики, неопределённости и изменяющейся структуры ограничений.

3.2. Контур адаптивного управления

Адаптивное управление гетерогенной распределённой вычислительной системой реализуется в виде замкнутого контура, обеспечивающего непрерывное наблюдение за состоянием системы, анализ отклонений, формирование управляющих решений и оценку последствий выполненных воздействий [14–17]. Такой контур соответствует общей логике управления сложными динамическими системами и позволяет учитывать изменения вычислительной нагрузки, доступности ресурсов и состояния коммуникационной инфраструктуры.

В общем виде контур адаптивного управления может быть представлен следующей последовательностью:

$$S(t) \rightarrow Y(t) \rightarrow \hat{S}(t) \rightarrow X(t) \rightarrow S(t+1).$$

Рисунок 3 иллюстрирует замкнутый контур адаптивного управления, связывающий процессы мониторинга, анализа, планирования, исполнения и обратной связи.

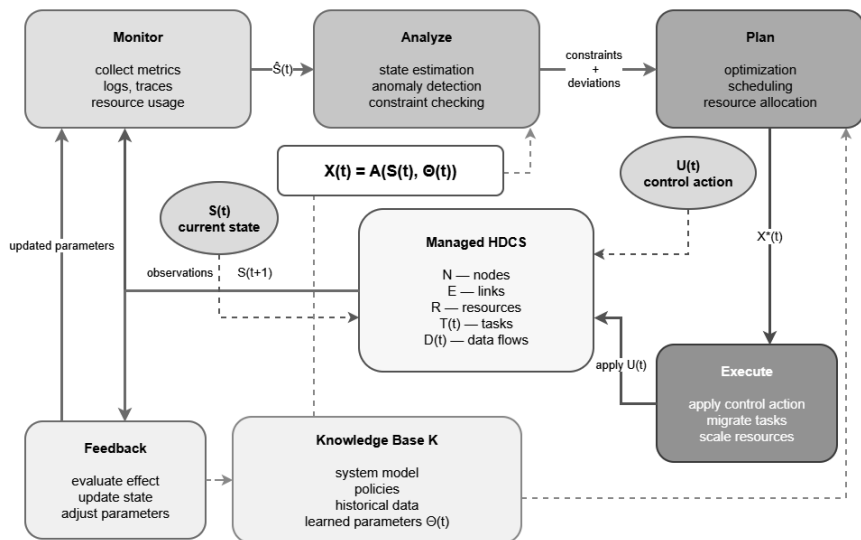
Первым этапом является мониторинг (Monitor), обеспечивающий сбор данных о текущем состоянии вычислительной системы. На этом этапе формируются наблюдаемые параметры, характеризующие загрузку узлов, использование ресурсов, состояние очередей задач, параметры сетевых каналов и показатели качества обслуживания. Формально процесс наблюдения может быть представлен в виде:

$$\hat{S}(t) = O(Y(t)),$$

где $Y(t)$ — множество наблюдаемых данных, включающее сведения о N, L, R, T и потоках данных.

Следует учитывать, что в реальных распределённых системах наблюдаемое состояние не всегда совпадает с истинным состоянием системы. Данные мониторинга могут поступать с задержкой, содержать шум или отражать только часть параметров вычислительной среды. Поэтому на практике управляющая подсистема оперирует оценкой состояния $\hat{S}(t)$, полученной на основе метрик, логов, трассировок и событий [5, 12].

Adaptive Control Loop for Heterogeneous Distributed Computing Systems



Closed adaptive control loop: monitoring, analysis, planning, execution, feedback, and knowledge update

Рисунок 3 – Контур адаптивного управления

Вторым этапом является анализ (Analyze). Его задача состоит в интерпретации наблюдаемого состояния, выявлении отклонений от целевых параметров и определении причин ухудшения качества функционирования системы. В простейшем виде отклонение состояния может быть задано следующим образом:

$$\Delta S = S(t) - S_{target},$$

где S_{target} — целевое или нормативное состояние системы.

На этапе анализа могут выявляться перегруженные узлы, недостаточно используемые ресурсы, нарушения временных ограничений, рост сетевых задержек, аномальные состояния и потенциальные отказные ситуации. Результаты анализа формируют основу для последующего планирования управляющих воздействий.

Третьим этапом является планирование (Plan), на котором осуществляется выбор управляющего решения. В контексте ГРВС планирование обычно сводится к решению задачи распределения ресурсов, балансировки нагрузки или изменения структуры выполнения задач. Формально задача планирования может быть представлена как задача оптимизации:

$$X^*(t) = \arg \min_{\hat{X}} F(X, \hat{S}(t))$$

при ограничениях $\hat{g}_i(X, \hat{S}(t)) \leq b_i$.

Здесь X^* — оптимальное или допустимое распределение задач и ресурсов, $F(X, \hat{S}(t))$ — целевая функция управления, а $\hat{g}_i(X, \hat{S}(t)) \leq b_i$ — система ограничений, зависящих от текущего состояния системы. Именно зависимость ограничений от $\hat{S}(t)$ отличает адаптивное управление от статических подходов к распределению ресурсов.

Четвёртым этапом является исполнение (Execute), в ходе которого сформированное управляющее воздействие применяется к системе. Управляющее воздействие может включать миграцию задач, изменение маршрутов передачи данных, перераспределение вычислительных ресурсов, масштабирование сервисов или изменение приоритетов обработки. Динамика системы после применения управляющего воздействия может быть представлена следующим образом:

$$S(t+1) = f(S(t), X^*(t)),$$

где $X^*(t)$ — управляющее воздействие, сформированное на этапе планирования, а f — функция перехода системы из текущего состояния в последующее.

Замкнутость контура управления обеспечивается обратной связью: после выполнения управляющего воздействия система вновь переходит к этапу мониторинга, и цикл повторяется. Это позволяет не только реагировать на изменения состояния системы, но и оценивать последствия ранее принятых решений, корректировать параметры управления и повышать устойчивость функционирования ГРВС.

Таким образом, контур адаптивного управления является базовой организационной схемой, обеспечивающей переход от статического распределения ресурсов к динамическому управлению, основанному на наблюдении, анализе, планировании и непрерывной корректировке управляющих воздействий [14–17].

3.3. Классификация методов адаптации

Методы адаптации в гетерогенных распределённых вычислительных системах различаются по источнику информации, характеру реакции на изменение состояния системы и способу корректировки управляющих воздействий. В общем случае можно выделить три основных класса методов: реактивные, проактивные и обучающиеся. Данная классификация отражает эволюцию подходов от простого реагирования на изменения к прогнозированию состояния системы и последующему самообучению управляющих механизмов [14–17].

3.3.1. Реактивные методы

Реактивные методы адаптации основаны на непосредственной реакции системы управления на уже зафиксированные изменения состояния вычислительной среды. Управляющее воздействие в этом случае формируется на основе текущего наблюдаемого состояния:

$$X(t) = f(S^{\wedge}(t)),$$

где $S^{\wedge}(t)$ — текущее состояние системы, а f — функция формирования управляющего воздействия.

К реактивным методам относятся динамическое перераспределение нагрузки, миграция задач при перегрузке узлов, масштабирование ресурсов при превышении пороговых значений, изменение приоритетов очередей и переключение маршрутов передачи данных при ухудшении характеристик каналов связи.

Основным достоинством реактивных методов является относительная простота реализации. Такие методы не требуют сложных прогнозных моделей и могут эффективно применяться в системах, где изменения состояния легко обнаруживаются с помощью мониторинга. Однако их существенным ограничением является запаздывающий характер управления: управляющее воздействие формируется только после возникновения отклонения. В условиях быстро меняющейся нагрузки это может приводить к росту времени отклика, перегрузке отдельных компонентов и временным нарушениям требований качества обслуживания [5, 12].

3.3.2. Проактивные методы

Проактивные методы адаптации основаны на прогнозировании будущего состояния системы и формировании управляющих воздействий до наступления критических изменений. В этом случае используется оценка будущего состояния:

$$S^{\wedge}(t + 1) = P(S(t)),$$

где P — модель прогнозирования состояния системы.

Управляющее воздействие формируется с учётом прогнозируемого состояния:

$$X(t) = f(S^{\wedge}(t + 1)).$$

Проактивные методы позволяют учитывать предполагаемое изменение вычислительной нагрузки, рост очередей задач, возможную перегрузку сетевых каналов и снижение доступности ресурсов. К числу таких методов относятся предиктивное масштабирование, прогнозное перераспределение задач, предупреждающая миграция сервисов и адаптивное резервирование ресурсов.

Преимущество проактивных методов заключается в возможности предотвращать перегрузки и снижать вероятность нарушения временных ограничений. Вместе с тем эффективность таких методов существенно зависит от качества прогнозной модели, полноты наблюдаемых данных и

устойчивости статистических закономерностей в поведении системы [15–17]. При неточном прогнозе проактивное управление может приводить к избыточному выделению ресурсов или ошибочному перераспределению нагрузки.

3.3.3. Обучающиеся методы

Обучающиеся методы адаптации предполагают изменение параметров управляющей модели на основе накопленного опыта функционирования системы. В отличие от реактивных и проактивных методов, они ориентированы не только на текущее или прогнозируемое состояние среды, но и на постепенное улучшение стратегии управления во времени.

В общем виде изменение параметров модели управления может быть представлено следующим образом:

$$\theta(t + 1) = \theta(t) + \eta \cdot \nabla J,$$

где $\theta(t)$ — параметры модели управления, η — коэффициент обучения, J — функция качества управления, а ∇J — направление изменения параметров, определяемое выбранным методом оптимизации.

Функция качества J может учитывать время выполнения задач, степень загрузки ресурсов, число нарушений SLA, энергопотребление, стоимость вычислений и устойчивость функционирования системы. В зависимости от постановки задачи обучение может быть реализовано с использованием методов машинного обучения, обучения с подкреплением, нейросетевых моделей или гибридных интеллектуальных алгоритмов [14, 16, 17].

Преимущество обучающихся методов заключается в способности адаптировать стратегию управления к повторяющимся режимам функционирования системы, выявлять скрытые закономерности в данных и повышать качество решений по мере накопления опыта. Однако применение таких методов требует достаточного объема данных, контроля устойчивости обучения и защиты от переобучения на ограниченном наборе сценариев.

Таким образом, реактивные, проактивные и обучающиеся методы адаптации образуют взаимодополняющие классы подходов к управлению ГРВС. В практических системах наиболее эффективными оказываются гибридные схемы, сочетающие быструю реакцию на текущие изменения, прогнозирование будущих состояний и обучение управляющих моделей на основе накопленных данных.

3.4. Методы принятия решений в условиях неопределённости

Одной из ключевых особенностей управления гетерогенными распределёнными вычислительными системами является необходимость принятия решений в условиях неопределённости. В реальных

распределённых средах управляющая подсистема, как правило, не располагает полной и абсолютно достоверной информацией о состоянии вычислительной инфраструктуры, сетевых потоков и вычислительной нагрузки [5, 11, 15].

Неопределённость в ГРВС проявляется в нескольких формах:

- неполнота наблюдаемых данных;
- шум и ошибки измерений;
- задержки мониторинга;
- стохастический характер поступления задач;
- динамическое изменение параметров среды;
- непредсказуемое поведение отдельных компонентов системы.

Вследствие этого задача управления не может рассматриваться как задача детерминированной оптимизации с полностью известными параметрами. Для принятия управляющих решений необходимо использовать методы, способные функционировать в условиях неполной и изменяющейся информации [12–14].

3.4.1. Эвристические методы

Эвристические методы являются одним из наиболее распространённых классов подходов к управлению распределёнными вычислительными системами. В таких методах управляющее воздействие формируется на основе набора правил, эвристик или локальных критериев:

$$X(t) = h(S^{\wedge}(t)),$$

где h — эвристическая функция принятия решений, зависящая от текущего состояния системы.

Эвристические методы ориентированы преимущественно на получение допустимого решения за ограниченное время и особенно эффективны в задачах с высокой вычислительной сложностью, где поиск глобального оптимума является практически невозможным.

К числу наиболее распространённых эвристических подходов относятся:

- greedy-алгоритмы распределения задач;
- алгоритмы минимальной загрузки;
- правила приоритетного обслуживания;
- эвристики маршрутизации;
- локальные методы балансировки нагрузки.

Основным преимуществом эвристических методов является их высокая скорость работы и возможность применения в режиме реального времени. Однако подобные подходы обычно не гарантируют достижение глобально оптимального решения и могут демонстрировать нестабильное качество управления при изменении характеристик среды [8, 13].

3.4.2. Оптимизационные методы

Оптимизационные методы основаны на формальном поиске решения, обеспечивающего экстремальное значение целевой функции при

выполнении системы ограничений. В общем виде задача управления может быть представлена следующим образом:

$$X^* = \operatorname{argmin}_X F(X)$$

при ограничениях

$$g_i(X, S^\wedge(t)) \leq b_i.$$

В отличие от эвристических подходов, оптимизационные методы ориентированы на достижение формально обоснованного решения с учётом множества критериев и ограничений. В распределённых вычислительных системах подобные методы используются для задач распределения ресурсов, балансировки нагрузки, маршрутизации потоков данных и управления энергопотреблением [12, 15].

В исследованиях автора и представителей научной школы рассматривались различные классы оптимизационных методов, включая:

- генетические алгоритмы;
- многокритериальные методы оптимизации;
- методы адаптивного распределения ресурсов;
- эволюционные подходы к управлению вычислительной нагрузкой;
- методы интеллектуального поиска допустимых конфигураций системы [1–3].

Особый интерес представляют многокритериальные постановки задачи управления, в которых одновременно учитываются временные ограничения, загрузка ресурсов, стоимость вычислений, надёжность и качество обслуживания. В подобных задачах управление сводится к поиску компромиссного решения в пространстве противоречивых критериев.

Недостатком оптимизационных методов является высокая вычислительная сложность, особенно в условиях крупномасштабных распределённых систем с динамически изменяющимся состоянием среды. Это ограничивает возможность их применения в задачах оперативного управления без использования приближённых или гибридных алгоритмов.

3.4.3. Вероятностные методы

Вероятностные методы ориентированы на описание динамики распределённой вычислительной системы в терминах вероятностных процессов и случайных состояний. В таких моделях переход системы между состояниями рассматривается как стохастический процесс:

$$P(S^\wedge(t+1) | S^\wedge(t), X(t)),$$

где вероятность перехода в состояние $S^\wedge(t+1)$ зависит от текущего состояния системы $S^\wedge(t)$ и выбранного управляющего воздействия $X(t)$.

Использование вероятностных моделей позволяет учитывать случайный характер поступления задач, изменение нагрузки, отказы оборудования и неопределённость сетевых параметров. К числу наиболее распространённых вероятностных подходов относятся:

- марковские модели;
- байесовские методы;
- стохастические процессы;
- вероятностные модели очередей;
- методы статистического прогнозирования.

Вероятностные методы особенно эффективны в задачах прогнозирования нагрузки, оценки надёжности, анализа рисков и построения систем проактивного управления [10, 16, 17]. Они позволяют формировать управляющие воздействия с учётом вероятности возникновения перегрузок, отказов и иных неблагоприятных состояний системы.

Вместе с тем применение вероятностных моделей требует наличия достаточного объёма статистических данных и построения адекватных моделей распределения вероятностей. В условиях высокودинамичных вычислительных сред это может представлять значительную сложность.

Таким образом, принятие решений в условиях неопределённости является одной из фундаментальных задач управления гетерогенными распределёнными вычислительными системами. Эвристические, оптимизационные и вероятностные методы образуют основу современных подходов к интеллектуальному управлению распределённой вычислительной инфраструктурой. На практике наибольшую эффективность демонстрируют гибридные подходы, сочетающие преимущества различных классов методов в зависимости от текущего состояния системы и характера решаемой задачи [14–17].

3.5. Предиктивное управление

Предиктивное управление является развитием адаптивного подхода и основано на использовании прогноза будущего состояния системы при формировании управляющих воздействий. В отличие от реактивных методов, ориентированных на устранение уже возникших отклонений, предиктивное управление стремится заранее оценить возможную динамику вычислительной нагрузки, состояния ресурсов и сетевой инфраструктуры [14–17].

В гетерогенных распределённых вычислительных системах необходимость предиктивного управления обусловлена тем, что многие управляющие воздействия имеют ненулевое время реализации. Например, миграция задач, масштабирование сервисов, перераспределение потоков данных или запуск дополнительных вычислительных ресурсов требуют времени. Если управляющее воздействие формируется только после возникновения перегрузки, система может не успеть предотвратить нарушение ограничений качества обслуживания.

Общая модель прогноза состояния системы может быть представлена следующим образом:

$$S^{\wedge}(t+k) = M(S(t), S(t-1), \dots, S^{\wedge}(t-k)),$$

где $S^{\wedge}(t+k)$ — прогнозируемое состояние системы через k шагов, M — модель прогнозирования, использующая текущие и предшествующие состояния системы.

В качестве модели M могут применяться статистические методы, регрессионные модели, нейросетевые архитектуры, модели временных рядов, методы машинного обучения и гибридные подходы. Выбор модели зависит от характера нагрузки, доступности исторических данных, требований к точности прогноза и допустимых вычислительных затрат [15, 16].

Прогнозируемое состояние системы может включать ожидаемую загрузку узлов, длину очередей задач, состояние сетевых каналов, вероятность перегрузки, уровень энергопотребления и вероятность нарушения временных ограничений. На основе этих данных управляющая подсистема получает возможность формировать решения не только с учётом текущего состояния, но и с учётом ожидаемой динамики системы.

Задача оптимизации с прогнозом может быть записана в следующем виде:

$$X^* = \operatorname{argmin}_X \sum_{k=1}^H F(X, S^{\wedge}(t+k)),$$

где H — горизонт планирования, определяющий число будущих шагов, учитываемых при выборе управляющего решения.

Горизонт планирования играет важную роль в предиктивном управлении. При малом значении H система быстро реагирует на ближайшие изменения, но может игнорировать долгосрочные последствия управляющих воздействий. При большом значении H управление становится более стратегическим, однако возрастает вычислительная сложность и снижается достоверность прогноза.

В практических задачах предиктивное управление может использоваться для:

- заблаговременного перераспределения задач при ожидаемом росте нагрузки;

- предиктивного масштабирования вычислительных ресурсов;

- предупреждения перегрузки сетевых каналов;

- оптимизации энергопотребления;

- снижения вероятности нарушения SLA;

- подготовки резервных ресурсов при прогнозируемых отказах.

Особое значение предиктивное управление приобретает в сочетании с мультиагентными и интеллектуальными подходами. В этом случае отдельные агенты могут формировать локальные прогнозы состояния своих областей ответственности, а координирующий уровень — объединять их в глобальную стратегию управления. Такой подход

позволяет учитывать как локальную динамику отдельных узлов, так и общесистемные тенденции изменения нагрузки [3, 14, 17].

Следует учитывать, что эффективность предиктивного управления определяется качеством прогнозной модели и полнотой наблюдаемых данных. Ошибки прогноза могут приводить к неэффективному распределению ресурсов, преждевременному масштабированию или ошибочным управляющим воздействиям. Поэтому предиктивные методы должны сочетаться с механизмами обратной связи и корректировки решений по мере поступления новой информации.

Таким образом, предиктивное управление обеспечивает переход от реакции на уже возникшие изменения к предупреждающему управлению состоянием гетерогенной распределённой вычислительной системы. Оно является важным элементом адаптивного управления и создаёт основу для применения интеллектуальных методов прогнозирования, рассматриваемых в последующих главах монографии.

3.6. Самоорганизация и самонастройка

Развитие адаптивного управления в гетерогенных распределённых вычислительных системах связано не только с возможностью корректировки отдельных управляющих воздействий, но и со способностью системы изменять собственную структуру, параметры и режимы функционирования. В этом контексте важное значение приобретают свойства самоорганизации и самонастройки, характерные для класса самоуправляемых и автономных вычислительных систем [14–17].

Под самоорганизацией будем понимать способность распределённой вычислительной системы изменять структуру взаимодействия компонентов, распределение функций и конфигурацию ресурсов без полного централизованного предписания. Самонастройка, в свою очередь, связана с автоматическим изменением параметров управляющих алгоритмов и режимов функционирования системы на основе анализа текущего состояния и результатов предыдущих управляющих воздействий.

В литературе по автономным вычислительным системам обычно выделяют четыре базовых свойства самоуправления: self-configuration, self-optimization, self-healing и self-protection [14, 16]. В контексте ГРВС они приобретают следующий смысл.

Self-configuration означает способность системы автоматически изменять собственную конфигурацию при добавлении, удалении или изменении доступности вычислительных узлов, сервисов и сетевых соединений. Данное свойство особенно важно для облачных и периферийных вычислительных сред, где состав ресурсов может изменяться во времени.

Self-optimization характеризует способность системы улучшать показатели функционирования за счёт динамической настройки параметров управления, перераспределения задач, балансировки нагрузок и выбора более эффективных маршрутов передачи данных. В этом случае система стремится поддерживать заданные критерии качества без постоянного вмешательства оператора.

Self-healing отражает способность системы обнаруживать отказы, деградацию производительности и аномальные состояния, а затем автоматически выполнять восстановительные действия. К таким действиям могут относиться перезапуск сервисов, миграция задач, переключение на резервные ресурсы и перестройка маршрутов взаимодействия.

Self-protection связана со способностью системы выявлять потенциально опасные состояния, предотвращать нарушения политики безопасности, ограничивать некорректные управляющие воздействия и обеспечивать устойчивость к внешним и внутренним угрозам. Для распределённых вычислительных систем это свойство включает не только классические механизмы безопасности, но и контроль целостности управляющего контура.

Формально процесс самоорганизации и самонастройки может быть представлен как преобразование состояния и структуры системы под воздействием механизма адаптации:

$$S^{\wedge}(t+1) = \Psi(S^{\wedge}(t), X(t), K(t)),$$

где $S^{\wedge}(t)$ — текущее состояние системы, $X(t)$ — управляющее воздействие, $K(t)$ — накопленные знания о функционировании системы, а Ψ — функция адаптивного изменения состояния, параметров или структуры системы.

В отличие от исходной записи, учитывающей только текущее состояние, данная форма подчёркивает, что самоорганизация зависит не только от состояния системы, но и от управляющих решений, истории функционирования, накопленных знаний и действующих политик управления.

В мультиагентных системах самоорганизация реализуется через локальные решения агентов и их координацию. Отдельные агенты могут изменять свои роли, перераспределять зоны ответственности, инициировать восстановительные действия или согласовывать новые стратегии взаимодействия. Это позволяет системе сохранять работоспособность и эффективность даже при изменении состава ресурсов и частичных отказах компонентов [3, 15, 17].

Самонастройка параметров управления особенно важна в условиях нестационарной нагрузки. Например, пороговые значения для масштабирования, веса критериев в целевой функции, параметры прогнозных моделей и политики балансировки могут изменяться в

зависимости от текущего режима работы системы. В результате управление становится не только адаптивным, но и эволюционирующим во времени.

Таким образом, самоорганизация и самонастройка являются важными свойствами адаптивных систем управления ГРВС. Они обеспечивают способность системы сохранять эффективность, устойчивость и управляемость в условиях изменения нагрузки, отказов компонентов, неполноты информации и динамической трансформации вычислительной инфраструктуры.

3.7. Интеграция адаптивного управления с формальной моделью

Построенная в главе 2 формальная модель гетерогенной распределённой вычислительной системы создаёт основу для строгого описания процессов адаптивного управления. Интеграция модели состояния системы, распределения ресурсов, потоков данных и системы ограничений позволяет представить задачу управления ГРВС как задачу динамической адаптивной оптимизации [12–15].

Согласно введённой ранее формализации, состояние распределённой вычислительной системы определяется вектором:

$$S^{\wedge}(t) = (L(t), U(t), Q(t), C(t)),$$

где $L(t)$ характеризует нагрузку вычислительных узлов, $U(t)$ — использование ресурсов, $Q(t)$ — состояние очередей задач, а $C(t)$ — состояние коммуникационной инфраструктуры.

Данный вектор отражает текущее состояние вычислительной среды и выступает основным источником информации для формирования управляющих воздействий. В отличие от статических моделей, адаптивное управление предполагает непрерывное изменение решений в зависимости от динамики состояния системы.

Управляющее воздействие в этом случае формируется как функция текущего состояния:

$$X(t) = A(S^{\wedge}(t)),$$

где A — адаптивный алгоритм управления, определяющий распределение задач, ресурсов и потоков данных на основе наблюдаемого состояния системы.

Следует отметить, что распределение ресурсов $X(t)$ в рассматриваемой постановке является динамической переменной, изменяющейся во времени под воздействием вычислительной нагрузки, сетевых ограничений и результатов функционирования управляющего контура.

Система ограничений также зависит от текущего состояния вычислительной среды:

$$g_i(X, S^{\wedge}(t)) \leq b_i.$$

Данная зависимость отражает фундаментальную особенность гетерогенных распределённых вычислительных систем: допустимость и эффективность решений определяются не только самим распределением ресурсов, но и текущими характеристиками вычислительных узлов, сетевой инфраструктуры и информационных потоков.

Например, доступная вычислительная мощность узла, пропускная способность канала связи или допустимое время выполнения задачи могут изменяться во времени вследствие перегрузки, изменения маршрутов передачи данных, отказов оборудования или внешних воздействий. Следовательно, множество допустимых решений также становится динамическим.

Цель адаптивного управления заключается в минимизации целевой функции, учитывающей текущее состояние системы:

$$\min F(X, S^{\wedge}(t)).$$

В качестве компонентов целевой функции могут выступать время выполнения задач, степень загрузки ресурсов, энергопотребление, стоимость вычислений, показатели качества обслуживания и устойчивости функционирования системы [8, 13, 16].

С учётом введённых обозначений итоговая постановка задачи адаптивного управления гетерогенной распределённой вычислительной системой может быть представлена следующим образом:

$$\min F(X, S^{\wedge}(t))$$

при ограничениях $g_i(X, S^{\wedge}(t)) \leq b_i$.

Данная постановка отражает ключевые особенности управления современными распределёнными вычислительными системами:

- динамический характер среды функционирования;
- зависимость ограничений от текущего состояния системы;
- многокритериальность задачи управления;
- необходимость непрерывной корректировки управляющих воздействий;
- невозможность использования фиксированных статических решений.

В отличие от классических задач распределения ресурсов, рассматриваемая постановка ориентирована не на поиск единственного оптимального решения, а на непрерывное формирование адаптивной стратегии управления в условиях изменяющегося состояния вычислительной среды [14–17].

Таким образом, интеграция адаптивного управления с формальной моделью ГРВС создаёт теоретическую основу для дальнейшего рассмотрения распределённых, мультиагентных и интеллектуальных методов управления, обеспечивающих согласованное функционирование компонентов системы в условиях высокой динамики и неопределённости.

3.8. Ключевые свойства адаптивного управления

Адаптивное управление гетерогенными распределёнными вычислительными системами обладает рядом свойств, отличающих его от классических статических и жёстко детерминированных методов управления. Эти свойства определяют способность системы сохранять эффективность функционирования в условиях изменяющейся нагрузки, неполной информации и динамической структуры ограничений [14–17].

Первым ключевым свойством является контекстность управления. Управляющее решение формируется не изолированно, а с учётом текущего состояния системы $S(t)$, доступности ресурсов, загрузки узлов, состояния сетевых каналов, состава задач и действующих ограничений. Это означает, что одно и то же распределение ресурсов может быть эффективным в одном состоянии системы и неэффективным в другом. Следовательно, адаптивное управление всегда связано с анализом конкретного контекста функционирования вычислительной среды.

Вторым свойством является динамичность. В адаптивной системе управляющие воздействия изменяются во времени в зависимости от изменения состояния среды. В отличие от статических алгоритмов, параметры и решения адаптивного управления не фиксируются заранее, а корректируются в процессе функционирования системы. Динамичность управления проявляется в перераспределении задач, изменении маршрутов передачи данных, масштабировании ресурсов, изменении приоритетов обработки и перестройке взаимодействия между компонентами системы.

Третьим свойством является наличие обратной связи. Адаптивное управление предполагает непрерывную оценку последствий принятых решений и использование результатов этой оценки для последующей корректировки управляющих воздействий. Обратная связь позволяет выявлять отклонения от целевых параметров, оценивать эффективность выполненных действий, корректировать параметры моделей управления и повышать устойчивость системы к изменениям среды [5, 12].

Четвёртым свойством является интеллектуальность. В современных ГРВС адаптивное управление всё чаще опирается на методы прогнозирования, машинного обучения, оптимизации и интеллектуального анализа данных. Интеллектуальность управления проявляется в способности системы выявлять закономерности в изменении нагрузки, прогнозировать будущие состояния, выбирать рациональные стратегии распределения ресурсов и адаптировать параметры управления на основе накопленного опыта [15–17].

Перечисленные свойства тесно взаимосвязаны. Контекстность обеспечивает учёт текущего состояния системы, динамичность — возможность изменения решений во времени, обратная связь — механизм оценки результатов управления, а интеллектуальность — средства анализа,

прогноза и выбора эффективных управляющих воздействий. В совокупности они формируют основу адаптивного управления гетерогенными распределёнными вычислительными системами и создают предпосылки для перехода к мультиагентным и интеллектуальным архитектурам управления, рассматриваемым в последующих главах.

3.9. Ограничения адаптивного управления

Несмотря на существенные преимущества адаптивного управления, его применение в гетерогенных распределённых вычислительных системах связано с рядом ограничений, которые необходимо учитывать при проектировании архитектуры управления и выборе алгоритмических методов [14–17].

Одним из основных ограничений является вычислительная сложность. Адаптивное управление требует регулярного анализа состояния системы, оценки множества вариантов распределения ресурсов и пересчёта управляющих воздействий при изменении параметров среды. В крупномасштабных распределённых системах число возможных конфигураций может быть очень большим, что приводит к росту вычислительных затрат на принятие решений. В связи с этим на практике часто применяются приближённые, эвристические или гибридные методы, позволяющие находить допустимые решения за ограниченное время.

Вторым существенным ограничением являются задержки в принятии и реализации управляющих решений. Даже если алгоритм управления формирует корректное решение, его практическое применение может требовать времени: миграция задач, масштабирование ресурсов, перестройка маршрутов передачи данных и синхронизация состояния компонентов не выполняются мгновенно. В высокодинамичных системах задержка между обнаружением изменения состояния и применением управляющего воздействия может привести к снижению эффективности адаптации [5, 12].

Третьим ограничением является возможная нестабильность управления при использовании неточных или неадекватных моделей. Если модель состояния системы, прогнозная модель или функция качества не отражают реальную динамику вычислительной среды, адаптивные решения могут приводить к колебаниям нагрузки, избыточному перераспределению задач или ухудшению показателей качества обслуживания. Особенно актуальна данная проблема для интеллектуальных методов управления, основанных на обучаемых моделях.

Четвёртым ограничением является зависимость адаптивного управления от качества данных. Для корректного формирования управляющих воздействий необходимы своевременные, полные и

достоверные данные о состоянии системы. Ошибки мониторинга, задержки поступления метрик, неполнота наблюдаемой информации и шум в данных могут снижать качество решений и приводить к некорректной оценке состояния системы [15, 16].

Кроме того, адаптивные методы усложняют верификацию и сопровождение системы управления. В отличие от статических алгоритмов, поведение адаптивной системы может изменяться во времени, что затрудняет формальное доказательство корректности, прогнозирование поведения в нестандартных ситуациях и обеспечение воспроизводимости результатов. Следовательно, применение адаптивного управления требует баланса между сложностью управляющих алгоритмов, скоростью принятия решений, качеством используемых данных и устойчивостью модели управления. На практике наиболее эффективными оказываются архитектуры, в которых адаптивные методы сочетаются с механизмами мониторинга, ограничениями безопасности, правилами допустимого поведения и процедурами оценки качества принимаемых решений [14–17].

Таким образом, адаптивное управление не устраняет полностью сложность управления гетерогенными распределёнными вычислительными системами, но предоставляет более гибкий и эффективный инструмент работы с динамикой, неопределённостью и изменяющимися ограничениями. Указанные ограничения должны учитываться при переходе к мультиагентным и интеллектуальным архитектурам управления.

Выводы по главе 3

В главе рассмотрены принципы адаптивного управления гетерогенными распределёнными вычислительными системами и выполнена интеграция методов адаптивного управления с формальной моделью ГРВС, разработанной в предыдущей главе [12–15]. Показано, что динамический характер распределённых вычислительных сред, изменяющаяся структура ограничений и неполная наблюдаемость состояния системы делают применение статических методов управления недостаточно эффективным.

Установлено, что управление ГРВС должно рассматриваться как непрерывный процесс адаптивного формирования и корректировки управляющих воздействий на основе анализа текущего состояния системы, прогнозирования её динамики и оценки результатов ранее принятых решений. В отличие от классических подходов, параметры и структура управления в адаптивных системах изменяются во времени в зависимости от состояния вычислительной среды [5, 14].

Предложен формальный контур адаптивного управления, включающий этапы мониторинга, анализа, планирования и исполнения

управляющих воздействий. Показано, что замкнутый характер контура обеспечивает возможность непрерывной корректировки решений и реализации обратной связи, необходимой для функционирования системы в условиях изменяющейся нагрузки и неопределённости.

Выполнена интеграция адаптивного управления с формальной моделью ГРВС. Показано, что задача управления может быть представлена как задача динамической многокритериальной оптимизации:

$$\min F(X, S^{\wedge}(t))$$

при ограничениях $g_i(X, S^{\wedge}(t)) \leq b_i$.

Данная постановка отражает зависимость допустимых решений и критериев эффективности от текущего состояния распределённой вычислительной системы.

Рассмотрены основные классы методов адаптации, включая реактивные, проактивные и обучающиеся подходы. Показано, что наиболее перспективными являются методы, сочетающие прогнозирование состояния системы, интеллектуальный анализ данных и корректировку параметров управления на основе накопленного опыта [15–17].

Особое внимание уделено методам принятия решений в условиях неопределённости. Проанализированы эвристические, оптимизационные и вероятностные подходы к управлению распределёнными вычислительными системами. Установлено, что в условиях высокой динамики и неполной информации наибольшую эффективность демонстрируют гибридные интеллектуальные методы управления.

Рассмотрены принципы предиктивного управления, самоорганизации и самонастройки распределённых вычислительных систем. Показано, что современные адаптивные архитектуры должны обеспечивать не только реакцию на текущие изменения состояния системы, но и возможность прогнозирования, автоматической перестройки структуры управления и изменения параметров функционирования системы во времени.

Одновременно выявлены ограничения адаптивного управления, связанные с вычислительной сложностью, зависимостью от качества данных, задержками принятия решений и риском неустойчивости при использовании неточных моделей. Это подтверждает необходимость разработки распределённых механизмов координации и интеллектуального управления, способных функционировать в условиях ограниченной информации и высокой динамики среды.

Таким образом, результаты главы формируют теоретическую основу для перехода к мультиагентному подходу к управлению гетерогенными распределёнными вычислительными системами, рассматриваемому в следующей главе монографии.

ГЛАВА 4. МУЛЬТИАГЕНТНЫЙ ПОДХОД К АДАПТИВНОМУ УПРАВЛЕНИЮ ГЕТЕРОГЕННЫМИ РАСПРЕДЕЛЁННЫМИ ВЫЧИСЛИТЕЛЬНЫМИ СИСТЕМАМИ

4.1. Необходимость мультиагентного подхода

В предыдущих главах была сформирована формальная модель гетерогенной распределённой вычислительной системы и показано, что задача управления ГРВС представляет собой задачу динамической многокритериальной оптимизации:

$$\min F(X, S(t)),$$

при ограничениях $g_i(X, S(t)) \leq b_i$. В практическом контуре управления вместо полного состояния $S(t)$ используется его оценка $\hat{S}(t)$, формируемая на основе данных наблюдения.

Управление распределённой вычислительной системой осуществляется в условиях изменяющейся нагрузки, неполной информации и динамической структуры ограничений [12–15]. Состояние вычислительной среды непрерывно изменяется во времени, а эффективность управляющих воздействий зависит от текущего состояния вычислительных ресурсов, сетевой инфраструктуры и потоков данных.

Проведённая формализация показывает, что задача управления ГРВС характеризуется рядом фундаментальных особенностей:

- высокой размерностью пространства состояний и управляющих воздействий;
- динамическим характером ограничений;
- распределённой природой вычислительных ресурсов и данных;
- необходимостью принятия решений в режиме, близком к реальному времени;
- неполной наблюдаемостью состояния системы;
- высокой степенью гетерогенности вычислительной среды.

В подобных условиях использование полностью централизованных архитектур управления становится ограниченным. По мере роста числа вычислительных узлов, потоков данных и параметров среды вычислительная сложность централизованного управления возрастает нелинейно, что приводит к увеличению времени принятия решений и снижению масштабируемости системы [8, 13].

Дополнительной проблемой централизованных архитектур является возникновение узких мест в управляющем контуре. Центральный компонент управления вынужден агрегировать информацию о состоянии всей вычислительной инфраструктуры, координировать распределение ресурсов и обеспечивать согласованность принимаемых решений. В

условиях высокой динамики нагрузки это приводит к росту коммуникационных задержек и ухудшению устойчивости управления.

Существенным ограничением централизованного подхода является и снижение отказоустойчивости распределённой вычислительной среды. Наличие единого центра принятия решений повышает зависимость всей инфраструктуры от состояния управляющего узла. Его перегрузка или отказ способны привести к деградации процессов распределения ресурсов и потере управляемости системы в целом [10, 14].

Отдельную сложность представляет распределённый характер информации о состоянии вычислительной среды. Данные о загрузке вычислительных узлов, доступности ресурсов, параметрах сетевых взаимодействий и потоках задач локализованы в различных компонентах системы и изменяются асинхронно. Передача всей информации в единый центр управления требует значительных коммуникационных затрат и повышает вероятность принятия решений на основе устаревших данных.

Следовательно, эффективное управление гетерогенными распределёнными вычислительными системами требует перехода от централизованных схем управления к распределённым архитектурам, способным принимать решения локально и координировать поведение отдельных компонентов вычислительной среды.

Наиболее естественным развитием данного подхода является использование мультиагентной модели управления. В мультиагентной системе процессы управления распределяются между множеством взаимодействующих агентов, каждый из которых обладает собственной областью ответственности, локальной информацией о состоянии среды и механизмами принятия решений [15–17].

Использование мультиагентного подхода позволяет:

- распределить вычислительную нагрузку управления;
- уменьшить зависимость от централизованных компонентов;
- повысить масштабируемость распределённой системы;
- обеспечить локальную адаптацию к изменениям среды;
- повысить устойчивость к отказам отдельных компонентов;
- реализовать параллельное принятие решений;
- учитывать распределённый характер информации о состоянии

системы.

Важным преимуществом мультиагентных архитектур является возможность сочетания локальной автономности и глобальной координации. Отдельные агенты способны принимать решения на основе локальной информации о состоянии вычислительной среды, тогда как механизмы взаимодействия обеспечивают согласованность функционирования системы в целом.

При этом эффективность мультиагентного управления напрямую зависит от уровня наблюдаемости распределённой вычислительной среды.

Агенты должны обладать возможностью оперативного получения информации о состоянии вычислительных узлов, сетевых взаимодействий, потоков задач и ограничений инфраструктуры. В дальнейшем это создаёт основу для интеграции интеллектуальных методов анализа, прогнозирования и адаптивного управления.

Таким образом, переход к мультиагентному подходу является закономерным следствием особенностей гетерогенных распределённых вычислительных систем и ограничений централизованных методов управления. Мультиагентные архитектуры формируют основу для построения масштабируемых, адаптивных и интеллектуальных систем управления распределённой вычислительной инфраструктурой, рассматриваемых в последующих разделах монографии.

4.2. Формальная модель мультиагентной системы управления

Для реализации распределённого адаптивного управления гетерогенными распределёнными вычислительными системами вводится формальная модель мультиагентной системы управления. Использование мультиагентного подхода позволяет декомпозировать общую задачу управления на совокупность локальных задач, выполняемых взаимодействующими интеллектуальными компонентами — агентами [14–17].

В рамках рассматриваемого подхода мультиагентная система управления определяется следующим кортежем:

$$MAS = \langle A, E, P, K, Comm \rangle,$$

где $A = \{a_1, \dots, a_p\}$ — множество агентов, E — среда функционирования, соответствующая гетерогенной распределённой вычислительной системе, P — множество политик поведения агентов, K — совокупность знаний, используемых агентами, а $Comm$ — множество коммуникационных протоколов взаимодействия.

Среда E соответствует формальной модели ГРВС, введённой в главе 2, и включает вычислительные узлы, каналы связи, ресурсы, задачи, потоки данных и текущее состояние системы:

$$E = \langle N, L, R, T, D, S(t) \rangle.$$

Здесь обозначение L используется для множества каналов связи, чтобы не смешивать его с обозначением среды E . Такое уточнение необходимо, поскольку в формальной модели мультиагентной системы символ E обозначает управляемую среду в целом, а не множество рёбер или каналов связи.

В данной постановке агенты являются активными компонентами системы управления, взаимодействующими как со средой, так и друг с другом. Каждый агент обладает локальным представлением состояния

системы, механизмом принятия решений и набором допустимых управляющих воздействий. Формально отдельный агент a_i может быть представлен следующим образом:

$$a_i = \langle Obs_i, Dec_i, Act_i, K_i \rangle,$$

где Obs_i — функция наблюдения, Dec_i — функция принятия решений, Act_i — множество выполняемых действий, а K_i — локальные знания агента.

Функция наблюдения Obs_i определяет механизм получения информации о состоянии среды. Агент может наблюдать параметры загрузки вычислительных узлов, состояние очередей задач, характеристики каналов связи, доступность ресурсов и результаты ранее выполненных действий. В большинстве случаев наблюдаемая информация является локальной и неполной, что соответствует реальным условиям функционирования распределённых систем [5, 12].

Функция принятия решений Dec_i определяет способ формирования управляющего воздействия на основе наблюдаемого состояния среды и накопленных знаний. В зависимости от архитектуры системы она может реализовываться с использованием эвристических правил, методов оптимизации, вероятностных моделей, алгоритмов машинного обучения, нейросетевых моделей, LLM-компонентов или механизмов коллективного согласования решений.

Множество действий Act_i определяет допустимые управляющие воздействия агента. К таким воздействиям относятся назначение задач вычислительным узлам, изменение приоритетов обработки, миграция сервисов, перераспределение потоков данных, масштабирование ресурсов, запуск процедур восстановления и инициирование взаимодействия с другими агентами.

Компонента K_i описывает знания агента о системе. Она может включать локальные модели состояния среды, историю функционирования, прогнозы нагрузки, правила принятия решений, параметры обученных моделей и информацию о взаимодействии с другими агентами. В отличие от централизованных систем, в мультиагентной архитектуре знания распределены между агентами и формируются на основе локального опыта, наблюдений и коммуникации.

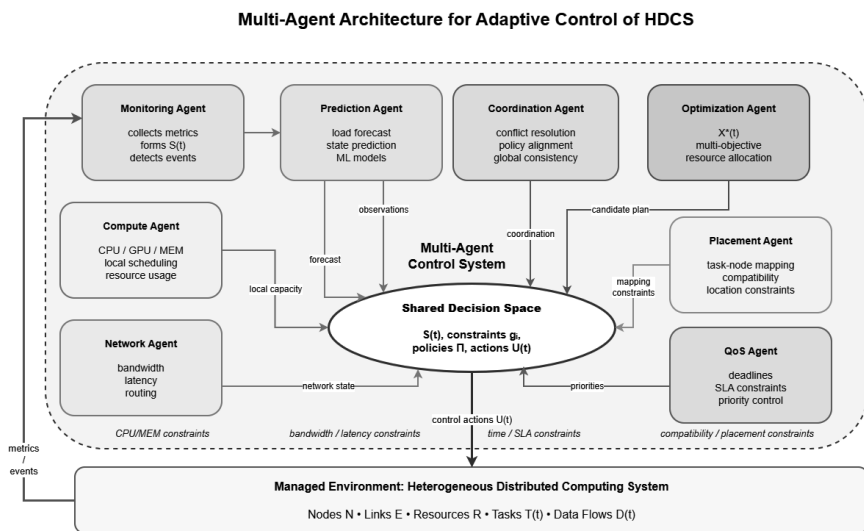
Особенностью мультиагентного подхода является то, что ни один агент не обязан обладать полной информацией о состоянии всей системы. Глобальное поведение формируется в результате локальных решений, обмена информацией и согласования действий между агентами [15–17]. Это позволяет снизить объём централизованной обработки данных и приблизить принятие решений к тем компонентам системы, где возникают изменения состояния.

Важную роль в мультиагентной системе играют коммуникационные протоколы C , обеспечивающие обмен информацией и координацию

действий агентов. Они могут включать передачу сообщений о состоянии системы, управляющие запросы, согласование распределения ресурсов, коллективное принятие решений, переговорные процедуры и механизмы кооперации. Использование коммуникационных протоколов позволяет обеспечить согласованность локальных решений и снизить вероятность конфликтов при распределении ресурсов.

Политики поведения Π задают общие правила функционирования агентов и определяют стратегии принятия решений в различных режимах работы системы. Политики могут быть фиксированными, адаптивными или обучаемыми и изменяться в процессе функционирования ГРВС. В адаптивной системе управления именно изменение политик поведения позволяет агентам корректировать свои действия при изменении нагрузки, доступности ресурсов, состояния сети и требований к качеству обслуживания.

Рисунок 4 иллюстрирует пример функциональной конкретизации формальной модели мультиагентной системы управления. В данной архитектуре специализированные агенты мониторинга, прогнозирования, вычислительных ресурсов, сети, размещения, качества обслуживания, оптимизации и координации взаимодействуют через общее пространство решений, формируя управляющие воздействия на гетерогенную распределённую вычислительную среду.



Multi-agent control architecture: specialized agents observe, optimize, coordinate, and execute adaptive control actions

Рисунок 4 – Мультиагентная архитектура управления

Таким образом, мультиагентная система управления представляет собой распределённую интеллектуальную среду, в которой управление формируется посредством координированного взаимодействия множества автономных агентов. Такая архитектура обеспечивает масштабируемость, адаптивность и устойчивость управления гетерогенными распределёнными вычислительными системами в условиях высокой динамики, неполной наблюдаемости и изменяющейся структуры ограничений.

4.3. Декомпозиция задачи управления

Одним из ключевых преимуществ мультиагентного подхода является возможность декомпозиции глобальной задачи управления распределённой вычислительной системой на совокупность локальных задач, решаемых отдельными агентами [14–17]. Такая декомпозиция позволяет снизить вычислительную сложность управляющего процесса, уменьшить объём централизованной обработки данных и обеспечить масштабируемость системы при росте числа вычислительных узлов, потоков задач и параметров состояния среды.

В рамках общей постановки задача управления ГРВС была представлена в виде:

$$\min F(X, S(t))$$

при ограничениях

$$g_i(X, S(t)) \leq b_i.$$

В централизованной архитектуре решение данной задачи требует одновременного анализа всего пространства состояний, управляющих воздействий и ограничений. Однако в распределённой вычислительной среде значительная часть информации локализована в отдельных компонентах системы, а многие ограничения имеют локальный или частично локальный характер. Это создаёт предпосылки для распределения функций управления между специализированными агентами.

В мультиагентной модели глобальная целевая функция может быть представлена как композиция локальных целевых функций:

$$F = \sum_k F_k,$$

где F_k — локальная целевая функция, связанная с областью ответственности агента или группы агентов. Такая постановка не означает механического разбиения исходной задачи на полностью независимые части. Напротив, она отражает возможность выделения локальных аспектов управления, каждый из которых может анализироваться и оптимизироваться отдельным агентом с учётом общесистемных ограничений.

Локальные целевые функции могут быть связаны с балансировкой нагрузки в отдельном кластере узлов, управлением сетевыми потоками,

минимизацией времени выполнения задач определённого класса, снижением энергопотребления, обеспечением устойчивости локальной подсистемы или контролем требований качества обслуживания. В результате каждый агент получает возможность работать с подзадачей меньшей размерности по сравнению с глобальной задачей управления всей ГРВС [12, 15].

Аналогичным образом между агентами может быть распределена система ограничений:

$$g_i \rightarrow a_j,$$

где ограничение g_i закрепляется за агентом a_j , ответственным за контроль соответствующего параметра системы. Например, агент вычислительных ресурсов контролирует ограничения загрузки процессоров и памяти; сетевой агент отвечает за пропускную способность каналов связи; агент качества обслуживания анализирует временные ограничения и SLA; агент энергопотребления контролирует энергетические параметры; агент устойчивости отслеживает показатели отказоустойчивости и резервирования.

Таким образом, каждый агент отвечает за определённую часть состояния системы, группу ограничений или класс управляющих воздействий. Это позволяет приблизить процесс принятия решений к тем компонентам вычислительной среды, где непосредственно возникают изменения состояния и формируются локальные управляющие потребности.

Следует подчеркнуть, что декомпозиция задачи управления не означает полной независимости агентов. Ограничения и локальные целевые функции в ГРВС, как правило, взаимосвязаны. Решение агента балансировки нагрузки может изменить сетевые потоки, повлиять на энергопотребление и изменить время выполнения задач. Поэтому мультиагентная декомпозиция должна дополняться механизмами координации, обмена информацией и согласования локальных решений [14–17].

С формальной точки зрения декомпозиция задачи управления приводит к переходу от единственной централизованной задачи оптимизации к распределённой системе локальных задач:

$$X_i^* = \arg \min_{X_i} F_i(X_i, \hat{S}_i(t))$$

при локальных ограничениях

$$g_i(X_i, \hat{S}_i(t)) \leq b_i.$$

Здесь X_i^* и $\hat{S}_i(t)$ соответствуют локальной области управления агента a_k . При этом глобальное управление формируется не простым суммированием локальных решений, а через их согласование в рамках

коммуникационной и координационной структуры мультиагентной системы.

Важным преимуществом данного подхода является возможность параллельного принятия решений. Отдельные агенты могут одновременно анализировать состояние своих областей ответственности, формировать управляющие воздействия и обмениваться информацией с соседними или координирующими агентами. Это повышает скорость адаптации системы к изменению нагрузки и уменьшает задержки принятия решений в управляющем контуре.

Кроме того, декомпозиция задачи управления повышает устойчивость системы. Отказ или перегрузка отдельного агента не приводит к полной потере управляемости, поскольку остальные агенты продолжают выполнять локальные функции управления. При наличии механизмов перераспределения ответственности такая архитектура позволяет сохранять частичную управляемость даже при деградации отдельных компонентов системы.

Таким образом, декомпозиция глобальной задачи управления является фундаментальным механизмом построения мультиагентных систем управления ГРВС. Она обеспечивает снижение вычислительной сложности, распределение ответственности между агентами, повышение масштабируемости и возможность локальной адаптации к изменениям состояния распределённой вычислительной среды. При этом эффективность такой декомпозиции определяется не только корректным выделением локальных задач, но и качеством механизмов координации, обеспечивающих согласованность поведения системы в целом.

4.4. Классы агентов в системе управления

На основе формальной модели гетерогенной распределённой вычислительной системы, представленной в главе 2, вводится типизация агентов мультиагентной системы управления. Разделение агентов по функциональным ролям позволяет структурировать управляющий контур, распределить ответственность между компонентами системы и обеспечить масштабируемость архитектуры управления [14–17].

В рассматриваемой модели каждый класс агентов связан с определённой областью состояния системы, группой ограничений или классом управляющих воздействий. Такое разделение соответствует декомпозиции глобальной задачи управления и обеспечивает возможность локальной адаптации при сохранении согласованности поведения системы в целом.

4.4.1. Агент мониторинга

Агент мониторинга является базовым информационным компонентом мультиагентной системы управления. Его основная задача

заключается в сборе, первичной обработке и передаче данных о состоянии гетерогенной распределённой вычислительной системы другим агентам управляющего контура [5, 12, 15].

Формально функция агента мониторинга может быть представлена следующим образом:

$$\hat{S}(t) = O(Y(t)).$$

где $O(Y(t))$ — функция наблюдения и агрегации данных, $Y(t)$ — поток наблюдаемых данных, а $\hat{S}(t)$ — оценка текущего состояния системы.

В отличие от идеализированной модели, в реальных распределённых системах агент мониторинга формирует не полное состояние $S(t)$, а его приближённую оценку $\hat{S}(t)$. Это связано с задержками сбора данных, неполнотой телеметрии, шумом измерений и распределённым характером источников информации. Поэтому мониторинг в ГРВС должен рассматриваться не только как техническая процедура сбора метрик, но и как механизм формирования наблюдаемого состояния системы.

Агент мониторинга собирает данные о загрузке вычислительных узлов, состоянии памяти и дисковых подсистем, параметрах сетевых каналов, очередях задач, логах, событиях и трассировках. На основе этих данных формируется агрегированное представление состояния среды, используемое агентами прогнозирования, оптимизации, согласования и исполнения.

Агент мониторинга не обязательно принимает непосредственные решения о распределении ресурсов, однако его роль является критической для всей системы управления. Качество данных, формируемых данным агентом, определяет качество решений остальных компонентов мультиагентного контура. Проверка практически всех ограничений

$$g_i(X, S(t)) \leq b_i$$

требует актуальной информации о состоянии системы. Например, ресурсные ограничения невозможно контролировать без данных о текущей загрузке узлов, сетевые ограничения — без информации о состоянии каналов связи, а временные ограничения — без сведений о состоянии очередей и времени выполнения задач.

В мультиагентной архитектуре агент мониторинга может быть реализован как единый логический компонент либо как совокупность локальных агентов, размещённых на отдельных узлах или кластерах системы. Второй вариант является более масштабируемым и лучше соответствует природе распределённых вычислительных сред [14–17].

Таким образом, агент мониторинга обеспечивает информационную основу адаптивного управления. Он связывает управляемую вычислительную среду с агентами принятия решений и формирует наблюдаемое состояние, необходимое для корректной работы всего мультиагентного управляющего контура.

4.4.2. Агент вычислительных ресурсов

Агент вычислительных ресурсов отвечает за управление загрузкой вычислительных узлов и контроль соблюдения ресурсных ограничений в пределах своей области ответственности. В зависимости от архитектуры мультиагентной системы такой агент может быть связан с отдельным узлом, группой узлов, кластером или логическим пулом вычислительных ресурсов [12–15].

Основная задача агента вычислительных ресурсов заключается в обеспечении допустимого и эффективного использования процессорной мощности, оперативной памяти, графических ускорителей и других локальных ресурсов. Формально агент контролирует выполнение ограничений вида:

$$\begin{aligned}\sum_i x_{ij} c_i &\leq CPU_j, \\ \sum_i x_{ij} m_i &\leq MEM_j,\end{aligned}$$

где x_{ij} определяет назначение задачи t_i узлу n_j , c_i — потребность задачи в вычислительных ресурсах, m_i — потребность задачи в оперативной памяти, CPU_j и MEM_j — доступные ресурсы узла n_j .

В расширенной модели аналогичные ограничения могут быть сформированы для графических ускорителей, подсистем хранения данных, сетевых интерфейсов, энергопотребления и иных характеристик вычислительного узла. Это особенно важно для гетерогенных сред, в которых узлы различаются не только производительностью, но и архитектурой, набором поддерживаемых технологий и допустимыми режимами эксплуатации [8, 13].

Агент вычислительных ресурсов оценивает текущую загрузку узлов, проверяет допустимость назначения новых задач, выполняет локальную балансировку нагрузки и при необходимости инициирует перераспределение задач. В отличие от централизованной системы управления, он принимает решения на основе локальной информации о состоянии подконтрольных ресурсов, что позволяет сократить задержки принятия решений и уменьшить объём передаваемых данных в управляющем контуре.

При возникновении перегрузки агент может инициировать перенос части задач на менее загруженные узлы, ограничить приём новых задач, изменить приоритеты выполнения или направить запрос координирующему агенту. Однако такое решение не может приниматься изолированно: назначение задачи зависит не только от доступности CPU и памяти, но и от состояния сети, временных ограничений, требований размещения и характеристик входных данных.

Поэтому агент вычислительных ресурсов взаимодействует с агентом мониторинга, сетевым агентом, агентом качества обслуживания, агентом размещения и агентом оптимизации. В совокупности эти взаимодействия

позволяют учитывать ресурсные ограничения как часть общей многокритериальной задачи управления ГРВС.

Таким образом, агент вычислительных ресурсов обеспечивает локальный контроль вычислительной мощности и памяти, поддерживает допустимый уровень загрузки узлов и участвует в формировании согласованной стратегии распределения ресурсов в гетерогенной распределённой вычислительной системе.

4.4.3. Агент сети

Агент сети отвечает за контроль состояния коммуникационной инфраструктуры и управление потоками данных между вычислительными узлами ГРВС. В распределённых вычислительных системах сетевое взаимодействие является одним из ключевых факторов, влияющих на производительность, задержки выполнения задач и эффективность распределения ресурсов [10, 12, 15].

Основное ограничение, контролируемое сетевым агентом, связано с пропускной способностью каналов связи:

$$Load_{ij} \leq B_{ij},$$

где $Load_{ij}$ — текущая нагрузка на канал между узлами n_i и n_j , а B_{ij} — доступная пропускная способность данного канала.

Нарушение данного ограничения приводит к перегрузке сетевой инфраструктуры, росту задержек передачи данных, увеличению времени выполнения распределённых задач и ухудшению качества обслуживания. Поэтому сетевой агент должен не только фиксировать текущее состояние каналов связи, но и участвовать в формировании решений о размещении задач и маршрутизации потоков данных.

Особенность сетевого агента состоит в тесной связи его решений с распределением вычислительных задач. Если две взаимозависимые задачи размещаются на разных узлах, между ними возникает поток данных, создающий дополнительную нагрузку на сеть. Следовательно, сетевой агент должен учитывать не только текущую топологию и параметры каналов, но и структуру зависимостей между задачами, их размещение и объём межузлового обмена.

В динамической среде состояние сетевых каналов изменяется во времени:

$$B_{ij} = B_{ij}(t), Load_{ij} = Load_{ij}(t).$$

Это означает, что сетевой агент должен работать с актуальными данными мониторинга и поддерживать возможность адаптивной маршрутизации. При ухудшении характеристик одного маршрута агент может инициировать передачу данных по альтернативным каналам или передать информацию координирующему агенту для изменения распределения задач.

Сетевой агент особенно важен в системах, где объём межузлового обмена сопоставим с вычислительной нагрузкой. В таких случаях

некорректное размещение задач может приводить к тому, что сетевые задержки становятся основным ограничивающим фактором производительности даже при достаточных вычислительных ресурсах узлов [13, 16].

Таким образом, агент сети обеспечивает адаптивное управление коммуникационной составляющей ГРВС, контролирует пропускную способность каналов, участвует в маршрутизации потоков данных и предотвращает возникновение сетевых перегрузок. Его работа является необходимым условием эффективного управления распределёнными вычислениями в гетерогенной среде.

4.4.4. Агент времени и качества обслуживания

Агент времени, или агент качества обслуживания, отвечает за контроль временных ограничений выполнения задач и соблюдение требований SLA в гетерогенной распределённой вычислительной системе. Его роль особенно важна в системах, где задержки обработки, время отклика и своевременность выполнения задач являются критическими показателями эффективности [10, 12, 15].

Основное ограничение, контролируемое данным агентом, имеет вид:

$$t_i(X) \leq \tau_i,$$

где $t_i(X)$ — фактическое или прогнозируемое время выполнения задачи t_i при заданном распределении ресурсов X , а τ_i — максимально допустимое время выполнения задачи.

Данное ограничение связывает задачу планирования вычислений с требованиями качества обслуживания. Даже если вычислительные и сетевые ресурсы формально достаточны для выполнения задачи, выбранное распределение может быть неприемлемым, если оно приводит к превышению допустимого времени обработки.

В отличие от агентов, контролирующих отдельные ресурсы, агент времени оценивает интегральный результат функционирования системы с позиции пользователя или прикладного сервиса. Время выполнения задачи зависит не только от мощности выбранного узла, но и от состояния очереди, задержек передачи данных, текущей загрузки ресурсов и выбранной стратегии планирования.

В динамической системе агент времени должен учитывать как текущее, так и прогнозируемое время выполнения задач:

$$t_i^{\wedge}(X, t) = f(c_i, m_i, d_i, S(t)),$$

где $t_i^{\wedge}(X, t)$ — оценка времени выполнения задачи, зависящая от характеристик задачи и текущего состояния системы.

При угрозе нарушения SLA агент времени может инициировать изменение приоритетов, перенос задачи на более производительный узел, выделение дополнительных ресурсов или пересмотр маршрутов передачи данных. В мультиагентной архитектуре такие действия требуют

согласования с агентами вычислительных ресурсов, сетевыми агентами и координирующим агентом [14–17].

Таким образом, агент времени обеспечивает контроль временной эффективности управления ГРВС, поддерживает выполнение требований SLA и участвует в формировании приоритетов обработки задач. Его включение в мультиагентную систему позволяет учитывать не только ресурсные и сетевые ограничения, но и требования качества обслуживания при принятии управляющих решений.

4.4.5. Агент размещения

Агент размещения отвечает за выбор допустимых вычислительных узлов для выполнения задач с учётом ограничений совместимости, локализации данных, требований безопасности и особенностей вычислительной среды. В гетерогенных распределённых вычислительных системах задача размещения имеет самостоятельное значение, поскольку не каждая задача может быть выполнена на любом доступном узле [12–15].

Основное ограничение, контролируемое агентом размещения, имеет вид:

$$X(t_i) \in S_i,$$

где $X(t_i)$ — узел или множество узлов, выбранных для выполнения задачи t_i , а S_i — множество допустимых узлов для данной задачи.

Множество S_i может формироваться на основе различных факторов: аппаратной совместимости, наличия необходимых программных библиотек, поддержки требуемой среды выполнения, доступности специализированных ускорителей, расположения исходных данных, требований безопасности, географических ограничений и политик использования ресурсов.

В отличие от агента вычислительных ресурсов, который оценивает возможность выполнения задачи с точки зрения текущей загрузки и доступности CPU, памяти или GPU, агент размещения определяет принципиальную допустимость выполнения задачи на конкретном узле. Например, задача может требовать определённой архитектуры процессора, наличия графического ускорителя, доступа к локальному хранилищу данных или выполнения в определённой программной среде.

В динамической ГРВС множество допустимых узлов также может изменяться во времени:

$$S_i = S_i(t).$$

Это связано с изменением доступности вычислительных ресурсов, отказами узлов, миграцией данных, изменением политик безопасности или появлением новых вычислительных компонентов в системе. Поэтому агент размещения должен учитывать не только статические характеристики узлов, но и текущее состояние среды.

При формировании решений агент размещения взаимодействует с агентом мониторинга, агентом вычислительных ресурсов, сетевым

агентом, агентом времени и агентом оптимизации. Такое взаимодействие необходимо потому, что допустимость размещения задачи должна сочетаться с эффективностью её выполнения, соблюдением временных ограничений и минимизацией сетевых издержек [14–17].

Таким образом, агент размещения обеспечивает выполнение логических, программных, инфраструктурных и организационных ограничений, связанных с назначением задач вычислительным узлам. Его использование позволяет исключить недопустимые конфигурации распределения задач и повысить корректность решений, формируемых мультиагентной системой управления.

4.4.6. Агент оптимизации

Агент оптимизации является одним из ключевых компонентов мультиагентной системы управления и отвечает за формирование согласованных решений по распределению ресурсов с учётом глобальных критериев эффективности функционирования гетерогенной распределённой вычислительной системы [12–15].

В отличие от агентов, ориентированных преимущественно на локальные ограничения и отдельные подсистемы, агент оптимизации рассматривает систему управления в целом и формирует решения, направленные на достижение общесистемных целей. Его задача может быть представлена следующим образом:

$$X^* = \operatorname{argmin}_X F(X),$$

где X^* — оптимальное распределение задач и ресурсов, а $F(X)$ — целевая функция управления.

В более общем случае агент оптимизации работает с многокритериальной постановкой:

$$X^* = \operatorname{argmin}_X F(X, S(t))$$

при ограничениях

$$g_i(X, S(t)) \leq b_i.$$

Таким образом, агент оптимизации учитывает текущее состояние системы, систему ограничений и результаты локальных решений других агентов.

Особое значение агент оптимизации приобретает при наличии конфликтующих критериев управления. Минимизация времени выполнения задач может приводить к увеличению сетевой нагрузки или энергопотребления, тогда как минимизация стоимости вычислений может ухудшать показатели качества обслуживания. В таких условиях задача управления сводится к поиску компромиссного решения в пространстве многокритериальной оптимизации [14–17].

В мультиагентной архитектуре агент оптимизации не обязательно является полностью централизованным компонентом. В зависимости от структуры системы он может функционировать как единый координирующий агент, иерархическая система агентов, распределённый

механизм коллективной оптимизации или совокупность взаимодействующих локальных оптимизаторов.

Алгоритмическая основа агента оптимизации может включать эвристические методы, генетические и эволюционные алгоритмы, методы многокритериальной оптимизации, распределённые методы поиска, интеллектуальные модели принятия решений и обучаемые алгоритмы адаптивной оптимизации. В работах автора и представителей научной школы рассматривались подходы к распределению ресурсов и управлению вычислительной нагрузкой, основанные на многокритериальной оптимизации, адаптивных стратегиях распределения задач и интеллектуальных механизмах согласования решений [1–3].

Важной особенностью агента оптимизации является его взаимодействие с другими агентами системы. Агент мониторинга предоставляет информацию о состоянии среды, агент вычислительных ресурсов — данные о загрузке узлов, сетевой агент — сведения о состоянии каналов связи, агент времени — информацию о рисках нарушения SLA, а агент размещения — ограничения совместимости и допустимости назначения задач.

Таким образом, агент оптимизации выполняет функцию интеграции локальных представлений о состоянии системы и формирования согласованной стратегии управления распределённой вычислительной инфраструктурой. Его использование позволяет перейти от набора локальных управляющих решений к координированному адаптивному управлению ГРВС.

4.4.7. Агент прогнозирования

Агент прогнозирования отвечает за оценку будущего состояния гетерогенной распределённой вычислительной системы и формирование прогнозной информации, используемой в механизмах предиктивного и адаптивного управления [14–17]. В условиях динамически изменяющейся вычислительной нагрузки и переменного состояния ресурсов прогнозирование становится важным элементом интеллектуального управления распределённой вычислительной средой.

Основная функция агента прогнозирования может быть представлена следующим образом:

$$\hat{S}(t+1) = M(S(t)),$$

где $\hat{S}(t+1)$ — прогнозируемое состояние системы, а M — модель прогнозирования, формируемая на основе текущего состояния вычислительной среды и накопленных данных наблюдения.

В более общем случае прогноз может учитывать историю изменения состояния системы:

$$\hat{S}(t+k) = M(S(t), S(t-1), \dots, S(t-k)).$$

Агент прогнозирования использует данные мониторинга, статистику функционирования системы и результаты предыдущих управляющих

воздействий для оценки вероятного изменения вычислительной нагрузки, состояния сетевых каналов, загрузки ресурсов и параметров качества обслуживания.

В отличие от реактивных агентов, работающих только с текущим состоянием системы, агент прогнозирования ориентирован на оценку будущей динамики вычислительной среды. Это позволяет системе управления принимать решения заблаговременно и предотвращать возникновение критических состояний до их фактического появления.

Прогнозная информация используется агентом оптимизации при выборе стратегии распределения ресурсов, агентом времени для предотвращения нарушения SLA, агентом вычислительных ресурсов для подготовки к росту нагрузки, сетевым агентом для предупреждения перегрузки каналов связи и координирующими агентами при формировании глобальной стратегии управления.

В качестве модели прогнозирования M могут применяться статистические модели временных рядов, регрессионные методы, нейросетевые архитектуры, рекуррентные сети, модели LSTM и Transformer, а также гибридные интеллектуальные модели. Особое значение имеют методы машинного обучения и нейросетевого прогнозирования, позволяющие учитывать сложные нелинейные зависимости между параметрами состояния системы [15, 16].

Следует учитывать, что качество прогнозирования напрямую влияет на эффективность предиктивного управления. Ошибки прогноза могут приводить к избыточному масштабированию ресурсов, преждевременному перераспределению задач или некорректной оценке риска перегрузки системы. Поэтому агент прогнозирования должен функционировать совместно с механизмами обратной связи и корректировки моделей прогнозирования.

Таким образом, агент прогнозирования обеспечивает переход от реактивного управления к проактивному и предиктивному управлению состоянием ГРВС. Его использование позволяет учитывать ожидаемую динамику среды и формировать управляющие воздействия с опережением изменения состояния системы.

4.4.8. Агент согласования

Агент согласования обеспечивает координацию локальных решений, формируемых различными агентами мультиагентной системы управления, и предотвращает возникновение конфликтов между ограничениями, целями и управляющими воздействиями. Его роль особенно важна в гетерогенных распределённых вычислительных системах, где отдельные агенты принимают решения на основе локальной информации и могут оптимизировать различные, иногда противоречивые, показатели функционирования системы [14–17].

Конфликт в мультиагентной системе управления возникает в том случае, если решение, допустимое с точки зрения одного класса ограничений, оказывается неэффективным или недопустимым с точки зрения другого класса ограничений. Например, агент вычислительных ресурсов может предложить размещение задачи на узле с минимальной загрузкой, однако сетевой агент может выявить, что такое размещение приведёт к перегрузке каналов связи. Аналогично решение, оптимальное по времени выполнения, может оказаться неэффективным с точки зрения энергопотребления или стоимости использования ресурсов.

Формально область потенциального конфликта между ограничениями может быть представлена как пересечение условий допустимости:

$$g_i(X) \cap g_j(X),$$

где $g_i(X)$ и $g_j(X)$ соответствуют различным ограничениям или группам ограничений, контролируемым разными агентами.

В более общем случае согласование должно учитывать зависимость ограничений от текущего состояния системы:

$$g_i(X, S(t)) \cap g_j(X, S(t)).$$

Это означает, что конфликт между решениями может возникать не только из-за структуры задачи, но и вследствие изменения состояния вычислительной среды, сетевой нагрузки, доступности ресурсов или требований качества обслуживания.

Агент согласования не обязательно формирует оптимальное решение самостоятельно. Его задача состоит в обеспечении совместимости локальных решений, вырабатываемых другими агентами, и приведении их к согласованной стратегии управления. В этом смысле он выполняет функцию связующего компонента между локальной автономностью агентов и глобальной целостностью системы управления.

В мультиагентной системе возможны различные механизмы согласования: приоритетные правила, переговорные процедуры, распределённое голосование, согласование через общую целевую функцию, контрактные механизмы и методы коллективной оптимизации [15–17]. Выбор конкретного механизма зависит от масштаба системы, критичности задач, требований к времени принятия решений и допустимого уровня автономности агентов.

В условиях адаптивного управления агент согласования должен работать в тесной связи с агентом мониторинга, агентом оптимизации, агентом прогнозирования и агентами, контролирующими отдельные классы ограничений. Например, при прогнозируемом росте нагрузки он может заранее согласовать перераспределение задач между агентами вычислительных ресурсов и сетевыми агентами, чтобы предотвратить возникновение перегрузок и нарушений SLA.

Таким образом, агент согласования обеспечивает целостность мультиагентного управления ГРВС. Он позволяет объединить локальные решения специализированных агентов в согласованную стратегию управления, учитывающую множество ограничений, динамическое состояние системы и глобальные цели функционирования распределённой вычислительной инфраструктуры.

4.5. Взаимодействие агентов

Эффективность мультиагентного управления гетерогенной распределённой вычислительной системой определяется не только функциональной специализацией отдельных агентов, но и характером их взаимодействия. Поскольку каждый агент обладает собственной областью ответственности и ограниченным представлением о состоянии системы, достижение согласованного глобального поведения требует организации устойчивого обмена информацией, согласования локальных решений и разрешения конфликтов между управляющими воздействиями [14–17].

Взаимодействие агентов может быть формализовано с помощью коммуникационного графа:

$$G_{agent} = (A, L_A),$$

где A — множество агентов мультиагентной системы управления, а L_A — множество каналов взаимодействия между ними.

Вершины графа G_A соответствуют агентам, а рёбра определяют возможность обмена информацией, запросами, управляющими воздействиями или результатами локального анализа. Структура коммуникационного графа может быть централизованной, иерархической, децентрализованной или гибридной. Выбор конкретной структуры определяется масштабом вычислительной среды, требованиями к отказоустойчивости, допустимыми задержками обмена сообщениями и степенью автономности агентов.

Каналы взаимодействия C используются для передачи оценок состояния системы, локальных решений агентов, сообщений о перегрузках и аномалиях, прогнозов изменения нагрузки, ограничений, политик управления, запросов на перераспределение задач и согласованных управляющих воздействий. Таким образом, коммуникационный граф является не вспомогательным элементом, а частью управляющего контура, обеспечивающей переход от локального анализа к общесистемному управлению.

В мультиагентной системе управления можно выделить несколько базовых типов взаимодействия агентов: кооперацию, конкуренцию и согласование. Кооперация предполагает совместное решение общей задачи управления, когда один агент передаёт другому информацию или результат локального анализа:

$$a_i \rightarrow a_j.$$

Например, агент мониторинга передаёт оценку состояния системы агенту прогнозирования, а агент прогнозирования предоставляет агенту оптимизации данные о предполагаемом изменении нагрузки. Такое взаимодействие позволяет объединять локальные знания агентов и формировать более полное представление о состоянии распределённой вычислительной среды.

Конкуренция возникает в ситуациях, когда агенты ориентированы на различные локальные цели или оптимизируют разные критерии:

$$F_i \neq F_j.$$

Например, агент вычислительных ресурсов может стремиться минимизировать загрузку узлов, тогда как агент времени ориентирован на сокращение задержек выполнения задач. В подобных случаях локально оптимальные решения могут вступать в противоречие друг с другом. Конкурентное взаимодействие не является негативным свойством мультиагентной системы, однако оно требует механизмов согласования и разрешения конфликтов.

Согласование направлено на поиск такого управляющего решения X , которое одновременно удовлетворяет ограничениям нескольких агентов:

$$\exists X: g_i(X, S(t)) \wedge g_j(X, S(t)).$$

Данный тип взаимодействия особенно важен при распределении задач, когда необходимо одновременно учитывать ресурсные, сетевые, временные и логические ограничения. Согласование позволяет преобразовать множество локальных решений в общесистемную стратегию управления, сохраняющую допустимость и эффективность функционирования ГРВС.

В практических системах взаимодействие агентов может реализовываться через обмен сообщениями, публикацию событий, распределённые очереди, общие хранилища состояния, контрактные механизмы или протоколы переговоров. При этом важным требованием является ограничение объёма коммуникаций. Избыточный обмен сообщениями может сам становиться источником нагрузки, увеличивать задержки в управляющем контуре и снижать эффективность распределённого принятия решений.

Особое значение имеет устойчивость коммуникационного графа агентов. Отказ отдельного канала взаимодействия или отдельного агента не должен приводить к полной потере управляемости системы. Поэтому в крупных распределённых средах целесообразно использовать избыточные связи, локальные механизмы автономного принятия решений и иерархические схемы координации [15–17].

Таким образом, взаимодействие агентов является механизмом формирования согласованного поведения мультиагентной системы управления. Через коммуникацию, кооперацию, конкуренцию и

согласование локальные решения отдельных агентов преобразуются в единую адаптивную стратегию управления гетерогенной распределённой вычислительной системой.

4.6. Распределённое принятие решений

Распределённое принятие решений является центральным механизмом функционирования мультиагентной системы управления ГРВС. В отличие от централизованной модели, где управляющее воздействие формируется единым управляющим компонентом, в мультиагентной архитектуре итоговое решение возникает как результат взаимодействия локальных решений, вырабатываемых отдельными агентами [14–17].

Пусть каждый агент a_i формирует локальное решение X_i , соответствующее его области ответственности. Тогда глобальное управляющее решение может быть представлено как композиция локальных решений:

$$X(t) = \bigcup_i X_i(t).$$

Такая запись отражает композицию локальных решений, однако итоговое допустимое управляющее воздействие $X^*(t)$ формируется после проверки ограничений и согласования конфликтов. При этом локальные решения могут относиться к различным аспектам управления: размещению вычислительных задач, маршрутизации потоков данных, контролю временных ограничений, управлению очередями, балансировке нагрузки или обеспечению отказоустойчивости.

Однако простое объединение локальных решений не всегда приводит к согласованному глобальному результату. Локальные решения могут быть конфликтующими, избыточными или недопустимыми с точки зрения общесистемных ограничений. Например, решение агента вычислительных ресурсов может быть допустимым по загрузке узла, но приводить к перегрузке сетевого канала или нарушению временных ограничений. Поэтому распределённое принятие решений должно включать механизмы проверки допустимости, координации и итеративного уточнения управляющих воздействий.

Формально процесс распределённого принятия решений может быть представлен как итеративная процедура:

$$X^{(k+1)} = \Phi(X^{(k)}, A, S(t)),$$

где $X^{(k)}$ — текущее приближение глобального решения на k -й итерации, A — множество агентов, участвующих в процессе управления, $S(t)$ —

текущее состояние вычислительной среды, а Φ — функция коллективного обновления решения.

На каждой итерации агенты анализируют состояние своей области ответственности, оценивают допустимость локальных решений, обмениваются информацией с другими агентами и корректируют управляющие действия. Процесс продолжается до достижения согласованного решения либо до выполнения заданных условий остановки. К таким условиям могут относиться ограничение по времени принятия решения, достижение допустимого значения целевой функции, отсутствие конфликтов между локальными решениями или стабилизация состояния системы.

В зависимости от архитектуры мультиагентной системы распределённое принятие решений может реализовываться различными способами: через локальное автономное принятие решений с последующим согласованием, через иерархическую координацию, переговорные процедуры, распределённую оптимизацию, коллективную оценку альтернатив, голосование или приоритетный выбор. Конкретный механизм определяется масштабом ГРВС, требованиями к времени реакции, уровнем критичности задач и допустимой степенью автономности агентов.

Важным преимуществом распределённого принятия решений является возможность параллельной обработки информации. Каждый агент анализирует только свою область ответственности, что позволяет уменьшить вычислительную нагрузку на управляющий контур и повысить скорость реакции системы на изменение среды. Это особенно важно для крупномасштабных ГРВС с большим числом узлов, задач, потоков данных и ограничений [12, 15].

Вместе с тем распределённое принятие решений требует обеспечения согласованности. В отсутствие координации локальные решения могут приводить к неэффективному использованию ресурсов или нарушению ограничений. Несколько агентов могут одновременно выбрать один и тот же ресурс, создать перегрузку сетевого канала, сформировать противоречивые приоритеты обработки задач или инициировать взаимоисключающие действия по перераспределению нагрузки.

Поэтому эффективная мультиагентная архитектура должна включать механизмы проверки допустимости локальных решений, разрешения конфликтов и обновления глобального состояния управления. Эти механизмы обеспечивают переход от множества локальных управляющих воздействий к единой адаптивной стратегии функционирования распределённой вычислительной системы.

Таким образом, распределённое принятие решений позволяет согласовать локальную автономию агентов с глобальными целями управления ГРВС. Оно обеспечивает масштабируемость, устойчивость и возможность адаптации системы к изменяющимся условиям функционирования, выступая одним из ключевых механизмов мультиагентного подхода к управлению гетерогенной распределённой вычислительной средой.

4.7. Архитектуры мультиагентной системы

Архитектура мультиагентной системы управления определяет способ организации агентов, характер их взаимодействия, распределение полномочий и механизм формирования общесистемного управляющего решения. Для гетерогенных распределённых вычислительных систем выбор архитектуры имеет принципиальное значение, поскольку он влияет на масштабируемость, устойчивость, скорость принятия решений и способность системы адаптироваться к изменению состояния вычислительной среды [14–17].

В общем случае можно выделить три базовых типа архитектур мультиагентного управления: иерархическую, децентрализованную и гибридную. Иерархическая архитектура предполагает наличие нескольких уровней управления. На верхнем уровне располагаются агенты, отвечающие за глобальную координацию, стратегическое планирование, оптимизацию и разрешение конфликтов. На нижнем уровне функционируют локальные агенты, связанные с отдельными вычислительными узлами, группами ресурсов, сетевыми сегментами или классами задач.

Преимуществом иерархической архитектуры является возможность сочетать локальный сбор информации и централизованное согласование решений. Такая структура удобна для крупных систем, в которых необходимо сохранять управляемость при большом числе компонентов. Однако чрезмерная централизация верхнего уровня может приводить к появлению узких мест и снижению устойчивости при отказе координирующих агентов.

Децентрализованная архитектура строится на принципе равноправия агентов. В такой системе отсутствует единый управляющий центр, а решения принимаются локально на основе информации, доступной отдельным агентам. Взаимодействие между агентами осуществляется через обмен сообщениями, переговорные процедуры или согласование локальных ограничений. Достоинством децентрализованной архитектуры является высокая устойчивость к отказам и хорошая масштабируемость. Вместе с тем полная децентрализация усложняет достижение глобальной

согласованности решений и может приводить к конфликтам между локальными целями агентов [15, 16].

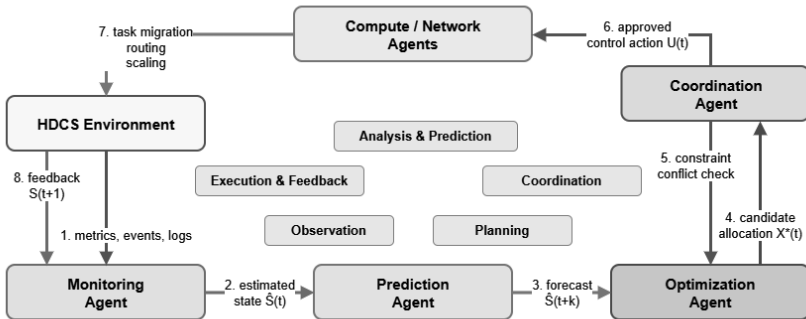
Гибридная архитектура сочетает преимущества иерархического и децентрализованного подходов. В такой системе локальные агенты принимают решения в пределах своей области ответственности, а координирующие агенты обеспечивают согласование решений, контроль глобальных ограничений и формирование общесистемной стратегии управления. Для гетерогенных распределённых вычислительных систем гибридная архитектура является наиболее предпочтительной, поскольку она позволяет сохранить локальную автономность агентов и одновременно обеспечить согласованность управляющих воздействий [14–17].

В рамках гибридной архитектуры локальные агенты могут отвечать за мониторинг, контроль вычислительных ресурсов, сетевые потоки, размещение задач и выполнение временных ограничений. Глобальные или координирующие агенты обеспечивают прогнозирование, оптимизацию, согласование конфликтующих решений и формирование общей стратегии управления. Подобная организация соответствует природе ГРВС, где информация и ресурсы распределены, но управление должно сохранять системную целостность.

Рисунок 5 иллюстрирует пример взаимодействия агентов в гибридной мультиагентной архитектуре адаптивного управления. Управляемая среда передаёт данные мониторинга агенту наблюдения; далее формируется оценка состояния $S^{\wedge}(t)$, выполняется прогноз $S^{\wedge}(t + k)$, строится кандидатное распределение ресурсов $X^*(t)$, после чего координирующий агент проверяет ограничения и конфликты между локальными решениями. Согласованное управляющее воздействие $U(t)$ передаётся агентам исполнения, отвечающим за вычислительные и сетевые ресурсы, а результат выполнения возвращается в контур управления в виде нового состояния $S(t + 1)$.

Таким образом, архитектура мультиагентной системы должна обеспечивать баланс между автономностью агентов и согласованностью глобального управления. Для задач адаптивного управления ГРВС наиболее рациональным является использование гибридной архитектуры, объединяющей локальное принятие решений, распределённую обработку информации, прогнозирование состояния среды, оптимизацию и глобальную координацию действий агентов.

Agent Interaction in the Adaptive Control Process



Interaction sequence of agents during adaptive control: observation, prediction, optimization, coordination, execution, and feedback

Рисунок 5 – Взаимодействие агентов в процессе адаптивного управления

4.8. Связь с адаптивным управлением

Рассмотренный в главе 3 контур адаптивного управления может быть непосредственно интерпретирован в терминах мультиагентной архитектуры. В этом случае отдельные этапы адаптивного управления реализуются специализированными агентами, взаимодействующими в рамках распределённой системы управления [14–17]. Такое соответствие позволяет перейти от абстрактной модели адаптивного управления к практически реализуемой архитектуре распределённого интеллектуального управления гетерогенной распределённой вычислительной системой.

На этапе мониторинга агенты наблюдения осуществляют сбор данных о состоянии вычислительных узлов, сетевых каналов, потоков задач, очередей и используемых ресурсов. Именно эти агенты формируют наблюдаемое состояние системы:

$$\hat{S}(t) = O(S),$$

где $\hat{S}(t)$ представляет собой оценку текущего состояния вычислительной среды, сформированную на основе доступных данных мониторинга, логов, метрик и событий. В условиях распределённой инфраструктуры такая оценка, как правило, является неполной и может содержать задержки, что требует последующей аналитической обработки.

Этап анализа реализуется агентами прогнозирования, согласования и контроля ограничений. На данном уровне выполняется выявление отклонений состояния системы, анализ динамики нагрузки, оценка риска перегрузки вычислительных узлов, прогнозирование состояния среды и

обнаружение конфликтов между локальными решениями. Агенты прогнозирования формируют оценку будущего состояния системы:

$$\hat{S}(t+1) = M(S(t)),$$

где M — модель прогнозирования, построенная на основе текущих и накопленных данных о функционировании ГРВС. Агенты согласования, в свою очередь, обеспечивают совместимость локальных решений и проверяют их допустимость относительно ресурсных, сетевых, временных и логических ограничений.

Этап планирования реализуется агентами оптимизации и координации. Они формируют допустимое управляющее воздействие, направленное на распределение задач, балансировку нагрузки, маршрутизацию потоков данных и изменение параметров функционирования системы. Формально задача планирования может быть представлена следующим образом:

$$X^*(t) = \arg \min_X F(X, \hat{S}(t))$$

при ограничениях

$$g_i(X, \hat{S}(t)) \leq b_i.$$

В отличие от централизованной постановки, в мультиагентной архитектуре поиск решения выполняется с учётом локальных оценок состояния, прогнозов, ограничений и предложений отдельных агентов. Поэтому итоговое решение X^* является результатом не только оптимизации, но и согласования локальных управляющих воздействий.

Этап исполнения реализуется агентами вычислительных ресурсов, сетевыми агентами и агентами размещения. Они выполняют непосредственное применение управляющих воздействий: перераспределение задач, изменение маршрутов передачи данных, локальную балансировку нагрузки, масштабирование ресурсов, перестройку потоков данных и изменение параметров обработки задач. После выполнения управляющего воздействия система переходит в новое состояние $\hat{S}(t+1)$, которое вновь фиксируется агентами мониторинга. Тем самым формируется замкнутый адаптивный контур управления.

Таким образом, мультиагентная система фактически представляет собой распределённую реализацию контура адаптивного управления. Формально это можно представить как отображение текущего состояния системы в совокупность согласованных действий агентов:

$$U(t) = A(S(t)),$$

где A обозначает мультиагентный механизм управления, а $U(t)$ — совокупное управляющее воздействие, формируемое в результате наблюдения, анализа, планирования, согласования и исполнения.

Следует отметить, что в мультиагентной архитектуре отсутствует жёсткое разделение между этапами адаптивного контура. Отдельные

агенты могут одновременно выполнять функции мониторинга, анализа и принятия решений, а распределённый характер взаимодействия позволяет реализовать параллельную обработку информации и локальную адаптацию к изменениям среды [15–17]. Это особенно важно для крупномасштабных ГРВС, где централизованная обработка всех данных и ограничений становится вычислительно и организационно неэффективной.

Связь мультиагентного подхода с адаптивным управлением имеет принципиальное значение. Именно мультиагентная архитектура обеспечивает практическую реализуемость адаптивного управления в гетерогенных распределённых вычислительных системах, поскольку позволяет сочетать локальную автономность компонентов, распределённую обработку информации, координацию решений и динамическую перестройку управляющих воздействий.

Таким образом, мультиагентная система выступает как распределённая интеллектуальная реализация адаптивного управления ГРВС. Она обеспечивает масштабируемость, устойчивость, локальную автономность и возможность динамической координации управляющих воздействий в условиях изменяющейся вычислительной среды.

4.9. Интегральная модель управления

Рассмотренные ранее формальная модель гетерогенной распределённой вычислительной системы, контур адаптивного управления и мультиагентная архитектура позволяют сформировать интегральную модель управления ГРВС. Данная модель объединяет процессы наблюдения, анализа состояния системы, принятия решений, согласования локальных воздействий и выполнения управляющих действий в рамках распределённой интеллектуальной среды [14–17].

В мультиагентной системе каждый агент реализует собственный локальный цикл управления. Он получает информацию о состоянии среды, выполняет локальный анализ, формирует решение и, при необходимости, инициирует управляющее воздействие. Однако глобальное поведение системы не сводится к простой сумме таких локальных действий. Оно возникает как результат их координации, согласования и адаптации к текущему состоянию вычислительной среды.

Интегральная модель управления может быть представлена следующим образом:

$$X(t) = \bigcup_{i=1}^p Act_i(Dec_i(O_i(Y_i(t)))).$$

где O_i — функция наблюдения агента a_i ; Dec_i — функция принятия решений; Act_i — функция формирования управляющего воздействия; $X(t)$

— совокупное управляющее воздействие системы; p — число агентов мультиагентной системы управления.

Данная модель отражает фундаментальную особенность мультиагентного управления: глобальное решение не формируется единым централизованным компонентом, а возникает как результат совместной деятельности множества взаимодействующих агентов. Каждый агент обрабатывает только ту часть информации, которая относится к его области ответственности, но через механизмы коммуникации и координации локальные решения включаются в общесистемное управление.

На первом этапе агент получает локальное представление состояния среды:

$$Obs_i(S(t)).$$

Объём и характер наблюдаемой информации зависят от области ответственности агента, доступных источников данных и структуры коммуникационного графа системы. Для агента вычислительных ресурсов это могут быть параметры загрузки CPU, памяти и ускорителей; для сетевого агента — состояние каналов связи; для агента качества обслуживания — данные о задержках, очередях и рисках нарушения SLA.

На втором этапе агент формирует локальное решение:

$$Dec_i(Obs_i(S(t))).$$

В процессе принятия решений агент может использовать эвристические правила, методы оптимизации, прогнозные модели, механизмы согласования, методы машинного обучения или интеллектуальные модели анализа состояния среды. Выбор конкретного механизма зависит от класса агента, требований к времени реакции и уровня неопределённости наблюдаемого состояния.

На третьем этапе локальное решение преобразуется в управляющее воздействие:

$$Act_i(\cdot).$$

Такое воздействие может включать перераспределение задач, изменение маршрутов передачи данных, корректировку параметров планирования, масштабирование ресурсов, изменение приоритетов обработки или перестройку взаимодействия между компонентами системы.

Совокупность локальных управляющих воздействий формирует глобальное управление $X(t)$. При этом важным условием является их согласованность. Если локальные действия агентов противоречат друг другу или нарушают общесистемные ограничения, они должны быть скорректированы механизмами координации. Поэтому интегральная модель мультиагентного управления должна рассматриваться не только как композиция локальных решений, но и как механизм формирования согласованного управляющего воздействия.

Важным свойством интегральной модели является распределённый характер вычислений и принятия решений. Каждый агент обрабатывает ограниченный фрагмент информации о состоянии среды, а глобальное поведение системы возникает как результат координации локальных решений. Это обеспечивает масштабируемость управления, устойчивость к отказам отдельных компонентов, возможность локальной адаптации, снижение вычислительной нагрузки на управляющий контур и параллельную обработку информации.

Представленная модель допускает как децентрализованное, так и гибридное управление. В децентрализованном варианте агенты обладают большей автономностью, а согласование достигается через коммуникационный граф и локальные протоколы взаимодействия. В гибридном варианте функции Dec_i и Act_i могут включать элементы глобальной координации, коллективной оптимизации и иерархического согласования решений [15–17].

Кроме того, интегральная модель может быть расширена за счёт включения механизмов обучения и накопления знаний:

$$K_i(t+1) = \Phi(K_i(t), S_i(t), X_i(t), S(t+1)).$$

где K_i — знания агента, а Φ — функция обновления модели поведения на основе опыта функционирования системы. Это расширение имеет принципиальное значение для перехода от фиксированных правил управления к интеллектуальным и обучаемым стратегиям поведения агентов.

В такой постановке знания агента могут включать историю наблюдений, результаты ранее принятых решений, параметры прогнозных моделей, правила разрешения конфликтов и оценки эффективности управляющих воздействий. Накопление и обновление этих знаний позволяет агентам адаптировать свои политики поведения к изменяющимся условиям функционирования ГРВС.

Таким образом, интегральная модель управления объединяет формальную модель ГРВС, принципы адаптивного управления и мультиагентный подход в единую распределённую интеллектуальную систему управления. Она создаёт теоретическую основу для последующего рассмотрения интеллектуальных агентов, обучаемых моделей, ZNN-подходов и больших языковых моделей как компонентов мультиагентной архитектуры управления распределённой вычислительной средой.

4.10. Ключевые преимущества мультиагентного подхода

Применение мультиагентного подхода к управлению гетерогенными распределёнными вычислительными системами позволяет преодолеть ряд ограничений централизованных и жёстко детерминированных архитектур

управления. Его преимущества обусловлены тем, что управляющие функции распределяются между множеством автономных, но взаимодействующих компонентов, каждый из которых отвечает за определённую часть состояния системы, группу ограничений или класс управляющих воздействий [14–17].

Одним из наиболее значимых свойств мультиагентного подхода является масштабируемость. При увеличении числа вычислительных узлов, задач и потоков данных мультиагентная система может расширяться за счёт добавления новых агентов, изменения областей ответственности существующих агентов или перестройки коммуникационного графа. Это позволяет избежать чрезмерной концентрации вычислительной и информационной нагрузки в едином центре управления и делает архитектуру пригодной для крупномасштабных распределённых сред.

Существенным преимуществом является также повышение устойчивости управления. Распределение функций между несколькими агентами снижает зависимость системы от одного управляющего компонента. Отказ отдельного агента, нарушение связи с частью инфраструктуры или деградация локальной подсистемы не приводит к полной потере управляемости, поскольку остальные агенты продолжают выполнять локальные функции управления и поддерживать функционирование вычислительной среды [15, 16].

Мультиагентная архитектура обеспечивает локальность принятия решений. Поскольку значительная часть информации о состоянии ГРВС имеет локальный характер, агенты могут принимать решения ближе к источнику данных и объекту управления. Это уменьшает задержки, снижает объём передаваемой информации и повышает оперативность реакции на изменение состояния вычислительных узлов, сетевых каналов, очередей задач и ресурсных ограничений.

Особое значение имеет адаптивность мультиагентного управления. Агенты могут изменять своё поведение в зависимости от локального состояния среды, результатов предыдущих действий, прогнозов изменения нагрузки и информации, полученной от других агентов. В результате мультиагентная система способна реализовывать как реактивную адаптацию к уже возникшим изменениям, так и проактивную адаптацию, основанную на прогнозировании будущих состояний вычислительной среды.

Мультиагентный подход также создаёт естественную основу для интеграции интеллектуальных методов управления. Каждый агент может быть дополнен алгоритмами машинного обучения, прогнозирования, оптимизации, нейросетевыми моделями или механизмами рассуждения. Это позволяет перейти от системы с фиксированными правилами поведения к интеллектуальной мультиагентной архитектуре, в которой

отдельные агенты способны обучаться, прогнозировать изменение состояния среды и участвовать в коллективном принятии решений [16, 17].

Для задач управления ГРВС особенно важным является сочетание локальной автономности агентов и глобальной координации их действий. Локальная автономность обеспечивает быструю реакцию на изменение состояния отдельных компонентов системы, а координационные механизмы позволяют согласовывать локальные решения с общесистемными целями управления. Именно это сочетание отличает мультиагентную архитектуру от простой совокупности независимых локальных регуляторов.

В исследованиях автора и представителей научной школы показано, что мультиагентный подход является эффективным инструментом управления распределёнными вычислительными системами, поскольку позволяет учитывать гетерогенность ресурсов, распределённый характер информации и необходимость согласования множества частных решений [2, 5]. В контексте настоящей монографии данный подход рассматривается не как самостоятельная технология, а как архитектурная основа адаптивного управления ГРВС.

Таким образом, мультиагентный подход обеспечивает методологическую и архитектурную основу для построения масштабируемых, устойчивых и адаптивных систем управления гетерогенными распределёнными вычислительными системами. Он является естественным развитием формальной модели и принципов адаптивного управления, рассмотренных в предыдущих главах, и создаёт основу для последующей интеграции интеллектуальных, нейросетевых и LLM-ориентированных компонентов управления.

Выводы по главе 4

В главе разработана мультиагентная модель управления гетерогенными распределёнными вычислительными системами, основанная на декомпозиции глобальной задачи управления на совокупность локальных задач, решаемых взаимодействующими агентами [14–17]. Показано, что переход к мультиагентной архитектуре является закономерным следствием высокой размерности задачи управления, динамического характера ограничений, неполной наблюдаемости состояния и распределённой природы вычислительных ресурсов и данных.

Выполнена формализация мультиагентной системы управления как совокупности агентов, управляемой среды, политик поведения, локальных знаний и коммуникационных механизмов. Установлено, что каждый агент обладает собственной функцией наблюдения, механизмом принятия решений и набором допустимых управляющих воздействий, а глобальное

поведение системы формируется в результате координации локальных решений.

Показано, что задача управления ГРВС

$$\min F(X, S(t))$$

при ограничениях

$$g_i(X, S(t)) \leq b_i$$

может быть декомпозирована между агентами, отвечающими за различные классы ограничений и отдельные области состояния вычислительной среды. Такая декомпозиция позволяет снизить вычислительную сложность управления, приблизить принятие решений к источникам данных и повысить масштабируемость распределённой вычислительной инфраструктуры.

В главе введена типизация агентов мультиагентной системы управления. Рассмотрены агенты мониторинга, вычислительных ресурсов, сети, времени и качества обслуживания, размещения, оптимизации, прогнозирования и согласования. Для каждого класса агентов определены область ответственности, связь с ограничениями формальной модели ГРВС и роль в общем контуре адаптивного управления [12–15].

Рассмотрены механизмы взаимодействия агентов, включая кооперацию, конкуренцию и согласование локальных решений. Показано, что распределённое принятие решений не сводится к простому объединению локальных управляющих воздействий, а требует проверки допустимости, разрешения конфликтов и итеративного уточнения решений. Именно эти механизмы обеспечивают переход от локальной автономности агентов к согласованному поведению системы в целом.

Проанализированы иерархическая, децентрализованная и гибридная архитектуры мультиагентных систем управления. Установлено, что для задач управления ГРВС наиболее рациональной является гибридная архитектура, сочетающая локальную автономию агентов, распределённую обработку информации и глобальную координацию управляющих воздействий [15–17].

Выполнена интеграция мультиагентного подхода с контуром адаптивного управления. Показано, что этапы наблюдения, анализа, планирования, согласования и исполнения могут быть реализованы специализированными агентами, взаимодействующими в рамках единого распределённого управляющего контура. Это позволяет рассматривать мультиагентную систему как практическую реализацию адаптивного управления ГРВС.

Сформирована интегральная модель мультиагентного управления:

$$X(t) = \bigcup_{i=1}^p \text{Act}_i(\text{Dec}_i(\hat{S}_i(t))).$$

Данная модель объединяет процессы наблюдения, анализа, принятия решений, согласования и выполнения управляющих воздействий в единую распределённую систему управления гетерогенной вычислительной средой.

Показано, что мультиагентный подход обеспечивает масштабируемость, устойчивость, локальность принятия решений, адаптивность и возможность естественной интеграции интеллектуальных методов управления. В исследованиях автора и представителей научной школы подтверждена эффективность мультиагентных подходов для задач управления распределёнными вычислительными системами [2, 5].

Таким образом, результаты главы формируют теоретическую и архитектурную основу для последующего рассмотрения интеллектуальных агентов, обучаемых моделей, нейросетевых методов и больших языковых моделей как компонентов адаптивного управления гетерогенными распределёнными вычислительными системами.

ГЛАВА 5. МЕТОДЫ РАСПРЕДЕЛЕНИЯ И ОПТИМИЗАЦИИ РЕСУРСОВ В ГЕТЕРОГЕННЫХ РАСПРЕДЕЛЁННЫХ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМАХ

5.1. Постановка задачи распределения ресурсов

С учётом формальной модели гетерогенной распределённой вычислительной системы, представленной в главе 2, и мультиагентной декомпозиции управления, рассмотренной в главе 4, задача распределения ресурсов может быть сформулирована как задача динамической многокритериальной оптимизации в условиях изменяющегося состояния вычислительной среды [12–15]. Если предыдущая глава определяла архитектуру распределённого управления, то настоящая глава раскрывает алгоритмические механизмы, с помощью которых формируются управляющие воздействия, распределяются вычислительные задачи и обеспечивается адаптация системы к изменению нагрузки.

В общем виде задача распределения ресурсов определяется следующим образом:

$$\min F(X, S(t))$$

при ограничениях

$$g_i(X, S(t)) \leq b_i,$$

где X — распределение вычислительных задач по ресурсам системы; $S(t)$ — текущее состояние распределённой вычислительной среды; F — целевая функция управления; g_i — ограничения, определяющие допустимость решений; b_i — допустимые значения соответствующих ограничений. В практическом контуре управления вместо полного состояния $S(t)$ используется его оценка $\hat{S}(t)$, формируемая на основе данных наблюдения.

В данной постановке распределение ресурсов может быть представлено как отображение множества задач на множество вычислительных узлов:

$$X: T \rightarrow N.$$

Однако в гетерогенной распределённой системе такое отображение не является простой процедурой назначения задач узлам. Оно должно учитывать текущее состояние вычислительной среды, включая загрузку узлов, доступность ресурсов, состояние сетевых каналов, параметры очередей, требования качества обслуживания, ограничения размещения и совместимости.

Целевая функция $F(X, S(t))$ в общем случае объединяет несколько критериев эффективности функционирования системы: время выполнения задач, уровень загрузки ресурсов, стоимость вычислений, энергопотребление, устойчивость функционирования, соблюдение SLA и

временных ограничений. На практике задача распределения ресурсов в ГРВС почти всегда является многокритериальной, поскольку различные показатели эффективности могут противоречить друг другу. Например, минимизация времени выполнения задач может требовать использования более производительных, но более дорогих вычислительных ресурсов, тогда как снижение стоимости вычислений может приводить к увеличению задержек обработки [14–17].

Система ограничений

$$g_i(X, S(t)) \leq b_i$$

включает ресурсные, сетевые, временные, энергетические и отказоустойчивые ограничения, а также ограничения размещения и совместимости. Принципиально важной особенностью задачи является зависимость ограничений от текущего состояния системы $S(t)$, а в практическом контуре управления — от его оценки $\hat{S}(t)$. Это означает, что множество допустимых решений изменяется во времени под воздействием вычислительной нагрузки, доступности ресурсов, состояния сети и внешних факторов.

Следовательно, задача распределения ресурсов носит динамический характер. Решение, являющееся эффективным в один момент времени, может стать неэффективным или недопустимым при изменении состояния среды. Поэтому распределение ресурсов в ГРВС должно рассматриваться как непрерывный процесс адаптивного управления, а не как однократная процедура статической оптимизации.

Дополнительной особенностью задачи является её высокая вычислительная сложность. Задача распределения ресурсов в гетерогенных распределённых вычислительных системах относится к классу NP-трудных задач [10, 13]. Это связано с большим числом возможных вариантов распределения задач, многомерностью пространства ограничений, необходимостью учёта взаимосвязей между задачами, распределённым характером ресурсов и динамикой состояния системы. При увеличении числа узлов, задач и ограничений размер пространства решений возрастает экспоненциально, что делает невозможным полный перебор допустимых конфигураций в системах реального масштаба.

Вследствие этого для решения задачи распределения ресурсов в ГРВС применяются эвристические методы, многокритериальная оптимизация, эволюционные алгоритмы, распределённые методы поиска, мультиагентные механизмы координации, а также интеллектуальные и обучаемые модели управления. В исследованиях автора и представителей научной школы показано, что наиболее перспективными являются гибридные подходы, сочетающие адаптивное управление, интеллектуальное прогнозирование и распределённое принятие решений [1–5].

Таким образом, задача распределения ресурсов в гетерогенной распределённой вычислительной системе представляет собой сложную динамическую многокритериальную задачу адаптивной оптимизации в условиях изменяющихся ограничений и неполной информации. Её решение требует не только выбора конкретного алгоритма распределения задач, но и построения согласованного механизма управления, способного учитывать текущее состояние среды, прогнозировать его изменение и корректировать управляющие воздействия во времени.

5.2. Декомпозиция задачи между агентами

В мультиагентной системе управления задача распределения ресурсов решается не централизованно, а посредством совокупности локальных решений, принимаемых специализированными агентами в пределах их области ответственности. В главе 4 такая декомпозиция рассматривалась как архитектурный принцип построения системы управления. В настоящей главе она используется как основа для выбора алгоритмических механизмов, применяемых различными агентами при распределении вычислительных задач, управлении потоками данных, контроле ограничений и адаптивной оптимизации [14–17].

С учётом мультиагентной декомпозиции глобальная целевая функция может быть представлена как совокупность локальных критериев:

$$F(X) = \sum_j F_j(X_j),$$

где $F_j(X_j)$ — локальная целевая функция агента a_j , а X_j — локальное распределение ресурсов или задач, формируемое данным агентом.

Каждый агент отвечает за определённый класс ограничений:

$$g_i \rightarrow a_k,$$

то есть ограничение g_i закрепляется за агентом a_k , осуществляющим контроль соответствующего аспекта функционирования системы. Например, агент вычислительных ресурсов контролирует загрузку CPU и памяти, сетевой агент — пропускную способность каналов связи, агент времени — SLA и временные ограничения, агент размещения — совместимость и допустимость назначения задач, а агент энергопотребления — энергетические параметры системы.

В результате каждый агент решает собственную локальную задачу оптимизации:

$$X_k^* = \operatorname{argmin}_{X_k} F_k(X_k)$$

при локальных ограничениях, определяемых его областью ответственности. Такая локальная задача может включать выбор узла для выполнения задачи, балансировку нагрузки, оптимизацию маршрутов передачи данных, минимизацию задержек, контроль использования ресурсов или предотвращение перегрузки отдельных компонентов системы.

При этом локальные решения агентов не являются полностью независимыми. Решение одного агента может изменять состояние среды, наблюдаемое другими агентами. Например, перераспределение задач между вычислительными узлами влияет на сетевые потоки, загрузку памяти, состояние очередей и время выполнения задач. Поэтому декомпозиция задачи распределения ресурсов должна дополняться механизмами согласования локальных решений и координации поведения агентов [12, 15].

Глобальное решение формируется как композиция локальных решений:

$$X(t) = \bigcup_k X_k(t).$$

При этом согласованное управляющее воздействие $X^*(t)$ формируется после проверки ограничений и разрешения конфликтов между локальными решениями. В зависимости от архитектуры системы такая композиция может выполняться через прямое объединение локальных решений, итеративное согласование, координационные агенты, распределённую оптимизацию или переговорные механизмы. Принципиально важно, что итоговое решение не является простой суммой локальных действий: оно должно быть согласовано с общесистемными ограничениями и целями управления.

Декомпозиция задачи распределения ресурсов обеспечивает снижение размерности локальных задач оптимизации, возможность параллельного принятия решений, повышение устойчивости управления и более гибкую интеграцию интеллектуальных методов. Различные агенты могут использовать разные алгоритмы прогнозирования, оптимизации, балансировки нагрузки и машинного обучения в зависимости от специфики решаемой подзадачи. Тем самым мультиагентная архитектура создаёт основу не только для распределения ответственности, но и для алгоритмической специализации компонентов управления.

Рисунок 6 иллюстрирует общую классификацию методов распределения и оптимизации ресурсов, рассматриваемых в последующих разделах главы. В классификации выделены статические, динамические, метаэвристические, нейросетевые и гибридные методы. Такая структура отражает переход от простых алгоритмов назначения задач к адаптивным и интеллектуальным механизмам управления, используемым в современных ГРВС.

Таким образом, декомпозиция задачи распределения ресурсов между агентами является механизмом перехода от единой централизованной задачи оптимизации к распределённой системе локального принятия решений и координации управляющих воздействий. В последующих разделах рассматриваются основные классы методов, которые могут использоваться в такой системе: от статических и простых динамических алгоритмов до гибридных интеллектуальных подходов.

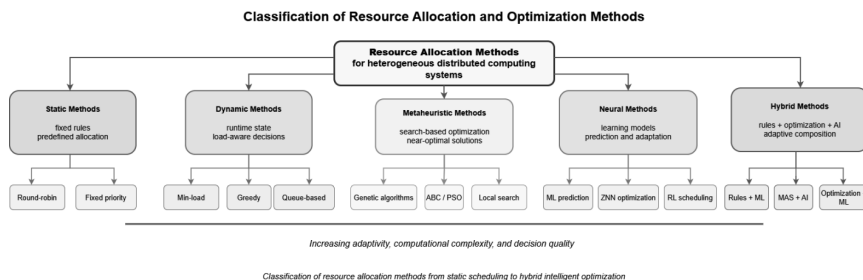


Рисунок 6 – Классификация методов распределения ресурсов

5.3. Статические и простые динамические методы распределения

Базовые методы распределения ресурсов используются как исходные алгоритмы для сравнения с более сложными адаптивными, мультиагентными и интеллектуальными подходами. Их преимущество заключается в простоте реализации, низких вычислительных затратах и предсказуемости поведения. Однако в условиях гетерогенных распределённых вычислительных систем такие методы обладают существенными ограничениями, поскольку слабо учитывают текущее состояние среды, различия между вычислительными узлами, сетевые ограничения и динамику поступления задач [8, 12, 13].

В настоящей работе статические и простые динамические методы рассматриваются не как самостоятельное решение задачи управления ГРВС, а как базовый уровень алгоритмического сравнения. Они позволяют оценить, какой выигрыш дают более сложные механизмы управления: динамические эвристики, многокритериальная оптимизация, мультиагентная координация и интеллектуальные методы адаптации.

5.3.1. Метод round-robin

Одним из наиболее простых методов распределения задач является round-robin. В этом случае задачи последовательно назначаются вычислительным узлам в циклическом порядке:

$$X(t_i) = n_{i \bmod m},$$

где t_i — i -я задача, m — число вычислительных узлов, а $n_{i \bmod m}$ — узел, выбранный по циклическому правилу.

Основным достоинством данного метода является простота реализации. Алгоритм не требует анализа состояния системы, не использует прогнозирование и практически не создаёт дополнительной вычислительной нагрузки на управляющий контур. Благодаря этому round-robin может применяться в простых распределённых системах с приблизительно однородными ресурсами и относительно равномерной нагрузкой.

Однако для гетерогенных распределённых вычислительных систем данный метод имеет существенные ограничения. Он не учитывает различия вычислительной мощности узлов, объём доступной памяти, состояние сетевых каналов, текущую загрузку ресурсов и требования конкретных задач. В результате задачи могут назначаться на недостаточно производительные или уже перегруженные узлы, что приводит к увеличению времени выполнения, росту очередей и снижению общей эффективности системы.

С точки зрения адаптивного управления round-robin представляет собой статический baseline. Он полезен прежде всего как эталонный алгоритм для экспериментального сравнения, поскольку позволяет оценить вклад механизмов наблюдения, анализа состояния среды и динамического перераспределения ресурсов [13, 16].

5.3.2. Метод минимальной загрузки

Метод минимальной загрузки относится к простым динамическим методам распределения задач. В отличие от round-robin, он учитывает текущее состояние вычислительных узлов и назначает новую задачу на наименее загруженный ресурс:

$$X(t_i) = \operatorname{argmin}_j L_j,$$

где L_j — текущая загрузка узла n_j .

Преимуществом данного метода является возможность учитывать фактическое распределение нагрузки в системе. Это позволяет уменьшить вероятность перегрузки отдельных узлов и повысить равномерность использования вычислительных ресурсов по сравнению со статическими алгоритмами.

Вместе с тем метод минимальной загрузки также имеет ограничения. Он ориентирован только на текущее состояние системы и не использует прогноз будущей нагрузки. Кроме того, в базовой форме метод не учитывает гетерогенность вычислительных ресурсов: одинаковое значение загрузки для различных узлов может иметь различный практический смысл, если узлы обладают разной производительностью. Также метод минимальной загрузки не всегда учитывает сетевые ограничения, объём передаваемых данных, требования размещения и временные ограничения задач.

В гетерогенной среде более корректной является нормированная оценка загрузки:

$$L^{\sim}_j = \frac{L_j}{CPU_j}.$$

В этом случае выбор узла осуществляется с учётом относительной, а не абсолютной загрузки вычислительного ресурса. Такая модификация позволяет частично учитывать неоднородность вычислительных узлов,

однако не решает задачу комплексного учёта сетевых, временных и организационных ограничений.

Несмотря на простоту, метод минимальной загрузки является важным базовым подходом, поскольку демонстрирует переход от полностью статического распределения к управлению, использующему текущее состояние среды. Однако для сложных ГРВС его необходимо дополнять механизмами прогнозирования, анализа требований задач, учёта сетевых ограничений и мультиагентной координации [14–17].

5.4. Динамические методы распределения

Динамические методы распределения ресурсов формируют управляющие решения с учётом текущего состояния системы, параметров задач и изменяющихся ограничений вычислительной среды. В отличие от статических методов, они позволяют учитывать загрузку узлов, состояние очередей, сетевые задержки, временные ограничения и приоритеты задач [12–15].

Такие методы занимают промежуточное положение между простыми статическими алгоритмами и интеллектуальными адаптивными подходами. С одной стороны, они остаются достаточно простыми для применения в режиме, близком к реальному времени. С другой стороны, они уже используют наблюдаемое состояние системы и тем самым создают основу для более сложных механизмов адаптивного управления.

5.4.1. Жадные алгоритмы

Жадные алгоритмы относятся к числу наиболее распространённых динамических методов распределения ресурсов. Их основная идея заключается в выборе на каждом шаге такого вычислительного узла, который обеспечивает наилучшее значение локального критерия для текущей задачи.

Формально распределение задачи t_i может быть задано следующим образом:

$$X(t_i) = \operatorname{argmin}_j f(n_j, t_i),$$

где $f(n_j, t_i)$ — функция стоимости назначения задачи t_i узлу n_j .

Функция стоимости может учитывать текущую загрузку узла, доступность вычислительных ресурсов, объём свободной памяти, сетевую удалённость от источника данных, предполагаемое время выполнения задачи, стоимость использования ресурса и риск нарушения временных ограничений. В простейшем случае f зависит только от текущей загрузки узла, однако в гетерогенных распределённых системах более целесообразно использовать комплексную оценку:

$$f(n_j, t_i) = \alpha_1 L_j + \alpha_2 T_{ij} + \alpha_3 C_{ij} + \alpha_4 B_{ij}^{-1},$$

где L_j — загрузка узла, T_{ij} — ожидаемое время выполнения задачи на узле, C_{ij} — стоимость использования ресурса, B_{ij} — характеристика доступной пропускной способности, а α_k — весовые коэффициенты критериев.

Преимуществом жадных алгоритмов является их относительная простота и возможность применения в режиме реального времени. Они позволяют быстро получать допустимые решения без полного перебора пространства вариантов, что особенно важно для крупномасштабных распределённых систем [13, 16].

Ограничение жадных алгоритмов состоит в том, что локально оптимальное решение не всегда приводит к глобально эффективному распределению ресурсов. Например, назначение текущей задачи на наименее загруженный узел может привести к перегрузке сетевого канала, нарушению ограничений размещения или ухудшению условий выполнения последующих задач. Поэтому в сложных ГРВС жадные методы целесообразно использовать не изолированно, а как быстрый локальный компонент гибридных и мультиагентных схем управления [14–17].

В исследованиях автора и представителей научной школы жадные и эвристические методы рассматривались как один из базовых инструментов оптимизации распределения ресурсов в гетерогенных системах, а также как элемент более сложных алгоритмов адаптивного управления [1–3].

5.4.2. Алгоритмы на основе очередей

Алгоритмы на основе очередей используются для управления порядком выполнения задач и распределения вычислительной нагрузки в условиях динамического поступления заявок. В таких алгоритмах особое значение имеют параметры времени ожидания, дедлайна, приоритета и важности задачи [10, 12].

Приоритет задачи может быть задан функцией:

$$Priority(t_i) = f(\tau_i, w_i),$$

где τ_i — допустимое время выполнения или дедлайн задачи, а w_i — весовой коэффициент, отражающий важность задачи.

В более общем случае приоритет может зависеть также от времени ожидания в очереди, вычислительной сложности задачи, объёма данных и риска нарушения SLA:

$$Priority(t_i) = f(\tau_i, w_i, q_i, c_i, d_i),$$

где q_i — время ожидания задачи в очереди, c_i — вычислительная сложность, d_i — объём данных.

Алгоритмы на основе очередей позволяют реализовать различные дисциплины обслуживания: FIFO, priority queue, shortest job first, earliest deadline first, weighted fair queueing и deadline-aware scheduling. Для ГРВС наибольший интерес представляют приоритетные и deadline-aware алгоритмы, поскольку они позволяют учитывать требования качества

обслуживания и минимизировать вероятность нарушения временных ограничений.

Преимуществом методов на основе очередей является их применимость в системах с потоковым поступлением задач. Они позволяют управлять не только распределением задач между узлами, но и порядком их выполнения внутри отдельных вычислительных компонентов.

Ограничение таких методов состоит в том, что они часто оптимизируют локальный порядок обработки задач, но не всегда учитывают глобальные характеристики системы: состояние сети, загрузку соседних узлов, доступность ресурсов и прогноз будущей нагрузки. Поэтому в мультиагентных системах алгоритмы на основе очередей должны использоваться совместно с агентами времени, агентами вычислительных ресурсов, агентами прогнозирования и агентами оптимизации [15–17].

Таким образом, динамические методы распределения ресурсов обеспечивают более гибкое управление по сравнению со статическими алгоритмами, однако требуют развитых механизмов наблюдения, анализа состояния системы и согласования решений. Они являются промежуточным уровнем между простыми эвристическими подходами и интеллектуальными методами адаптивного управления.

5.5. Методы оптимизации

Методы оптимизации занимают центральное место в задачах распределения ресурсов гетерогенных распределённых вычислительных систем, поскольку позволяют формально учитывать множество критериев, ограничений и вариантов размещения задач. В отличие от простых эвристических алгоритмов, которые формируют решение на основе локального правила, оптимизационные методы ориентированы на поиск распределения ресурсов, обеспечивающего более высокое качество управления при заданных условиях функционирования системы [12–15].

Для ГРВС применение оптимизационных методов особенно важно по двум причинам. Во-первых, распределение ресурсов в таких системах связано с большим числом взаимозависимых параметров: производительностью узлов, состоянием сети, временем выполнения задач, стоимостью вычислений, энергопотреблением и ограничениями качества обслуживания. Во-вторых, эффективное решение не всегда может быть найдено с помощью одного локального критерия, поскольку улучшение одного показателя часто приводит к ухудшению другого.

5.5.1. Многокритериальная оптимизация

Задача управления ГРВС практически всегда является многокритериальной. При распределении задач необходимо одновременно

учитывать время выполнения, загрузку ресурсов, сетевые задержки, стоимость вычислений, энергопотребление, надёжность и требования качества обслуживания. Эти критерии могут быть взаимно противоречивыми, поэтому задача управления сводится к поиску компромиссного решения [1–3, 14].

Один из способов формализации многокритериальной задачи состоит в сведении частных критериев к единой агрегированной целевой функции:

$$F(X, \hat{S}(t)) = \sum_k \alpha_k F_k(X, \hat{S}(t)),$$

где F_k — частный критерий качества управления, а α_k — весовой коэффициент, определяющий относительную значимость соответствующего критерия. В динамической ГРВС частные критерии F_k зависят не только от $\hat{\Delta}$ распределения X , но и от текущего состояния среды

$S(t)$ или его оценки $\hat{S}(t)$.

К частным критериям могут относиться время выполнения задач, степень загрузки вычислительных ресурсов, стоимость использования ресурсов, сетевые задержки, энергопотребление, число нарушений SLA и уровень отказоустойчивости. Выбор весовых коэффициентов α_k является отдельной задачей и может зависеть от режима функционирования системы, приоритетов пользователя, стоимости ресурсов или стратегических целей управления.

В работах автора и представителей научной школы рассматривались подходы, связанные с многоатрибутивными методами принятия решений и оптимизацией гетерогенных систем, что создаёт методологическую основу для применения многокритериальных моделей в задачах управления ГРВС [1–3].

Альтернативный подход состоит в использовании парето-оптимизации:

$$Pareto(F_1, F_2, \dots, F_n).$$

В этом случае задача заключается не в сведении всех критериев к одной числовой функции, а в поиске множества решений, для которых невозможно улучшить один критерий без ухудшения хотя бы одного другого. Такой подход особенно полезен в ситуациях, когда невозможно заранее определить точные весовые коэффициенты или когда системе управления требуется множество допустимых компромиссных решений.

Для адаптивного управления ГРВС многокритериальная оптимизация имеет принципиальное значение: она позволяет формализовать компромисс между производительностью, стоимостью, устойчивостью и качеством обслуживания, а также создать основу для

последующей интеграции интеллектуальных и гибридных методов управления.

5.5.2. Генетические алгоритмы

Генетические алгоритмы относятся к классу эволюционных методов оптимизации и применяются для решения задач с большим пространством поиска, сложной структурой ограничений и отсутствием простой аналитической формы целевой функции [13, 16].

В задаче распределения ресурсов решение может быть закодировано в виде вектора:

$$X = (x_1, x_2, \dots, x_n),$$

где элемент x_i определяет размещение задачи t_i на одном из вычислительных узлов системы.

Для оценки качества решения используется функция приспособленности:

$$Fitness(X) = -F(X).$$

Такое определение применяется в случае, когда задача исходно формулируется как минимизация целевой функции $F(X)$. Чем меньше значение $F(X)$, тем выше приспособленность соответствующего решения.

Основными операторами генетического алгоритма являются селекция, кроссовер и мутация. Селекция обеспечивает выбор наиболее приспособленных решений для формирования следующего поколения, кроссовер позволяет комбинировать части различных решений, а мутация вносит случайные изменения и поддерживает разнообразие популяции.

Преимуществом генетических алгоритмов является способность исследовать большое пространство решений и находить приемлемые варианты распределения ресурсов даже в условиях сложных ограничений. Они применимы к задачам, где точные методы оптимизации оказываются вычислительно неприемлемыми или требуют слишком большого времени поиска.

Ограничением генетических алгоритмов является сравнительно высокая вычислительная стоимость и отсутствие гарантии нахождения глобального оптимума за ограниченное время. Поэтому в системах реального времени они редко используются как единственный механизм управления. Более рациональным является их применение в составе гибридных схем, где эволюционный поиск используется для периодической оптимизации, настройки параметров или генерации стратегий, а быстрые эвристические методы обеспечивают оперативное локальное управление.

5.5.3. Эволюционные методы

Эволюционные методы представляют собой более широкий класс алгоритмов, основанных на принципах популяционного поиска, отбора и вариации решений. Помимо генетических алгоритмов, к ним относятся

эволюционные стратегии, дифференциальная эволюция, алгоритмы роевого интеллекта и другие методы коллективного поиска [14–17].

В задачах управления ГРВС эволюционные методы могут использоваться для поиска близких к оптимальным вариантов распределения ресурсов, настройки весов многокритериальной целевой функции, выбора стратегий балансировки нагрузки, оптимизации параметров управляющих алгоритмов и адаптации системы к изменяющимся режимам функционирования.

Особенность эволюционных методов состоит в том, что они не требуют строгой аналитической формы целевой функции и могут работать с дискретными, нелинейными и многокритериальными задачами. Это делает их полезными для гетерогенных распределённых сред, где зависимость между распределением задач и итоговыми показателями эффективности может быть сложной, нестационарной и плохо описываемой простыми аналитическими зависимостями.

Вместе с тем применение эволюционных методов в адаптивном управлении требует контроля времени работы алгоритма. Для оперативного управления распределёнными системами важно, чтобы оптимизационный метод формировал допустимые решения быстрее, чем существенно изменяется состояние среды. Поэтому в практических системах эволюционные алгоритмы часто применяются в гибридных схемах, где глобальный поиск сочетается с быстрыми локальными эвристиками и мультиагентной координацией [15–17].

5.5.4. Локальный поиск

Методы локального поиска основаны на последовательном улучшении текущего решения за счёт перехода к соседним решениям. В задаче распределения ресурсов соседним решением может быть перенос одной задачи на другой узел, обмен задачами между узлами, изменение маршрута передачи данных или корректировка приоритетов выполнения.

Формально соседнее решение может быть задано следующим образом:

$$X' = Neighbor(X),$$

где $Neighbor(X)$ — операция формирования решения, близкого к текущему распределению X .

Далее выбирается решение с меньшим значением целевой функции:

$$X = argmin(F(X), F(X')).$$

Локальный поиск обладает низкой вычислительной сложностью и может использоваться для быстрой корректировки распределения ресурсов при изменении состояния системы. Это делает его удобным инструментом адаптивного управления, особенно в ситуациях, когда требуется не полная перестройка распределения, а небольшое изменение текущего решения.

Недостатком локального поиска является возможность попадания в локальные оптимумы. Для преодоления этого ограничения применяются

модификации, такие как имитация отжига, tabu search, случайные перезапуски и гибридизация с эволюционными методами [13, 16].

В контексте ГРВС локальный поиск особенно важен как механизм оперативной адаптации. Он позволяет корректировать уже сформированное распределение задач при изменении состояния узлов, сетевых каналов или очередей, не выполняя полный пересчёт глобального решения. Поэтому локальный поиск хорошо сочетается с мультиагентной архитектурой, где отдельные агенты могут выполнять локальные улучшения в пределах своей области ответственности.

Таким образом, методы оптимизации позволяют formalизовать и решать задачу распределения ресурсов с учётом множества критериев и ограничений. Для ГРВС наиболее перспективными являются гибридные подходы, сочетающие многокритериальную постановку, эволюционный поиск, локальную адаптацию и мультиагентную координацию решений.

5.6. Управление потоками данных

Управление потоками данных является важной составляющей задачи распределения ресурсов в гетерогенных распределённых вычислительных системах. В таких системах эффективность выполнения вычислительных задач определяется не только производительностью узлов, но и характеристиками сетевого взаимодействия: пропускной способностью каналов, задержками передачи, надёжностью соединений и объёмом межузлового обмена данными [10, 12, 15].

При неэффективной организации потоков данных даже рациональное распределение вычислительных задач может приводить к перегрузке каналов связи, росту задержек и снижению общей производительности системы. Поэтому управление потоками данных должно рассматриваться не как вспомогательная сетевая функция, а как часть общей задачи адаптивного распределения ресурсов.

5.6.1. Маршрутизация

Маршрутизация определяет выбор пути передачи данных между вычислительными узлами. В графовой модели ГРВС путь передачи данных может быть выбран на основе минимизации суммарной стоимости рёбер маршрута:

$$Path = \operatorname{argmin} \sum w(l_{ij}),$$

где $w(l_{ij})$ — вес канала связи между узлами n_i и n_j .

Вес канала может учитывать задержку передачи, текущую загрузку канала, вероятность потери данных, стоимость передачи и доступную пропускную способность. В простейшем случае $w(l_{ij})$ может соответствовать задержке, однако в гетерогенных распределённых системах более целесообразно использовать составную оценку:

$$w(l_{ij}) = \alpha_1 S_{ij} + \alpha_2 Load_{ij} + \alpha_3 P_{ij} + \alpha_4 B_{ij}^{-1},$$

где S_{ij} — задержка передачи, $Load_{ij}$ — текущая загрузка канала, P_{ij} — вероятность потерь, B_{ij} — пропускная способность, а α_k — весовые коэффициенты.

Такое представление позволяет учитывать не только топологию сети, но и её текущее состояние. В условиях динамической нагрузки маршрут, являющийся оптимальным в один момент времени, может стать неэффективным при изменении сетевых параметров. Поэтому маршрутизация в ГРВС должна быть адаптивной и согласованной с текущим распределением вычислительных задач [13, 16].

5.6.2. Балансировка потоков

Балансировка потоков данных направлена на предотвращение перегрузки отдельных каналов связи и более равномерное использование сетевой инфраструктуры. В общем виде задача может быть выражена как минимизация нагрузки на наиболее загруженные каналы:

$$\min \max_{(i,j)} Load_{ij}.$$

Такая постановка особенно важна в системах, где вычислительные задачи имеют значительные объёмы входных, промежуточных или выходных данных. В этих условиях перегрузка отдельных каналов может становиться основным ограничивающим фактором производительности даже при достаточном объёме вычислительных ресурсов.

Практически балансировка потоков может реализовываться за счёт выбора альтернативных маршрутов передачи данных, ограничения интенсивности отдельных потоков, перераспределения задач между узлами, локализации вычислений ближе к данным, разделения крупных потоков на несколько частей и приоритизации критически важных сообщений.

Следует отметить, что балансировка сетевых потоков должна согласовываться с задачей распределения вычислений. Например, перенос задачи на менее загруженный узел может увеличить объём сетевого обмена, если необходимые данные находятся на другом узле. Поэтому эффективное управление потоками данных требует совместной работы сетевых агентов, агентов размещения и агентов оптимизации [14–17].

5.6.3. Разделение потоков

Разделение потоков данных применяется в случаях, когда один информационный поток может быть разбит на несколько частей и передан по различным маршрутам или обработан различными узлами. Формально поток данных может быть представлен как объединение частичных потоков:

$$D = \cup_k D_k,$$

где D_k — часть исходного потока данных.

Разделение потоков позволяет уменьшить нагрузку на отдельные каналы связи, повысить устойчивость передачи данных и использовать параллельные маршруты. Такой подход особенно эффективен в системах с избыточной сетевой топологией и большим объёмом передаваемой информации.

Однако разделение потоков требует дополнительных механизмов синхронизации, сборки данных и контроля целостности. Кроме того, передача частей одного потока по различным маршрутам может приводить к различным задержкам доставки, что необходимо учитывать при планировании вычислений и оценке времени выполнения задач.

В адаптивных системах управления решение о разделении потока должно приниматься с учётом текущего состояния сети, требований задачи и допустимых задержек. Для критичных по времени задач может быть предпочтительна передача по одному более надёжному маршруту, тогда как для больших объёмов данных эффективным может быть параллельное распределение потока по нескольким каналам.

Таким образом, управление потоками данных включает маршрутизацию, балансировку нагрузки на сетевые каналы и разделение потоков. Эти механизмы позволяют учитывать коммуникационную составляющую распределённых вычислений и повышать эффективность управления ГРВС в условиях динамической нагрузки, сетевой гетерогенности и изменяющегося состояния каналов связи.

5.7. Распараллеливание задач

Распараллеливание задач является одним из ключевых механизмов повышения производительности гетерогенных распределённых вычислительных систем. Его основная идея заключается в разделении исходной вычислительной задачи на набор независимых или слабо связанных подзадач, которые могут выполняться параллельно на различных вычислительных узлах [10, 12, 15].

В общем случае задача t_i может быть декомпозирована следующим образом:

$$t_i \rightarrow \{t_{i1}, t_{i2}, \dots, t_{ik}\},$$

где t_{ij} — j -я подзадача, полученная в результате распараллеливания исходной задачи t_i .

Каждая подзадача может быть назначена отдельному вычислительному узлу или специализированному ресурсу:

$$X(t_{ij}) \rightarrow n_j.$$

Распараллеливание позволяет уменьшить общее время выполнения вычислений, повысить степень использования вычислительных ресурсов, распределить нагрузку между узлами системы и использовать

специализированные аппаратные средства. Однако эффективность данного механизма определяется не только числом подзадач, но и тем, насколько корректно они распределены между неоднородными вычислительными ресурсами.

В идеальном случае время выполнения параллельной задачи определяется наиболее длительной подзадачей:

$$T_{exec} = \max_j t_{ij},$$

где t_{ij} — время выполнения i -й подзадачи. Данное выражение отражает тот факт, что завершение всей задачи возможно только после завершения всех её компонент. Следовательно, эффективность распараллеливания зависит от равномерности распределения подзадач и отсутствия критически медленных участков выполнения.

В реальных распределённых системах общее время выполнения зависит не только от вычислительной сложности подзадач, но и от затрат на передачу данных, синхронизацию результатов, сетевые задержки, неоднородность вычислительных ресурсов и зависимости между подзадачами. Поэтому более общая оценка времени выполнения может быть представлена как:

$$T_{exec} = \max_j t_{ij} + T_{comm} + T_{sync},$$

где T_{comm} — время коммуникаций между узлами, а T_{sync} — время синхронизации результатов выполнения.

Для гетерогенных распределённых вычислительных систем особенно важным является учёт различий производительности вычислительных узлов. Если подзадачи распределяются равномерно без учёта характеристик ресурсов, часть узлов может завершать вычисления значительно позже остальных, что приводит к снижению эффективности параллельного выполнения [13, 16].

В связи с этим задача распараллеливания тесно связана с задачей распределения ресурсов. При формировании подзадач необходимо учитывать вычислительную мощность узлов, доступность памяти и специализированных ускорителей, сетевую топологию, объём межзадачного обмена данными, локализацию исходных данных и требования ко времени выполнения. Иначе выигрыш от параллельного выполнения может быть полностью нивелирован коммуникационными и синхронизационными издержками.

В мультиагентной архитектуре задачи распараллеливания могут решаться совместно агентами размещения, агентами вычислительных ресурсов, сетевыми агентами и агентами оптимизации. Такое взаимодействие позволяет адаптировать структуру распараллеливания к текущему состоянию вычислительной среды и выбирать не только число подзадач, но и рациональное размещение их на доступных ресурсах [14–17].

Следует учитывать, что не все задачи эффективно поддаются распараллеливанию. Наличие последовательных участков вычислений, тесных зависимостей между подзадачами и интенсивного обмена данными может существенно ограничивать потенциальный выигрыш от параллельного выполнения. Данное ограничение отражается законом Амдала, согласно которому максимальное ускорение определяется долей последовательных вычислений в общей структуре задачи.

Таким образом, распараллеливание задач является важным механизмом повышения эффективности ГРВС, но его результативность зависит от совместного учёта вычислительных, сетевых и организационных ограничений. В контексте адаптивного управления распараллеливание должно рассматриваться не как изолированная техника ускорения вычислений, а как часть общей задачи распределения ресурсов, балансировки нагрузки и управления потоками данных.

5.8. Алгоритмы в мультиагентной системе

Одной из ключевых особенностей мультиагентной системы управления является возможность использования различных классов алгоритмов в зависимости от функциональной специализации агента. Поскольку отдельные агенты отвечают за различные аспекты функционирования гетерогенной распределённой вычислительной системы, применение единого универсального алгоритма управления оказывается неэффективным [14–17].

В мультиагентной архитектуре каждый агент использует методы, наиболее соответствующие его области ответственности, типу контролируемых ограничений и характеру принимаемых решений. Такой подход позволяет сочетать быстрые локальные эвристики, методы маршрутизации, алгоритмы управления очередями, многокритериальную оптимизацию и прогнозные модели в рамках единого распределённого управляющего контура.

Таблица 5.1 иллюстрирует соответствие между классами агентов, управленческими задачами и используемыми алгоритмическими механизмами.

Агент вычислительных ресурсов (Compute Agent) ориентирован на быстрое локальное принятие решений, поэтому в его работе наиболее часто используются жадные алгоритмы, методы минимальной загрузки и локальная балансировка. Такие алгоритмы обладают низкой вычислительной сложностью и позволяют оперативно реагировать на изменение состояния вычислительных узлов:

$$X(t_i) = \operatorname{argmin}_j f(n_j, t_i).$$

Основная задача данных алгоритмов заключается в поддержании допустимого уровня загрузки вычислительных ресурсов и предотвращении

локальных перегрузок. Однако их применение требует согласования с сетевыми и временными ограничениями, поскольку локально рациональное назначение задачи не всегда является эффективным с точки зрения системы в целом.

Таблица 5.1 — Соответствие классов агентов и алгоритмов управления

Класс агента	Управленческая задача	Используемые методы	Основное ограничение
Compute Agent	распределение вычислительной нагрузки	greedy-алгоритмы, min-load, локальная балансировка	слабый учёт сети и прогноза нагрузки
Network Agent	управление передачей данных	маршрутизация, балансировка потоков	зависимость от текущего состояния сети
QoS Agent	соблюдение временных ограничений	приоритетные очереди, earliest deadline first, deadline-aware scheduling	локальность оценки качества обслуживания
Optimization Agent	поиск согласованного распределения ресурсов	многокритериальная оптимизация, генетические и эволюционные алгоритмы	высокая вычислительная сложность
Prediction Agent	прогноз состояния вычислительной среды	регрессионные модели, временные ряды, ML, LSTM, Transformer	зависимость от качества данных и ошибки прогноза

Сетевой агент (Network Agent) использует алгоритмы маршрутизации и балансировки потоков данных. В его области ответственности находятся выбор маршрутов передачи данных, предотвращение перегрузки каналов связи и адаптация сетевых потоков к изменяющемуся состоянию коммуникационной инфраструктуры:

$$Path = \operatorname{argmin} \sum w(e_{ij}).$$

Для динамических распределённых систем особое значение имеют адаптивные алгоритмы маршрутизации, способные учитывать текущее состояние сети и изменять маршруты передачи данных в режиме, близком к реальному времени [13, 16]. При этом решения сетевого агента должны быть согласованы с размещением задач, поскольку изменение вычислительного узла может существенно изменить объём межузлового обмена.

Агент качества обслуживания (QoS Agent) использует алгоритмы управления очередями и приоритизации задач. Его основной целью является минимизация риска нарушения временных ограничений и SLA. В данной области применяются priority queue, earliest deadline first, weighted fair scheduling и deadline-aware scheduling. Приоритет задачи может определяться как:

$$Priority(t_i) = f(\tau_i, w_i),$$

где учитываются дедлайн задачи и её важность. В более сложных случаях приоритет может зависеть от времени ожидания в очереди, вычислительной сложности задачи, объёма передаваемых данных и риска нарушения SLA. Такие методы позволяют повысить управляемость потока задач, но требуют информации от агентов мониторинга, вычислительных ресурсов и прогнозирования.

Агент оптимизации (Optimization Agent) использует наиболее вычислительно сложные методы управления, ориентированные на поиск глобально эффективных или компромиссных решений. В его работе применяются методы многокритериальной оптимизации, генетические и эволюционные алгоритмы, распределённая оптимизация и гибридные методы. Многокритериальная постановка задачи может быть представлена следующим образом:

$$F(X) = \sum_k \alpha_k F_k(X),$$

либо через поиск множества Парето-оптимальных решений:

$$Pareto(F_1, F_2, \dots, F_n).$$

В работах автора и представителей научной школы подобные подходы использовались при решении задач распределения ресурсов и принятия решений в распределённых вычислительных средах [1–3]. В мультиагентной архитектуре агент оптимизации выполняет не только вычислительную, но и координирующую функцию, поскольку объединяет локальные оценки других агентов в согласованное управляющее решение.

Агент прогнозирования (Prediction Agent) использует методы анализа данных и интеллектуального прогнозирования состояния системы. К числу применяемых методов относятся регрессионные модели, статистические методы, временные ряды, методы машинного обучения, нейросетевые архитектуры, LSTM и Transformer-модели. Прогнозирование состояния системы может быть формализовано следующим образом:

$$S^{\wedge}(t + 1) = M(S(t)),$$

где M — модель прогнозирования. Использование прогнозных моделей позволяет реализовать проактивное управление распределённой вычислительной средой и принимать решения с учётом ожидаемой динамики нагрузки [15–17].

Следует отметить, что в мультиагентной архитектуре алгоритмы различных агентов не функционируют изолированно. Решения, сформированные одним агентом, становятся частью информационной

среды для других агентов. Например, прогноз роста нагрузки влияет на решения агента оптимизации, изменение маршрута передачи данных может потребовать перераспределения задач, а риск нарушения SLA может изменить приоритеты локальной балансировки.

В результате формируется распределённая система совместной обработки информации, в которой локальные алгоритмы объединяются в единую адаптивную архитектуру управления. Преимуществом такого подхода является возможность сочетать скорость локальных эвристик, адаптивность прогнозных моделей, гибкость многокритериальной оптимизации и устойчивость распределённого принятия решений.

Таким образом, мультиагентная система управления позволяет использовать специализированные алгоритмы для различных классов задач и объединять их в согласованный управляющий контур. Это обеспечивает переход от набора отдельных методов распределения ресурсов к интегрированной алгоритмической архитектуре адаптивного управления гетерогенной распределённой вычислительной системой.

5.9. Адаптивная оптимизация

В условиях динамически изменяющейся гетерогенной распределённой вычислительной среды задача распределения ресурсов не может рассматриваться как статическая процедура однократного поиска оптимального решения. Изменение вычислительной нагрузки, состояния сетевых каналов, состава задач и доступности ресурсов требует непрерывной корректировки распределения вычислений и адаптации управляющих воздействий [12–15].

В связи с этим задача управления ресурсами переходит от статической оптимизации к адаптивной оптимизации, в которой решение изменяется во времени в зависимости от текущего и прогнозируемого состояния системы. Центральным элементом такой постановки является не построение распределения «с нуля» на каждом шаге управления, а корректировка уже существующего распределения ресурсов:

$$X(t+1) = X(t) + \Delta X,$$

где $X(t)$ — текущее распределение задач и ресурсов; ΔX — корректирующее изменение распределения; $X(t+1)$ — новое состояние распределения ресурсов.

Корректировка распределения определяется как функция текущего и прогнозируемого состояния системы:

$$\Delta X = f(\hat{S}(t), \hat{S}(t+1)).$$

где $\hat{S}(t)$ — текущее состояние вычислительной среды, а $\hat{S}(t+1)$ — прогнозируемое состояние системы на следующем шаге управления.

Данная постановка отражает фундаментальную особенность адаптивного управления: решение формируется не только на основе текущих параметров среды, но и с учётом ожидаемой динамики изменения состояния системы. Это позволяет перейти от реактивного поведения, при котором система реагирует на уже возникшие перегрузки, к проактивному управлению распределённой вычислительной инфраструктурой [14–17].

Корректировка ΔX может включать перераспределение задач между узлами, изменение приоритетов выполнения, миграцию сервисов, изменение маршрутов передачи данных, масштабирование вычислительных ресурсов, перестройку структуры очередей или изменение параметров балансировки нагрузки. При этом важным требованием является соразмерность корректировки масштабу изменения состояния среды.

Одной из ключевых особенностей адаптивной оптимизации является стремление минимизировать объём изменений распределения ресурсов. Полная перестройка распределения задач может сопровождаться значительными накладными расходами: передачей данных между узлами, повторной инициализацией вычислений, миграцией состояния сервисов, дополнительной сетевой нагрузкой и временной деградацией производительности. Поэтому во многих случаях задача адаптивной оптимизации формулируется как поиск минимального изменения распределения, достаточного для компенсации изменений состояния среды:

$$\min(F(X(t+1), S(t+1)) + \lambda \|\Delta X\|),$$

где λ — коэффициент, определяющий стоимость изменения текущего распределения.

Такая постановка позволяет учитывать не только качество нового распределения, но и цену перехода от текущего состояния к новому. Это особенно важно для распределённых вычислительных систем, в которых миграция задач, перенос данных и изменение маршрутов могут создавать существенные накладные расходы.

В мультиагентной системе адаптивная оптимизация реализуется распределённо. Отдельные агенты формируют локальные корректировки:

$$\Delta X_i = f_i(S_i(t), S_i(t+1)).$$

после чего координирующие механизмы обеспечивают согласование локальных изменений и формирование глобального управляющего воздействия [15–17]. В такой архитектуре агент вычислительных ресурсов может инициировать перенос задач при локальной перегрузке, сетевой агент — изменение маршрутов передачи данных, агент качества обслуживания — изменение приоритетов выполнения, а агент оптимизации — согласование локальных корректировок с общесистемной целевой функцией.

Особую роль в адаптивной оптимизации играют агенты прогнозирования, позволяющие оценивать будущие изменения нагрузки и предотвращать возникновение перегрузок до их фактического появления. Использование прогнозных моделей позволяет заранее перераспределять вычислительные ресурсы, подготавливать дополнительные мощности, изменять маршруты передачи данных, корректировать очереди задач и уменьшать вероятность нарушения SLA.

В задачах управления ГРВС адаптивная оптимизация должна учитывать не только вычислительную эффективность, но и устойчивость системы. Чрезмерно частые изменения распределения ресурсов могут приводить к нестабильности управления, колебаниям нагрузки и росту накладных расходов. Поэтому важным свойством адаптивных алгоритмов является способность различать кратковременные флуктуации состояния среды и устойчивые тенденции изменения нагрузки.

В современных интеллектуальных системах управления адаптивная оптимизация всё чаще реализуется с использованием методов машинного обучения, reinforcement learning, нейросетевых прогнозных моделей, мультиагентной координации и гибридных методов распределённой оптимизации. Однако в рамках ГРВС эти методы должны рассматриваться не как замена классических алгоритмов распределения ресурсов, а как механизм повышения их адаптивности, предиктивности и устойчивости.

В исследованиях автора и представителей научной школы показано, что адаптивные методы распределения ресурсов позволяют повысить эффективность управления распределёнными вычислительными системами в условиях высокой динамики и неопределённости среды [2–5].

Таким образом, адаптивная оптимизация представляет собой непрерывный процесс корректировки распределения ресурсов на основе анализа текущего и прогнозируемого состояния системы. Она обеспечивает возможность динамической перестройки вычислительной среды и является необходимым элементом интеллектуального управления гетерогенными распределёнными вычислительными системами.

5.10. Гибридные методы

Современные задачи управления гетерогенными распределёнными вычислительными системами характеризуются высокой сложностью, многокритериальностью, распределённой природой информации и динамическим изменением состояния среды. В таких условиях использование только одного класса методов управления оказывается недостаточным для обеспечения требуемой эффективности, устойчивости и адаптивности системы [14–17].

Гибридные методы представляют собой подход, при котором различные алгоритмические и интеллектуальные механизмы

объединяются в единую систему управления. Их применение позволяет сочетать быстроедействие эвристик, качество оптимизационных методов, адаптивность прогнозных моделей и устойчивость мультиагентной координации.

Основная идея гибридного подхода заключается в комбинировании:

- эвристических правил и методов машинного обучения;
- алгоритмов оптимизации и прогнозных моделей;
- локальных и глобальных механизмов принятия решений;
- реактивного и проактивного управления;
- детерминированных и вероятностных моделей.

Такое сочетание позволяет использовать сильные стороны различных методов и одновременно компенсировать их ограничения. Эвристики обеспечивают высокую скорость принятия решений, но слабо адаптируются к новой среде. Методы оптимизации позволяют получать более качественные решения, но обладают высокой вычислительной сложностью. ML-модели способны выявлять сложные зависимости в данных, однако требуют обучения и могут быть чувствительны к ошибкам прогноза. Локальные решения обеспечивают оперативность, но не всегда согласованы с глобальными целями управления.

Одним из распространённых направлений гибридизации является сочетание правил и методов машинного обучения. В этом случае базовые механизмы управления реализуются через эвристики или экспертные правила, обеспечивающие предсказуемость и устойчивость системы, а интеллектуальные модели используются для адаптации параметров управления, прогнозирования нагрузки и выбора стратегий оптимизации.

Например, локальная балансировка нагрузки может выполняться с помощью greedy-алгоритма:

$$X(t_i) = \operatorname{argmin}_j f(n_j, t_i),$$

тогда как выбор функции стоимости f или её параметров определяется обучаемой моделью прогнозирования. Подобное сочетание позволяет сохранить высокую скорость принятия решений и одновременно учитывать сложные зависимости между параметрами системы, которые трудно формализовать вручную.

Вторым важным направлением является интеграция методов оптимизации и прогнозирования. В этом случае задача распределения ресурсов решается не только на основе текущего состояния системы, но и с учётом прогнозируемой динамики:

$$X^* = \operatorname{argmin}_X (F(X, \hat{S}(t + 1))).$$

Использование прогнозных моделей позволяет предотвращать перегрузку вычислительных узлов, заранее перераспределять потоки задач, выполнять предиктивное масштабирование, снижать вероятность нарушения SLA и адаптировать структуру распределения ресурсов к

ожидаемой нагрузке. Такие подходы особенно эффективны для систем с выраженной временной динамикой нагрузки, включая облачные платформы, edge-среды и высоконагруженные сервисные инфраструктуры [15–17].

Третье направление гибридации связано с сочетанием локальных и глобальных решений. В мультиагентной архитектуре локальные агенты принимают быстрые решения в пределах своей области ответственности, тогда как глобальные агенты координации и оптимизации обеспечивают согласование локальных действий и поддержание системной целостности.

Формально данная схема может быть представлена следующим образом:

$$X = X_{local} \cup X_{global},$$

где X_{local} — локальные решения агентов, а X_{global} — глобальные корректирующие воздействия.

Такой подход позволяет одновременно обеспечить высокую скорость реакции системы, устойчивость управления, масштабируемость, согласованность распределения ресурсов и адаптацию к изменению состояния среды. В результате формируется распределённая интеллектуальная система, в которой локальные алгоритмы взаимодействуют между собой и совместно обеспечивают адаптивное управление ГРВС.

Особое значение гибридные методы приобретают в интеллектуальных мультиагентных системах управления, где различные агенты могут использовать разные классы алгоритмов: эвристические, оптимизационные, вероятностные, обучаемые, нейросетевые и агентно-ориентированные. В такой архитектуре гибридность проявляется не только на уровне алгоритма, но и на уровне всей системы управления: одни агенты обеспечивают быстрые локальные реакции, другие выполняют прогнозирование, третьи формируют оптимизационные решения, а координирующие компоненты согласуют их с глобальными целями.

В современных исследованиях по интеллектуальному управлению распределёнными вычислительными системами гибридные архитектуры рассматриваются как одно из наиболее перспективных направлений развития. К таким направлениям относятся гибридные AI-системы, нейросимвольные подходы, распределённая интеллектуальная оптимизация, планирование с использованием ML-моделей, мультиагентное обучение с подкреплением и гибридное предиктивное управление. Их общая идея состоит в переходе от изолированных алгоритмов к композиции методов, способных работать в условиях неопределённости, неполной информации и динамического изменения состояния вычислительной среды.

В исследованиях автора и представителей научной школы также показано, что комбинирование адаптивных, интеллектуальных и

оптимизационных методов позволяет повысить эффективность распределения ресурсов и устойчивость функционирования распределённых вычислительных систем в условиях неопределённости и динамической нагрузки [2–5].

Таким образом, гибридные методы представляют собой естественное развитие адаптивного и мультиагентного подходов к управлению ГРВС. Их применение позволяет объединить преимущества различных алгоритмических и интеллектуальных механизмов в единую систему распределённого адаптивного управления. Именно гибридные методы формируют методологический переход от классических алгоритмов распределения ресурсов к интеллектуальным, нейросетевым и агентно-ориентированным механизмам управления, рассматриваемым в последующих главах монографии.

5.11. Связь с исследованиями автора

Представленные в главе методы распределения ресурсов, адаптивной оптимизации и мультиагентного управления опираются на результаты исследований автора и представителей научной школы в области управления гетерогенными распределёнными вычислительными системами, интеллектуального распределения ресурсов и адаптивных вычислительных сред [1–5].

Одним из ключевых направлений этих исследований являлось применение многоатрибутивных методов принятия решений к задачам управления распределёнными вычислительными системами. В рамках данного подхода задача распределения ресурсов рассматривалась как многокритериальная задача выбора, в которой одновременно учитываются производительность вычислительных узлов, загрузка ресурсов, стоимость вычислений, временные ограничения, сетевые характеристики, а также показатели устойчивости и надёжности.

Результаты этих исследований легли в основу многокритериальной постановки задачи распределения ресурсов:

$$F(X) = \sum_k \alpha_k F_k(X),$$

а также подходов к поиску компромиссных решений в пространстве конфликтующих критериев управления.

Важное место в исследованиях занимали методы оптимизации гетерогенных вычислительных систем. Рассматривались модели распределения вычислительной нагрузки между неоднородными ресурсами, отличающимися архитектурой, производительностью, сетевыми характеристиками и режимами функционирования. Полученные результаты показали, что эффективное распределение задач требует учёта не только вычислительной мощности узлов, но и состояния сети,

структуры потоков данных, ограничений размещения и динамики изменения состояния системы [2, 3].

Отдельное направление исследований было связано с мультиагентными моделями управления распределёнными вычислительными средами. В рамках данного направления рассматривались декомпозиция задачи управления между агентами, распределённое принятие решений, локальная адаптация агентов, координация управляющих воздействий и согласование ограничений. Эти результаты стали основой, представленной в монографии мультиагентной архитектуры управления ГРВС, в которой управление формируется как результат взаимодействия специализированных агентов, отвечающих за различные классы ресурсов, ограничений и управляющих воздействий [4, 5].

Существенное значение имеют также исследования, связанные с адаптивным управлением ресурсами распределённых вычислительных систем. В них рассматривались методы динамического перераспределения вычислительной нагрузки, адаптивной балансировки ресурсов, прогнозирования состояния вычислительной среды и корректировки параметров управления в условиях изменяющейся нагрузки и неполной информации.

Полученные результаты показывают, что адаптивные методы управления позволяют повысить эффективность использования ресурсов, уменьшить вероятность перегрузок, повысить устойчивость распределённой системы, улучшить показатели качества обслуживания и сократить время реакции на изменение состояния среды.

Особое значение для дальнейшего развития данного направления имеют исследования, связанные с использованием интеллектуальных методов управления, нейросетевых моделей и прогнозирования состояния распределённых вычислительных систем. Эти работы создают основу для интеграции методов искусственного интеллекта, обучаемых агентов, ZNN-подходов и LLM-агентов как когнитивного уровня мультиагентной архитектуры управления, что рассматривается в последующих главах монографии [4, 5].

Таким образом, представленные в главе методы распределения ресурсов, адаптивной оптимизации и мультиагентного управления являются развитием и обобщением результатов исследований автора и представителей научной школы в области интеллектуального управления гетерогенными распределёнными вычислительными системами. Они формируют методологическую основу для перехода от классических алгоритмов распределения ресурсов к интеллектуальным, нейросетевым и AI-ориентированным механизмам управления распределённой вычислительной инфраструктурой.

Выводы по главе 5

В главе рассмотрены методы распределения ресурсов и управления вычислительной нагрузкой в гетерогенных распределённых вычислительных системах. Показано, что задача распределения ресурсов представляет собой динамическую многокритериальную задачу оптимизации, в которой необходимо одновременно учитывать вычислительные, сетевые, временные, организационные, энергетические и отказоустойчивые ограничения [12–15].

Формализована постановка задачи распределения ресурсов:

$$\min F(X, S(t))$$

при ограничениях

$$g_i(X, S(t)) \leq b_i,$$

где допустимость и эффективность решений зависят от текущего состояния вычислительной среды. Установлено, что данная задача относится к классу вычислительно сложных задач и не может эффективно решаться методами полного перебора в системах реального масштаба.

Рассмотрены основные классы методов распределения ресурсов: статические алгоритмы, простые динамические методы, жадные эвристики, алгоритмы управления очередями, методы многокритериальной оптимизации, генетические и эволюционные алгоритмы, локальный поиск, методы управления потоками данных и механизмы распараллеливания вычислений. Показано, что каждый из этих классов решает отдельный аспект задачи управления, но не обеспечивает полного решения проблемы адаптивного управления ГРВС в условиях высокой динамики и неопределённости среды.

Показано, что статические методы обладают низкой вычислительной сложностью и могут использоваться как базовые алгоритмы сравнения, однако они не учитывают динамику состояния системы и гетерогенность ресурсов. Динамические методы и эвристики позволяют использовать информацию о текущей загрузке, состоянии очередей и параметрах задач, но требуют дополнения механизмами прогнозирования, согласования решений и учёта глобальных ограничений [13, 16].

Особое внимание уделено методам многокритериальной оптимизации и эволюционного поиска. Показано, что данные подходы позволяют учитывать конфликтующие критерии управления и находить компромиссные решения в условиях сложной структуры ограничений. Рассмотренные методы связаны с исследованиями автора и представителей научной школы в области многоатрибутивных методов принятия решений и оптимизации гетерогенных вычислительных систем [1–3].

Рассмотрены механизмы управления потоками данных, маршрутизации и балансировки сетевой нагрузки. Установлено, что коммуникационная составляющая является одним из ключевых факторов эффективности распределённых вычислений и должна рассматриваться совместно с задачей распределения вычислительных ресурсов. Аналогично, распараллеливание задач эффективно только при совместном учёте вычислительных характеристик узлов, сетевых задержек, объёма межузлового обмена и ограничений синхронизации.

Показано, что в мультиагентной системе различные классы агентов используют специализированные алгоритмы управления: агенты вычислительных ресурсов — greedy-алгоритмы и локальную балансировку нагрузки; сетевые агенты — маршрутизацию и управление потоками данных; агенты качества обслуживания — приоритетные очереди и deadline-aware scheduling; агенты оптимизации — многокритериальные и эволюционные методы; агенты прогнозирования — методы машинного обучения и прогнозные модели. Это позволяет объединять локальные алгоритмы в единую распределённую систему адаптивного управления [14–17].

Особое значение имеют адаптивная оптимизация и гибридные методы. Показано, что в динамических ГРВС управление должно рассматриваться как непрерывный процесс корректировки распределения ресурсов:

$$X(t + 1) = X(t) + \Delta X,$$

а не как однократный поиск оптимального решения. При этом важным требованием является минимизация избыточных изменений распределения, поскольку миграция задач, перенос данных и перестройка маршрутов могут создавать значительные накладные расходы.

Установлено, что гибридные подходы являются наиболее перспективным направлением развития интеллектуальных систем управления ГРВС. Они позволяют объединить быстродействие эвристических методов, качество оптимизационных алгоритмов, адаптивность прогнозных моделей, устойчивость мультиагентной координации и возможность проактивного управления вычислительной средой.

Выполнена связь рассмотренных методов с исследованиями автора и представителей научной школы в области многоатрибутивного принятия решений, оптимизации гетерогенных вычислительных систем, мультиагентного управления и адаптивного распределения ресурсов [1–5].

Таким образом, результаты главы формируют методологическую и алгоритмическую основу для перехода к интеллектуальным методам управления распределёнными вычислительными системами. Рассмотренные алгоритмы показывают, что эффективное управление ГРВС требует не изолированного применения отдельных методов, а их

интеграции в адаптивную, мультиагентную и интеллектуальную архитектуру, включающую прогнозирование, обучаемые модели, нейросетевые подходы и большие языковые модели как когнитивные компоненты поддержки анализа, координации и принятия решений в архитектуре управления вычислительной инфраструктурой.

ГЛАВА 6. НАБЛЮДАЕМОСТЬ И ДАННЫЕ В КОНТУРЕ АДАПТИВНОГО УПРАВЛЕНИЯ

6.1. Роль данных в управлении распределёнными системами

В предыдущих главах было показано, что управление гетерогенной распределённой вычислительной системой может быть представлено как отображение:

$$\hat{X}(t) = A(\hat{S}(t)).$$

$\hat{S}(t)$ — состояние системы, A — алгоритм или совокупность алгоритмов управления [12–15].

Данная постановка означает, что качество управляющих воздействий напрямую зависит от качества оценки состояния вычислительной среды. Любое решение о распределении задач, балансировке нагрузки, маршрутизации потоков данных или изменении параметров функционирования системы основывается на информации о текущем состоянии ресурсов, сети, очереди задач и вычислительных процессов.

Следовательно, адаптивное управление возможно только при наличии адекватной, своевременной и достаточно полной информации о состоянии системы. Недостаточная наблюдаемость среды, задержки поступления данных, ошибки измерений или неполнота информации приводят к снижению эффективности управления и могут вызывать формирование некорректных управляющих воздействий [10, 14].

В распределённых вычислительных системах данные становятся не вспомогательным элементом, а фундаментальным компонентом управляющего контура. Процессы мониторинга, анализа, прогнозирования, оптимизации и принятия решений образуют единую информационно-управляющую среду, в которой данные выступают основным ресурсом адаптивного управления.

В контексте ГРВС данные выполняют три ключевые функции: формирование наблюдаемого состояния системы, обеспечение обратной связи и поддержка прогнозирования.

Первая функция связана с наблюдением состояния вычислительной среды. Данные мониторинга используются для формирования оценки текущего состояния системы:

$$\hat{S}(t) = O(Y(t)).$$

где $Y(t)$ — поток наблюдаемых данных, O — функция наблюдения и обработки данных, а $\hat{S}(t)$ — оценка состояния системы.

Такая оценка может включать сведения о загрузке вычислительных узлов, использовании CPU, памяти и GPU, состоянии сетевых каналов, параметрах очередей задач, характеристиках потоков данных, состоянии

сервисов, событиях отказов и аномалиях. Без механизма наблюдения система управления фактически теряет возможность адаптации к изменяющемуся состоянию вычислительной среды.

Вторая функция данных связана с реализацией обратной связи. После выполнения управляющего воздействия система должна оценить результат изменения распределения ресурсов:

$$S(t + 1) = f(S(t), X(t)).$$

Данные обратной связи позволяют определить, привело ли управляющее воздействие к улучшению состояния системы, выполнены ли ограничения, уменьшилась ли нагрузка, соблюдены ли временные требования и не возникли ли новые перегрузки или конфликты. Тем самым данные обеспечивают замкнутость адаптивного контура управления и позволяют корректировать поведение системы на основе результатов предыдущих решений [15–17].

Третья функция данных связана с прогнозированием состояния вычислительной среды. Исторические и текущие данные используются для построения прогнозных моделей:

$$\hat{S}(t + 1) = M(S(t), H(t)),$$

где $H(t)$ — накопленная история наблюдений, а M — модель прогнозирования.

Прогнозирование позволяет заранее выявлять рост нагрузки, оценивать вероятность перегрузки ресурсов, прогнозировать нарушение SLA, выполнять предиктивное масштабирование и формировать проактивные управляющие воздействия. В современных распределённых вычислительных системах именно данные становятся основой интеллектуального управления, поскольку эффективность методов машинного обучения, прогнозирования и адаптивной оптимизации напрямую зависит от качества доступной информации.

Особенностью ГРВС является распределённый и неоднородный характер данных. Источники информации находятся на различных вычислительных узлах, имеют различную структуру, периодичность обновления и степень достоверности. Данные могут поступать асинхронно, содержать шум, пропуски или противоречивые значения [12, 15]. Поэтому для их использования в контуре управления необходимы механизмы распределённого мониторинга, агрегации, фильтрации, синхронизации, оценки достоверности и интеллектуального анализа потоков данных.

В мультиагентной архитектуре данные становятся связующим элементом между агентами системы управления. Агент мониторинга формирует наблюдаемое состояние среды, агенты прогнозирования используют исторические данные для построения моделей, агенты оптимизации принимают решения на основе агрегированных

характеристик системы, а агенты согласования используют информацию для разрешения конфликтов распределения ресурсов.

Таким образом, данные являются фундаментальной основой адаптивного управления гетерогенными распределёнными вычислительными системами. Именно качество наблюдаемой информации определяет эффективность прогнозирования, корректность принимаемых решений и способность системы адаптироваться к изменению состояния вычислительной среды.

6.2. Формализация процесса наблюдения

В реальных гетерогенных распределённых вычислительных системах состояние среды не может наблюдаться непосредственно и в полном объёме. Управляющая система получает лишь совокупность измеряемых параметров, отражающих отдельные аспекты функционирования вычислительной инфраструктуры: загрузку ресурсов, состояние сети, параметры очередей, события выполнения задач, логи и трассировки [10, 12, 15].

В связи с этим оценка состояния системы формируется не напрямую, а на основе обработки наблюдаемых данных:

$$\hat{S}(t) = O(Y(t)),$$

где $Y(t)$ — множество наблюдаемых данных; O — функция агрегации, фильтрации и обработки информации; $\hat{S}(t)$ — оценка состояния системы.

Наблюдаемые данные $Y(t)$ могут включать системные метрики, телеметрию вычислительных узлов, показатели загрузки CPU, памяти и GPU, сетевые характеристики, состояние очередей задач, журналы событий, данные трассировки, сообщения агентов мониторинга, события отказов и предупреждения. Однако даже при наличии развитой системы мониторинга эти данные не являются полным и точным отражением состояния ГРВС.

Принципиально важно, что функция O формирует не истинное состояние системы $S(t)$, а его приближённую оценку $\hat{S}(t)$. Это связано с ограниченной наблюдаемостью распределённой вычислительной среды, особенностями сбора телеметрии и распределённым характером источников данных.

В ГРВС наблюдение состояния обладает рядом фундаментальных особенностей.

Во-первых, наблюдение носит неполный характер. Ни один компонент системы управления не обладает полной и абсолютно актуальной информацией о состоянии всей вычислительной среды. Данные распределены между множеством узлов, собираются различными

средствами мониторинга и могут быть недоступны в отдельные моменты времени. Неполнота наблюдения может быть обусловлена ограничениями доступа к данным, потерей телеметрии, асинхронностью обновления информации, отказами компонентов мониторинга или высокой стоимостью полного сбора информации.

Формально это можно представить следующим образом:

$$Y(t) \subset S_{full}(t),$$

где $S_{full}(t)$ — полное состояние системы.

Во-вторых, наблюдение носит зашумлённый характер. Измеряемые данные могут содержать ошибки, случайные колебания и неточности:

$$Y(t) = Y_{true}(t) + \varepsilon(t),$$

где $\varepsilon(t)$ — шум измерений.

Источниками шума могут быть нестабильность измерений, ошибки телеметрии, сетевые задержки, кратковременные колебания нагрузки, неполная синхронизация источников данных и программные ошибки мониторинга. Шум особенно критичен для адаптивных систем управления, поскольку может приводить к ложным управляющим воздействиям, нестабильности распределения ресурсов и чрезмерной реакции системы на кратковременные флуктуации нагрузки [14–17].

В-третьих, наблюдение носит задержанный характер. Между фактическим изменением состояния среды и поступлением информации в управляющий контур существует временная задержка. В упрощённом виде это может быть записано как:

$$Y(t) = S(t - \Delta t),$$

где Δt — задержка наблюдения.

Причинами задержек являются время сбора телеметрии, сетевые задержки, периодичность обновления метрик, время обработки данных и распределённый характер вычислительной среды. В результате управляющая система принимает решения не на основе абсолютно актуального состояния среды, а на основе его запаздывающей оценки. Для высокодинамичных систем это может существенно снижать качество управления.

В связи с перечисленными особенностями функция наблюдения O должна включать механизмы фильтрации шумов, агрегации данных, синхронизации информации, оценки достоверности, восстановления пропущенных значений и, при необходимости, прогнозирования текущего состояния системы. В современных интеллектуальных системах управления для реализации этой функции используются статистические методы обработки данных, методы фильтрации, временные модели, корреляционный анализ, машинное обучение и нейросетевые модели обработки телеметрии.

Особую роль играет распределённый характер наблюдения. В мультиагентной системе различные агенты могут обладать собственными локальными представлениями состояния среды:

$$\hat{S}_i(t) = O_i(Y_i(t)).$$

Глобальная оценка состояния системы формируется на основе агрегации и согласования локальных наблюдений:

$$\hat{S}(t) = \bigcup_i \hat{S}_i(t).$$

При этом такое объединение не следует понимать как простое механическое сложение данных. Локальные оценки могут иметь различную степень актуальности, точности и достоверности, поэтому их объединение требует механизмов согласования, фильтрации противоречий и учёта доверия к источникам данных.

Таким образом, процесс наблюдения в ГРВС представляет собой распределённую процедуру формирования оценки состояния вычислительной среды на основе неполных, зашумлённых и задержанных данных. Качество данного процесса напрямую определяет эффективность адаптивного и интеллектуального управления распределённой вычислительной системой.

6.3. Метрики состояния системы

Формирование адекватного представления о состоянии гетерогенной распределённой вычислительной системы требует использования совокупности количественных характеристик, отражающих состояние вычислительных ресурсов, сетевой инфраструктуры, потоков задач и параметров функционирования системы в целом [10, 12, 15].

Совокупность наблюдаемых характеристик может быть представлена как множество метрик:

$$M = \{m_1, m_2, \dots, m_k\},$$

где каждая метрика m_i является функцией состояния системы:

$$m_i(t) = f_i(S(t)).$$

Метрики выполняют роль наблюдаемых признаков, на основе которых формируется оценка состояния вычислительной среды:

$$\hat{S}(t) = O(M(t)).$$

В распределённых системах метрики являются основным источником информации для мониторинга, адаптивного управления, прогнозирования нагрузки, обнаружения аномалий, оптимизации распределения ресурсов и оценки качества функционирования системы. При этом отдельная метрика редко даёт достаточное представление о состоянии ГРВС; управленчески значимой является совокупность взаимосвязанных показателей.

В зависимости от области применения метрики могут быть разделены на несколько классов: ресурсные, сетевые, метрики задач и системные метрики.

6.3.1. Ресурсные метрики

Ресурсные метрики характеризуют состояние вычислительных ресурсов отдельных узлов системы. Они используются агентами мониторинга, агентами вычислительных ресурсов и агентами оптимизации для оценки доступности и загрузки вычислительной инфраструктуры.

К числу основных ресурсных метрик относятся загрузка процессора,

$$CPU(t),$$

использование оперативной памяти,

$$MEM(t),$$

загрузка GPU, использование дисковой подсистемы, скорость операций ввода-вывода, энергопотребление вычислительных узлов и число активных вычислительных процессов.

Ресурсные метрики позволяют оценивать степень загруженности узлов, наличие свободных ресурсов, вероятность возникновения перегрузок и эффективность распределения вычислительной нагрузки. В гетерогенных системах особое значение имеет нормализация ресурсных метрик с учётом различий производительности вычислительных узлов [13, 16]. Например, одинаковая загрузка CPU на разных узлах может соответствовать различной фактической вычислительной мощности, что должно учитываться при принятии решений о размещении задач.

6.3.2. Сетевые метрики

Сетевые метрики характеризуют состояние коммуникационной инфраструктуры распределённой вычислительной системы. К основным сетевым метрикам относятся пропускная способность каналов связи,

$$B_{ij}(t),$$

задержка передачи данных,

$$L_{ij}(t),$$

уровень потерь пакетов, загрузка сетевых каналов, интенсивность потоков данных, доступность сетевых соединений и стабильность маршрутов передачи данных.

Сетевые метрики используются для адаптивной маршрутизации, балансировки потоков данных, предотвращения перегрузки сети, оценки времени передачи данных и выбора узлов размещения задач. В распределённых вычислительных средах сетевые ограничения часто оказывают существенное влияние на общую производительность системы, особенно при интенсивном межузловом обмене данными [10, 15].

Для задач управления ГРВС сетевые метрики должны анализироваться совместно с ресурсными метриками. Вычислительный узел может обладать достаточной производительностью, но быть

неэффективным для размещения задачи из-за высокой сетевой задержки, перегруженного канала или удалённости от данных.

6.3.3. Метрики задач

Метрики задач характеризуют процессы выполнения вычислительных задач и состояние очередей обработки. К ним относятся время выполнения задач,

$$t_i(t),$$

длина очереди,

$$Q(t),$$

время ожидания задачи, количество активных задач, число завершённых задач, интенсивность поступления новых задач, процент нарушения SLA и число повторных запусков задач.

Данные метрики позволяют оценивать эффективность планирования, качество балансировки нагрузки, состояние системы обслуживания задач, вероятность перегрузки и соблюдение временных ограничений. Особое значение метрики задач имеют для агентов времени и качества обслуживания, отвечающих за контроль SLA и приоритизацию вычислительной нагрузки [14–17].

В отличие от ресурсных и сетевых метрик, метрики задач отражают не только состояние инфраструктуры, но и результат взаимодействия нагрузки с системой управления. Например, рост очереди может быть следствием недостатка вычислительных ресурсов, сетевых задержек, неудачного размещения задач или некорректных приоритетов обработки.

6.3.4. Системные метрики

Системные метрики характеризуют функционирование распределённой вычислительной среды в целом и отражают интегральные показатели эффективности управления. К ним относятся отказоустойчивость, доступность, коэффициент использования ресурсов, уровень деградации производительности, число отказов компонентов, среднее время восстановления, соблюдение SLA, устойчивость распределения ресурсов и уровень согласованности системы.

Системные метрики используются агентами координации, агентами оптимизации, средствами анализа эффективности и механизмами стратегического управления. В адаптивных системах управления именно системные метрики часто используются как компоненты глобальной целевой функции:

$$F(X, S(t)).$$

Следует отметить, что в ГРВС отдельные метрики могут быть взаимосвязаны. Рост загрузки CPU может приводить к увеличению длины очередей, перегрузка сети — к увеличению времени выполнения задач, а нарушение SLA может быть связано как с вычислительными, так и с коммуникационными ограничениями. Поэтому эффективное управление требует не анализа отдельных метрик изолированно, а формирования

интегральной модели состояния системы на основе совокупности наблюдаемых параметров [15–17].

В современных интеллектуальных системах управления для анализа метрик используются методы корреляционного анализа, многомерная статистика, машинное обучение, нейросетевые модели, методы обнаружения аномалий и потоковая аналитика телеметрии. Эти методы позволяют выявлять скрытые зависимости между параметрами состояния и повышать качество оценки вычислительной среды.

Таким образом, система метрик является основой наблюдаемости ГРВС и обеспечивает количественное описание состояния вычислительной среды. Именно на основе метрик формируются процессы мониторинга, адаптивного управления, прогнозирования и интеллектуального анализа состояния распределённой вычислительной системы.

6.4. Наблюдаемость (Observability)

Наблюдаемость является одним из ключевых свойств современных распределённых вычислительных систем и определяет возможность восстановления и интерпретации внутреннего состояния системы на основе внешних данных, получаемых в процессе её функционирования [10, 12, 15]. В отличие от мониторинга, ориентированного преимущественно на сбор отдельных показателей, наблюдаемость предполагает способность связать метрики, события и трассировки с причинами изменения состояния вычислительной среды.

Формально наблюдаемость может быть представлена следующим образом:

$$\hat{S}(t) = O(Logs, Metrics, Traces).$$

где $\hat{S}(t)$ — состояние системы, а O — функция восстановления и интерпретации состояния на основе логов, метрик и трассировок.

Данная запись подчёркивает, что состояние системы в реальных условиях восстанавливается приближённо. Управляющая подсистема не имеет полного доступа ко всем внутренним процессам ГРВС, поэтому использует внешние наблюдаемые данные для построения оценки состояния:

$$\hat{S}(t) = O(Logs, Metrics, Traces).$$

Классическая триада наблюдаемости включает метрики, логи и трассировки. Метрики представляют собой числовые показатели, характеризующие состояние отдельных компонентов системы или системы в целом:

$$m_i(t).$$

К ним относятся загрузка процессора, использование памяти, пропускная способность каналов связи, задержки передачи данных, длина очередей, время выполнения задач, число ошибок и другие количественные характеристики. Метрики удобны для агрегированного анализа состояния системы, выявления трендов и построения прогнозных моделей [14–17].

Логи представляют собой последовательности событий, фиксируемых различными компонентами системы:

$$Log = \{e_1, e_2, \dots, e_n\}.$$

События могут отражать запуск и завершение задач, ошибки выполнения, изменения конфигурации, отказы компонентов, предупреждения безопасности, изменения состояния сервисов и управляющие действия агентов. Логи позволяют восстанавливать историю функционирования системы и анализировать причинно-следственные связи между изменениями состояния и управляющими воздействиями.

Трассировки описывают прохождение запроса, задачи или потока данных через последовательность компонентов распределённой системы:

$$Trace = (s_1 \rightarrow s_2 \rightarrow \dots \rightarrow s_k).$$

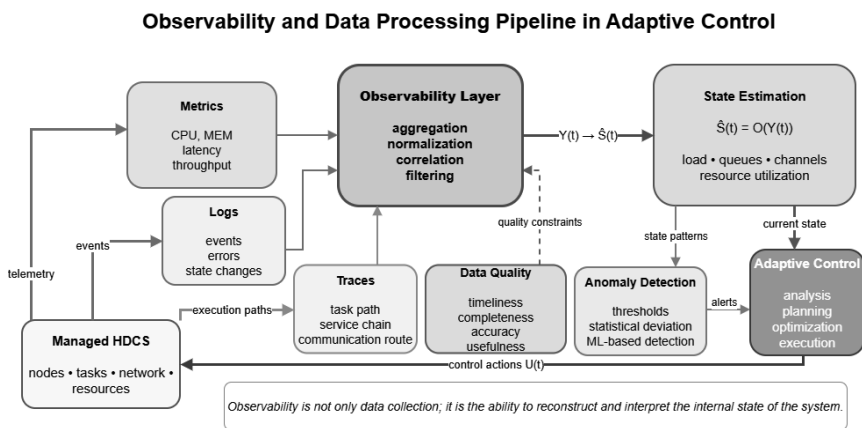
Трассировки особенно важны в сложных распределённых средах, где выполнение одной задачи может включать взаимодействие множества сервисов, узлов и сетевых маршрутов. Они позволяют выявлять узкие места, задержки, ошибки маршрутизации и проблемы взаимодействия между компонентами.

Принципиально важно, что наблюдаемость не сводится к накоплению большого объёма данных. Система может собирать множество метрик и логов, но оставаться плохо наблюдаемой, если эти данные не позволяют объяснить причины изменения состояния и сформировать корректное управляющее воздействие. Поэтому наблюдаемость должна включать не только сбор данных, но и их нормализацию, корреляцию, агрегацию, фильтрацию и интерпретацию.

В контексте адаптивного управления наблюдаемость выполняет несколько функций: формирует оценку текущего состояния системы, обеспечивает обнаружение отклонений и аномалий, позволяет анализировать последствия управляющих воздействий, поддерживает прогнозирование будущего состояния и обеспечивает обратную связь в контуре управления.

Для ГРВС наблюдаемость особенно важна вследствие распределённости, гетерогенности и асинхронности процессов. Данные о состоянии системы поступают из различных источников, имеют различную частоту обновления, могут содержать шум, задержки и противоречивые значения. Поэтому построение наблюдаемости требует согласованной обработки разнородных источников информации и оценки качества данных [15–17].

Рисунок 7 иллюстрирует pipeline наблюдаемости и обработки данных в контуре адаптивного управления. Управляемая ГРВС формирует телеметрию, события и трассировки, которые преобразуются в метрики, логи и описания путей выполнения задач. Далее эти данные поступают в слой наблюдаемости, где выполняются агрегация, нормализация, корреляция и фильтрация. На основе обработанных данных формируется оценка состояния $\hat{S}(t)$, используемая контуром адаптивного управления. Дополнительно в схеме отражены контроль качества данных и выявление аномалий, влияющие на корректность принимаемых управляющих решений.



Observability pipeline: metrics, logs, and traces are transformed into state estimates used by adaptive control

Рисунок 7 – Pipeline наблюдаемости и обработки данных

Таким образом, наблюдаемость является необходимым условием эффективного адаптивного управления гетерогенными распределёнными вычислительными системами. Она обеспечивает переход от разрозненных данных мониторинга к целостному представлению состояния системы, необходимому для анализа, прогнозирования, оптимизации и принятия управляющих решений.

6.5. Обработка потоков данных

В гетерогенных распределённых вычислительных системах данные мониторинга поступают не как статический набор наблюдений, а как непрерывный поток сообщений, метрик, логов, событий и трассировок. Поэтому обработка данных в контуре адаптивного управления должна учитывать временную природу наблюдений, задержки поступления

информации и необходимость оперативного формирования оценки состояния системы [10, 12, 15].

Поток наблюдаемых данных может быть представлен следующим образом:

$$Y(t) = \{y_1(t), y_2(t), \dots, y_n(t)\},$$

где $y_i(t)$ — отдельное наблюдение, поступившее от компонента системы в момент времени t .

К таким наблюдениям могут относиться значения метрик, записи логов, события изменения состояния, сообщения агентов мониторинга, данные трассировки и результаты локального анализа. В отличие от пакетной обработки, потоковая обработка ориентирована на непрерывное обновление состояния системы и быстрое выявление изменений, требующих управляющей реакции.

6.5.1. Потоковая обработка

Потоковая обработка предполагает формирование оценки состояния системы на основе данных, поступивших за некоторый временной интервал:

$$\hat{S}(t) = f(Y(t - \Delta t, t)),$$

где $Y(t - \Delta t, t)$ — поток наблюдений, полученных на интервале времени от $t - \Delta t$ до t , а f — функция обработки и агрегации данных.

Использование временного окна Δt позволяет сгладить случайные колебания, уменьшить влияние шума и сформировать более устойчивую оценку состояния системы. При этом выбор размера окна имеет принципиальное значение. Слишком короткое окно повышает чувствительность системы к флуктуациям, а слишком длинное может привести к запаздыванию реакции управляющего контура.

Потоковая обработка используется для оценки текущей загрузки ресурсов, выявления роста очередей задач, обнаружения сетевых перегрузок, анализа изменений задержек передачи данных, выявления аномалий, формирования признаков для прогнозных моделей и обновления состояния агентов управления.

В адаптивной системе управления потоковая обработка обеспечивает непрерывное обновление оценки состояния $\hat{S}(t)$, используемой агентами мониторинга, прогнозирования, оптимизации и согласования. Благодаря этому управляющий контур получает не разовые измерения, а динамическое представление о состоянии вычислительной среды.

Следует отметить, что потоковая обработка должна учитывать не только значения отдельных наблюдений, но и их временной контекст. Например, кратковременный всплеск загрузки CPU может не требовать перераспределения задач, тогда как устойчивый рост очереди или

задержек передачи данных может свидетельствовать о формировании перегрузки и необходимости адаптивного управляющего воздействия.

6.5.2. Event-driven модель

Наряду с потоковой обработкой метрик в распределённых вычислительных системах широко используется событийная модель управления. В такой модели система реагирует не только на агрегированные показатели состояния, но и на отдельные события, имеющие управленческое значение [14–17].

Событие может быть представлено следующим образом:

$$e = (type, timestamp, payload),$$

где *type* — тип события, *timestamp* — момент его возникновения, а *payload* — содержательная часть события, включающая параметры, необходимые для анализа и принятия решений.

К событиям, значимым для управления ГРВС, относятся превышение порога загрузки узла, отказ вычислительного компонента, деградация сетевого канала, рост очереди задач, нарушение SLA, завершение критической задачи, изменение доступности ресурса и обнаружение аномального поведения.

В событийной модели управляющее воздействие может формироваться непосредственно как реакция на событие:

$$X(t) = f(e).$$

Такой подход позволяет быстро реагировать на критические изменения состояния системы без необходимости ожидать завершения полного цикла агрегации данных. Например, событие отказа узла может немедленно инициировать миграцию задач, а событие превышения порога задержки — изменение маршрута передачи данных.

Однако event-driven модель должна использоваться с учётом риска ложных срабатываний. Отдельное событие может быть результатом кратковременной флуктуации, ошибки измерения или задержки мониторинга. Поэтому в практических системах событийная обработка обычно сочетается с потоковой агрегацией, фильтрацией и анализом контекста.

В мультиагентной архитектуре события могут инициировать взаимодействие между агентами. Например, агент мониторинга фиксирует событие перегрузки, агент вычислительных ресурсов оценивает локальные возможности перераспределения, сетевой агент проверяет доступность маршрутов, а агент согласования формирует совместимое управляющее решение.

Таким образом, обработка потоков данных в ГРВС должна сочетать потоковую модель анализа метрик и событийную модель реакции на значимые изменения состояния системы. Потоковая обработка

обеспечивает устойчивую оценку состояния среды, а событийная модель — оперативную реакцию на критические изменения. Их совместное использование является необходимым условием эффективного адаптивного управления распределённой вычислительной системой.

6.6. Оценка состояния системы

В распределённых вычислительных системах наблюдаемые данные не являются непосредственным представлением внутреннего состояния среды. Метрики, логи, события и трассировки содержат шум, могут поступать с задержками и отражают только отдельные аспекты функционирования системы. Поэтому в контуре адаптивного управления необходим специальный этап оценки состояния, преобразующий поток наблюдений в согласованное представление о состоянии вычислительной среды [10, 12, 15].

Формально оценка состояния системы может быть представлена следующим образом:

$$\hat{S}(t) = O(Y(t)),$$

где $Y(t)$ — множество наблюдаемых данных; O — алгоритм оценки состояния; $\hat{S}(t)$ — оценка текущего состояния системы.

В отличие от истинного состояния $S(t)$, которое в большинстве случаев недоступно напрямую, оценка $\hat{S}(t)$ представляет собой приближённое описание вычислительной среды, формируемое на основе обработки наблюдений. Основная задача алгоритма O заключается в том, чтобы преобразовать разнородные, неполные и зашумлённые данные в такое представление состояния, которое может быть использовано агентами прогнозирования, оптимизации, согласования и исполнения управляющих воздействий.

Качество оценки состояния напрямую влияет на эффективность адаптивного управления. Ошибочная оценка может приводить к некорректному распределению ресурсов, ложным реакциям на кратковременные флуктуации, перегрузке отдельных компонентов системы, нарушению SLA и нестабильности управляющего контура [14–17].

В современных системах управления используются различные методы оценки состояния: фильтрация, агрегация, статистический анализ, вероятностное моделирование и методы машинного обучения.

Фильтрация предназначена для подавления шумов и случайных колебаний наблюдаемых данных:

$$\hat{y}(t) = \text{Filter}(Y(t)).$$

В распределённых вычислительных системах шум может быть связан с нестабильностью измерений, асинхронностью обновления метрик, сетевыми задержками, кратковременными всплесками нагрузки и ошибками мониторинга. Фильтрация позволяет уменьшить влияние случайных флуктуаций и сформировать более устойчивое представление состояния системы. Для этого могут использоваться скользящие средние, экспоненциальное сглаживание, медианная фильтрация, фильтры Калмана и адаптивные фильтры.

Особенно важна фильтрация для event-driven систем, где ложные всплески метрик могут инициировать ненужные управляющие воздействия. В таких условиях фильтрация выполняет не только техническую, но и управленческую функцию: она помогает отделить устойчивое изменение состояния от случайного отклонения.

В ГРВС данные поступают от множества распределённых источников, поэтому для формирования интегральной оценки состояния необходима агрегация информации:

$$\hat{S}(t) = \text{Aggregate}(\hat{S}_1(t), \hat{S}_2(t), \dots, \hat{S}_n(t)).$$

Агрегация может выполняться по времени, по узлам системы, по группам сервисов, по типам ресурсов или по уровням иерархии управления. Например, локальные оценки загрузки отдельных узлов могут объединяться в оценку состояния вычислительного кластера, а затем — всей распределённой вычислительной среды. Это позволяет уменьшить объём обрабатываемых данных, повысить устойчивость оценки, выявлять глобальные тенденции изменения состояния системы и поддерживать иерархическое управление.

Статистические методы используются для анализа распределений наблюдаемых параметров, оценки вероятностей событий и выявления аномалий:

$$P(S(t) | Y(t)).$$

К таким методам относятся корреляционный и регрессионный анализ, оценка распределений, модели временных рядов, кластеризация, байесовские модели и вероятностная фильтрация. Они позволяют выявлять закономерности изменения нагрузки, оценивать вероятность перегрузок, обнаруживать аномальное поведение и прогнозировать изменение состояния системы. В условиях неполной наблюдаемости вероятностные методы особенно важны, поскольку позволяют формировать оценку состояния даже при наличии неполных или противоречивых данных [13, 16].

В современных распределённых вычислительных системах всё более широкое применение получают методы машинного обучения, позволяющие строить интеллектуальные модели оценки состояния среды:

$$\hat{S}(t) = ML(Y(t)).$$

ML-модели способны выявлять сложные нелинейные зависимости между параметрами системы и учитывать скрытые закономерности, плохо описываемые аналитическими моделями. Для оценки состояния могут использоваться регрессионные модели, нейронные сети, autoencoder-модели, модели временных рядов, LSTM, Transformer, методы обнаружения аномалий и графовые нейросетевые модели.

Преимуществом ML-подходов является способность адаптироваться к изменению структуры нагрузки и особенностей функционирования системы. Это особенно важно для гетерогенных распределённых сред, в которых параметры состояния изменяются динамически и могут зависеть от большого числа факторов [15–17]. Вместе с тем применение ML требует достаточного объёма данных, корректной настройки моделей, контроля устойчивости обучения, оценки интерпретируемости результатов и защиты от деградации моделей при изменении среды.

В мультиагентной системе оценка состояния может выполняться как централизованно, так и распределённо. Локальные агенты формируют частные оценки:

$$\hat{S}_i(t) = E_i(Y_i(t)),$$

после чего глобальная оценка системы строится на основе согласования локальных представлений состояния среды. Такой подход позволяет учитывать распределённую природу данных и снижать нагрузку на центральные компоненты управления, однако требует механизмов согласования локальных оценок, проверки достоверности источников и разрешения противоречий между наблюдениями.

Таким образом, оценка состояния является ключевым элементом контура адаптивного управления ГРВС. Она обеспечивает преобразование потока наблюдаемых данных в целостное представление состояния вычислительной среды и создаёт основу для прогнозирования, оптимизации и принятия интеллектуальных управляющих решений.

6.7. Выявление аномалий

Выявление аномалий является важным элементом наблюдаемости и адаптивного управления гетерогенными распределёнными вычислительными системами. Под аномалией будем понимать такое состояние системы, отдельной метрики или их совокупности, которое существенно отклоняется от ожидаемого режима функционирования и может указывать на перегрузку, отказ, деградацию производительности, нарушение SLA или некорректное поведение компонента [10, 12, 15].

В простейшем случае аномалия может быть определена как выход наблюдаемой метрики за допустимый интервал:

$$m_i(t) \notin [m_i^{min}, m_i^{max}],$$

где $m_i(t)$ — значение метрики в момент времени t , а m_i^{min} и m_i^{max} — нижняя и верхняя границы допустимого диапазона.

Такой подход удобен для контроля ресурсных и сетевых параметров: загрузки CPU, использования памяти, задержки передачи данных, длины очереди задач или числа ошибок. Однако в сложных распределённых системах пороговые методы не всегда достаточны, поскольку аномальное состояние может проявляться не в одной метрике, а в необычной комбинации нескольких параметров. Например, умеренная загрузка CPU сама по себе может не быть проблемой, но в сочетании с ростом сетевых задержек, увеличением очереди и снижением пропускной способности может указывать на формирование системной перегрузки.

Более общий подход основан на вероятностной оценке состояния системы:

$$P(S(t)) < \varepsilon,$$

где $P(S(t))$ — вероятность наблюдаемого состояния системы, а ε — заданный порог аномальности.

Если вероятность текущего состояния оказывается ниже допустимого порога, состояние рассматривается как нетипичное и требует дополнительного анализа или управляющей реакции. Данный подход позволяет выявлять сложные аномалии, связанные с редкими сочетаниями значений метрик, нестандартной динамикой нагрузки или необычными паттернами взаимодействия компонентов системы [13, 16].

В контуре адаптивного управления выявление аномалий выполняет две основные функции. Первая связана с запуском адаптации. Обнаружение аномального состояния может служить основанием для изменения режима управления, перераспределения задач, масштабирования ресурсов, изменения маршрутов передачи данных или корректировки политик обработки очередей. В этом случае аномалия выступает как событие, инициирующее управляющее воздействие:

$$e_{anom} \rightarrow U(t).$$

Вторая функция связана с предотвращением сбоев. Своевременное выявление аномалий позволяет обнаруживать признаки деградации системы до наступления критического отказа. Например, постепенный рост задержек, увеличение числа повторных попыток передачи данных или нестандартное изменение нагрузки могут указывать на приближающуюся перегрузку, деградацию сетевого канала или отказ вычислительного компонента.

Методы выявления аномалий могут быть разделены на несколько классов: пороговые методы, статистические методы, анализ временных рядов, кластеризация, методы машинного обучения, нейросетевые модели обнаружения аномалий и графовые методы анализа поведения распределённой системы.

Пороговые методы просты и хорошо интерпретируемы, однако требуют ручного задания допустимых диапазонов и плохо работают при изменяющейся норме поведения системы. Статистические методы позволяют учитывать распределения метрик и выявлять маловероятные состояния. Методы машинного обучения и нейросетевые модели способны обнаруживать скрытые закономерности и сложные многомерные аномалии, но требуют достаточного объема данных, контроля качества моделей и проверки устойчивости результатов [15–17].

В мультиагентной архитектуре обнаружение аномалий может выполняться распределённо. Локальные агенты мониторинга выявляют отклонения в своих областях ответственности, после чего передают информацию агентам согласования, прогнозирования или оптимизации. Это позволяет быстро реагировать на локальные нарушения и одновременно оценивать их влияние на состояние системы в целом.

При этом важно различать локальные и системные аномалии. Локальная аномалия может быть связана с отдельным узлом, каналом связи или очередью задач и устраняться локальным управляющим воздействием. Системная аномалия затрагивает несколько подсистем одновременно и требует координации решений между агентами вычислительных ресурсов, сети, качества обслуживания и оптимизации.

Таким образом, выявление аномалий является не только средством диагностики, но и механизмом запуска адаптивных управляющих воздействий. Оно обеспечивает раннее обнаружение перегрузок, отказов и нарушений качества обслуживания, повышая устойчивость и надёжность функционирования гетерогенной распределённой вычислительной системы.

6.8. Замыкание контура управления

Рассмотренные ранее процессы наблюдения, обработки данных, оценки состояния системы и формирования управляющих воздействий образуют единый замкнутый контур адаптивного управления гетерогенной распределённой вычислительной системой [12–15].

В рамках данного контура состояние вычислительной среды непрерывно наблюдается, анализируется и используется для формирования новых управляющих решений, которые, в свою очередь, изменяют состояние системы. Следовательно, управление ГРВС представляет собой циклический процесс взаимодействия управляемой среды и управляющей подсистемы.

Общая схема замыкания контура управления может быть представлена следующим образом:

$$S(t) \rightarrow Y(t) \rightarrow \hat{S}(t) \rightarrow X(t) \rightarrow S(t + 1),$$

где $S(t)$ — текущее состояние системы; $Y(t)$ — поток наблюдаемых данных; $\hat{S}(t)$ — оценка состояния системы; $X(t)$ — управляющее воздействие; $S(t+1)$ — новое состояние системы после применения управления.

На первом этапе текущее состояние вычислительной среды $S(t)$ формирует поток наблюдаемых данных $Y(t)$. Данные поступают от множества распределённых источников и включают метрики ресурсов, сетевые показатели, логи событий, трассировки, телеметрию вычислительных процессов и сообщения агентов мониторинга.

Поскольку наблюдение распределённой системы является неполным, зашумлённым и задержанным, наблюдаемые данные не совпадают с истинным состоянием среды. Поэтому следующим этапом является оценка состояния системы:

$$\hat{S}(t) = E(Y(t)),$$

где E — алгоритм обработки, агрегации и интерпретации данных. На данном этапе выполняются фильтрация шумов, агрегация распределённых наблюдений, восстановление пропущенной информации, анализ аномалий, прогнозирование состояния среды и формирование признакового описания системы.

Полученная оценка состояния $\hat{S}(t)$ используется алгоритмами адаптивного управления для формирования управляющего воздействия:

$$X(t) = A(\hat{S}(t)).$$

Управляющее воздействие может включать перераспределение задач, изменение маршрутов передачи данных, балансировку нагрузки, масштабирование вычислительных ресурсов, изменение параметров очередей, миграцию сервисов или запуск механизмов восстановления.

После применения управляющего воздействия система переходит в новое состояние:

$$S(t+1) = f(S(t), X(t)),$$

которое вновь становится объектом наблюдения и анализа. Тем самым контур приобретает замкнутый характер, обеспечивая непрерывную адаптацию вычислительной среды к изменению нагрузки, состояния ресурсов и внешних условий.

Принципиально важным свойством замкнутого контура является наличие обратной связи. Именно обратная связь позволяет оценивать эффективность принятых решений, корректировать параметры управления, предотвращать накопление ошибок, поддерживать устойчивость системы и адаптироваться к изменению среды.

В условиях ГРВС замыкание контура управления осложняется распределённым характером наблюдений, задержками передачи данных, асинхронностью обновления информации, высокой размерностью

состояния системы и динамическим изменением структуры вычислительной среды. Поэтому в современных системах управления используются распределённые механизмы наблюдения, потоковая обработка данных, мультиагентная координация, прогнозные модели и интеллектуальные методы анализа состояния среды [14–17].

В мультиагентной архитектуре отдельные этапы контура управления распределяются между специализированными агентами. Агенты мониторинга формируют поток данных $Y(t)$, агенты анализа и прогнозирования строят оценку $S(t)$, агенты оптимизации и координации вычисляют $X(t)$, а агенты вычислительных ресурсов и сети реализуют управляющие воздействия. Такое распределение функций позволяет обеспечить масштабируемость и устойчивость контура адаптивного управления.

Следует отметить, что эффективность замкнутого контура определяется не только качеством отдельных алгоритмов, но и согласованностью всех этапов обработки информации. Ошибки наблюдения, неточности оценки состояния, задержки передачи данных или несогласованность действий агентов могут приводить к формированию неэффективных или нестабильных управляющих воздействий.

Поэтому современные интеллектуальные системы управления всё чаще рассматриваются как системы, управляемые данными (data-driven systems), в которых качество обработки информации становится одним из ключевых факторов эффективности управления распределённой вычислительной средой.

Таким образом, замыкание контура управления объединяет процессы наблюдения, анализа, оценки состояния и адаптивного распределения ресурсов в единую интеллектуальную систему управления гетерогенной распределённой вычислительной средой.

6.9. Задержки и их влияние

Одной из фундаментальных особенностей гетерогенных распределённых вычислительных систем является наличие задержек в процессах наблюдения, передачи данных, обработки информации и формирования управляющих воздействий. В результате система управления практически никогда не располагает абсолютно актуальной информацией о состоянии вычислительной среды [10, 12, 15].

В реальных условиях оценка состояния системы формируется с некоторым временным запаздыванием:

$$\hat{S}(t) = S(t - \Delta t),$$

где $S(t)$ — фактическое состояние системы в момент времени t , $\hat{S}(t)$ — доступная системе управления оценка состояния, а Δt — совокупная задержка наблюдения и обработки данных.

Данная задержка включает время сбора телеметрии, передачу данных по сети, периодичность обновления метрик, агрегацию и фильтрацию информации, обработку логов и трассировок, а также время работы алгоритмов анализа и оценки состояния. В результате управляющая система принимает решения на основе информации, которая уже частично устарела к моменту формирования управляющего воздействия.

Для высокодинамичных распределённых систем это имеет принципиальное значение. Если состояние среды изменяется быстрее, чем обновляется информация о системе, управляющий контур начинает реагировать на уже неактуальные состояния вычислительной инфраструктуры. Следствием этого могут быть снижение эффективности распределения ресурсов, запаздывающая реакция на перегрузки, рост вероятности нарушения SLA, нестабильность балансировки нагрузки, циклические колебания распределения ресурсов и возникновение ложных управляющих воздействий.

Например, к моменту обнаружения перегрузки вычислительного узла нагрузка может уже перераспределиться естественным образом, а дополнительное управляющее воздействие приведёт к новой перегрузке других компонентов системы. Подобные эффекты особенно характерны для систем с высокой интенсивностью изменения нагрузки и распределённым характером наблюдения [14–17].

Влияние задержек может быть представлено следующим образом:

$$X(t) = A(\hat{S}(t)) = A(S(t - \Delta t)).$$

Таким образом, управление формируется не на основе актуального состояния среды, а на основе его запаздывающей оценки. При увеличении Δt эффективность адаптивного управления снижается, а в предельном случае задержка может приводить к потере устойчивости управляющего контура:

$$\Delta t \rightarrow \text{instability}.$$

Для компенсации данного эффекта в современных системах управления используются прогнозные модели, позволяющие оценивать текущее и будущее состояние среды на основе накопленных наблюдений:

$$\hat{S}(t + 1) = M(S(t), H(t)),$$

где M — модель прогнозирования, а $H(t)$ — история наблюдений.

Использование прогноза позволяет перейти от реактивного управления к проактивному:

$$X(t) = A(\hat{S}(t + 1)).$$

В этом случае система принимает решения не только на основе наблюдаемого прошлого состояния среды, но и с учётом ожидаемой динамики изменения нагрузки, состояния сети и доступности ресурсов. Прогнозирование особенно важно для систем реального времени, облачных и edge-платформ, высоконагруженных сервисов, систем с интенсивным межузловым обменом данными и мультиагентных архитектур управления.

В качестве прогнозных моделей могут использоваться статистические методы, временные ряды, регрессионные модели, нейросетевые архитектуры, LSTM, Transformer, reinforcement learning и гибридные модели адаптивного прогнозирования [15–17]. Их задача заключается не только в предсказании будущего состояния, но и в компенсации запаздывания, возникающего в контуре наблюдения и принятия решений.

Следует отметить, что задержки возникают не только в наблюдении, но и в реализации управляющих воздействий. После формирования решения требуется время на миграцию задач, изменение маршрутов передачи данных, масштабирование ресурсов, обновление конфигурации системы и синхронизацию распределённых компонентов. Поэтому эффективное управление ГРВС требует учёта задержек на всех этапах контура:

Observe → Analyze → Decide → Execute.

В мультиагентных системах проблема задержек дополнительно осложняется необходимостью координации решений между агентами. Различные агенты могут обладать различной степенью актуальности локальной информации, что приводит к необходимости согласования распределённых представлений состояния среды. Поэтому в таких системах важны не только прогнозные модели, но и механизмы синхронизации, оценки доверия к данным и ограничения частоты управляющих воздействий.

Таким образом, задержки являются фундаментальным фактором, влияющим на устойчивость и эффективность адаптивного управления распределёнными вычислительными системами. Их наличие делает необходимым использование прогнозных моделей, механизмов компенсации запаздывания и интеллектуальных методов анализа динамики состояния вычислительной среды.

6.10. Неопределённость данных

Неопределённость данных является неотъемлемым свойством гетерогенных распределённых вычислительных систем и непосредственно влияет на качество адаптивного управления. В реальных условиях управляющая подсистема оперирует не истинным состоянием системы

$S(t)$, а его приближённой оценкой $\hat{S}(t)$, сформированной на основе наблюдаемых данных, метрик, логов, событий и трассировок [10, 12, 15].

Формально связь между истинным состоянием системы и его оценкой может быть представлена следующим образом:

$$\hat{S}(t) = S(t) + \varepsilon(t)$$

оценка состояния отличается от истинного состояния на ошибку $\varepsilon(t)$.

Ошибка ε отражает расхождение между фактическим состоянием распределённой вычислительной среды и тем состоянием, которое доступно системе управления. Чем больше величина ошибки, тем выше риск формирования некорректных управляющих воздействий.

Основными источниками неопределённости данных являются шум измерений, задержки поступления информации, потери данных и противоречивость наблюдений. Шум возникает вследствие нестабильности измерительных механизмов, кратковременных флуктуаций нагрузки, ошибок телеметрии и асинхронности источников данных. Задержки приводят к тому, что система управления использует устаревшее представление о состоянии среды. Потери данных возникают при сбоях мониторинга, перегрузке каналов связи, отказах компонентов или неполной доступности отдельных узлов. Противоречивость наблюдений может быть связана с тем, что разные источники фиксируют состояние системы с различной частотой, точностью и задержкой.

Неопределённость данных может проявляться в виде неточных значений метрик, пропусков наблюдений, противоречивых данных от разных источников, устаревшей информации о состоянии ресурсов, неполных сведений о сетевых потоках, ложных событий и некорректно выявленных аномалий.

В условиях неопределённости управляющее воздействие фактически формируется на основе приближённой модели состояния:

$$X(t) = A(\hat{S}(t)),$$

тогда как его последствия определяются истинным состоянием системы:

$$S(t+1) = f(S(t), X(t)).$$

Это означает, что даже корректный с точки зрения наблюдаемой информации алгоритм управления может приводить к неэффективному результату, если оценка состояния существенно отличается от реального состояния вычислительной среды. Например, система может перераспределить задачи с узла, который по данным мониторинга выглядит перегруженным, хотя фактически перегрузка уже исчезла; либо, наоборот, не обнаружить формирующуюся перегрузку из-за запаздывания или потери телеметрии.

Для снижения влияния неопределённости используются методы фильтрации, агрегации, статистической оценки, восстановления

пропущенных данных и вероятностного моделирования. Эти методы позволяют уменьшить влияние шума, сгладить случайные флуктуации, сопоставить данные из различных источников и сформировать более устойчивую оценку состояния. В интеллектуальных системах управления дополнительно применяются методы машинного обучения, способные выявлять устойчивые закономерности в зашумлённых данных и формировать более надёжные оценки состояния [15–17].

В мультиагентной архитектуре неопределённость данных имеет распределённый характер. Каждый агент обладает собственной локальной оценкой состояния:

$$\hat{S}_i(t) = E_i(Y_i(t)),$$

которая может отличаться от оценок других агентов по актуальности, полноте и достоверности. Поэтому важную роль приобретают механизмы согласования локальных представлений состояния, обмена информацией, оценки доверия к источникам данных и разрешения противоречий между локальными наблюдениями.

Таким образом, неопределённость данных является одним из ключевых факторов, определяющих сложность адаптивного управления ГРВС. Эффективная система управления должна учитывать ошибки наблюдения, задержки, шум, потери и противоречивость данных, а также использовать методы компенсации неопределённости при формировании управляющих воздействий.

6.11. Связь наблюдаемости с мультиагентной системой

Рассмотренные процессы наблюдения, обработки данных, оценки состояния системы и выявления аномалий непосредственно интегрируются в мультиагентную архитектуру управления гетерогенной распределённой вычислительной системой [14–17]. В мультиагентной модели функции обработки информации распределяются между специализированными агентами, каждый из которых отвечает за отдельный этап контура адаптивного управления.

Такой подход позволяет обеспечить масштабируемость обработки данных, распределённый анализ состояния среды, локальную адаптацию, параллельное принятие решений и устойчивость к отказам отдельных компонентов системы. В отличие от централизованной схемы, где вся телеметрия должна быть передана в единый управляющий компонент, мультиагентная архитектура позволяет выполнять первичную обработку и анализ данных ближе к источникам их возникновения.

В мультиагентной архитектуре процессы наблюдаемости могут быть распределены между различными классами агентов.

Таблица 6.1 — Распределение функций наблюдаемости между агентами

Компонент контура управления	Агент
Сбор и первичная обработка данных	Monitoring Agent
Анализ состояния и прогнозирование	Prediction Agent
Принятие и согласование решений	Optimization Agent / Coordination Agent
Исполнение управляющих воздействий	Compute Agent / Network Agent

Агент мониторинга отвечает за сбор, агрегацию и предварительную обработку данных наблюдения:

$$Y(t) = O(S(t)).$$

Он получает системные метрики, телеметрию ресурсов, сетевые показатели, события, логи, трассировки и информацию о состоянии вычислительных узлов. На этом уровне могут выполняться фильтрация шумов, нормализация данных, синхронизация наблюдений и базовое обнаружение аномалий. В результате формируются локальные представления состояния системы, которые передаются другим агентам управляющего контура.

Агент прогнозирования выполняет анализ наблюдаемых данных и формирует оценку будущего состояния среды:

$$\hat{S}(t + 1) = M(S(t)).$$

Его функции связаны с прогнозированием нагрузки, анализом временных рядов, выявлением скрытых закономерностей, оценкой вероятности перегрузок, обнаружением аномалий и оценкой риска нарушения SLA. Для этого могут применяться статистические методы, ML-модели, нейросетевые архитектуры, временные модели и методы анализа потоковых данных. Результаты прогнозирования передаются агентам оптимизации и согласования для формирования управляющих воздействий [15–17].

Агент оптимизации и агент согласования используют наблюдаемое и прогнозируемое состояние системы для принятия решений:

$$X^*(t) = A(\hat{S}(t)).$$

На этом этапе выполняются распределение задач, балансировка нагрузки, выбор маршрутов передачи данных, многокритериальная оптимизация, согласование локальных решений и формирование глобального управляющего воздействия. В зависимости от сложности задачи могут использоваться эвристические методы, локальный поиск, генетические алгоритмы, многокритериальная оптимизация и гибридные интеллектуальные методы.

Агенты исполнения, в частности Compute Agent и Network Agent, реализуют сформированные управляющие воздействия:

$$S(t + 1) = f(S(t), X(t)).$$

Compute Agent отвечает за перераспределение вычислительной нагрузки, управление вычислительными ресурсами, локальную балансировку, миграцию задач и масштабирование вычислений. Network Agent управляет маршрутизацией потоков данных, балансировкой сетевой нагрузки, изменением сетевых маршрутов и адаптацией коммуникационной инфраструктуры.

Принципиально важным является то, что взаимодействие между агентами формирует распределённый контур наблюдаемости и управления. Данные, полученные агентами мониторинга, становятся основой прогнозирования; результаты прогнозирования используются агентами оптимизации и согласования; сформированные решения реализуются агентами исполнения; после чего обновлённое состояние среды вновь поступает в систему наблюдения. Таким образом, мультиагентная архитектура реализует замкнутый цикл:

Observe → Analyze → Decide → Execute.

Распределённый характер наблюдаемости особенно важен для крупномасштабных ГРВС, поскольку централизованный сбор и обработка всей информации становятся вычислительно неэффективными и плохо масштабируемыми. Мультиагентный подход позволяет распределить обработку данных между локальными компонентами системы и тем самым уменьшить нагрузку на управляющую инфраструктуру [14–17].

Кроме того, мультиагентная архитектура обеспечивает естественную интеграцию интеллектуальных методов анализа данных. Отдельные агенты могут использовать специализированные ML-модели, обучаемые алгоритмы и локальные механизмы прогнозирования, адаптированные к особенностям конкретной подсистемы. Это позволяет строить не только распределённый, но и интеллектуализированный контур наблюдаемости.

Таким образом, процессы наблюдаемости и обработки данных являются неотъемлемой частью мультиагентной системы управления ГРВС. Именно распределённое взаимодействие агентов обеспечивает возможность непрерывного мониторинга, анализа состояния среды, прогнозирования изменений и адаптивного управления распределённой вычислительной инфраструктурой.

6.12. Ключевой вывод

Проведённый анализ показывает, что данные являются центральным элементом адаптивного управления гетерогенными распределёнными вычислительными системами. Именно на основе данных формируются процессы наблюдения, оценки состояния среды, прогнозирования, обнаружения аномалий и принятия управляющих решений [12–15].

В отличие от классических детерминированных систем управления, в ГРВС управление осуществляется в условиях неполной, зашумлённой, задержанной и динамически изменяющейся информации. Поэтому эффективность распределения ресурсов, устойчивость функционирования системы и качество управляющих воздействий напрямую зависят от того, насколько корректно организованы процессы наблюдения, обработки данных и оценки состояния среды.

Показано, что наблюдаемость представляет собой не просто механизм сбора телеметрии, а способность системы восстанавливать и интерпретировать своё внутреннее состояние на основе совокупности метрик, логов, трассировок и потоковых данных:

$$\hat{S}(t) = O(Logs, Metrics, Traces).$$

Таким образом, наблюдаемость является необходимым условием реализации адаптивного управления распределённой вычислительной средой. Без формирования адекватной оценки состояния системы невозможны корректное распределение ресурсов, адаптивная балансировка нагрузки, прогнозирование состояния среды, выявление аномалий, интеллектуальная оптимизация и реализация обратной связи в контуре управления.

Особое значение наблюдаемость приобретает в мультиагентных системах управления, где различные агенты используют данные для локального анализа, прогнозирования, согласования и координации распределённых решений. В результате данные становятся связующим элементом всей интеллектуальной архитектуры управления ГРВС.

Выводы по главе 6

В главе рассмотрены процессы наблюдения, обработки данных и формирования оценки состояния гетерогенной распределённой вычислительной системы. Показано, что данные являются информационной основой адаптивного управления распределённой вычислительной средой [10, 12, 15].

Формализован процесс наблюдения состояния системы:

$$\hat{S}(t) = O(Y(t)),$$

где оценка состояния вычислительной среды формируется на основе обработки наблюдаемых данных. Установлено, что наблюдение распределённой системы носит неполный, зашумлённый и задержанный характер, что существенно влияет на качество принимаемых управляющих решений.

Введена модель метрик и наблюдаемых данных, включающая ресурсные, сетевые, задачные и системные показатели. Показано, что метрики являются основным источником информации о состоянии

вычислительной среды и используются в процессах мониторинга, прогнозирования, обнаружения аномалий и адаптивного управления.

Рассмотрено понятие наблюдаемости как способности восстанавливать внутреннее состояние системы по внешним данным:

$$\hat{S}(t) = O(Logs, Metrics, Traces).$$

Установлено, что наблюдаемость включает не только сбор информации, но и её агрегацию, нормализацию, корреляцию, фильтрацию и интерпретацию. Особое внимание уделено потоковой обработке данных и event-driven моделям управления, обеспечивающим непрерывное обновление состояния системы и оперативную реакцию на значимые изменения среды.

Формализован процесс оценки состояния системы:

$$\hat{S}(t) = E(Y(t)),$$

а также рассмотрены методы фильтрации, агрегации, статистического анализа, вероятностного моделирования и машинного обучения, применяемые для формирования устойчивой оценки состояния распределённой вычислительной среды.

Показано, что выявление аномалий является не только диагностическим механизмом, но и средством запуска адаптивных управляющих воздействий. Обнаружение отклонений позволяет своевременно реагировать на перегрузки, деградацию производительности, отказы компонентов и нарушения SLA.

Выполнено замыкание контура адаптивного управления:

$$S(t) \rightarrow Y(t) \rightarrow \hat{S}(t) \rightarrow X(t) \rightarrow S(t+1),$$

что позволило объединить процессы наблюдения, анализа, принятия решений и исполнения управляющих воздействий в единую data-driven архитектуру управления.

Особое внимание уделено проблеме задержек и неопределённости данных. Показано, что система управления работает с приближённой и запаздывающей оценкой состояния:

$$\hat{S}(t) = S(t) + \varepsilon(t),$$

что делает необходимым использование фильтрации, прогнозирования, оценки достоверности данных и интеллектуальных методов анализа состояния среды.

Рассмотрена интеграция процессов наблюдаемости с мультиагентной системой управления. Показано, что функции сбора данных, анализа состояния, прогнозирования, оптимизации и исполнения управляющих воздействий могут быть распределены между специализированными агентами, формирующими замкнутый контур адаптивного управления.

Таким образом, результаты главы формируют информационную основу интеллектуального управления ГРВС и подготавливают переход к рассмотрению методов искусственного интеллекта, обучаемых агентов и больших языковых моделей как компонентов адаптивной мультиагентной системы управления распределённой вычислительной средой.

ГЛАВА 7. ИНТЕЛЛЕКТУАЛЬНЫЕ МЕТОДЫ УПРАВЛЕНИЯ ГЕТЕРОГЕННЫМИ РАСПРЕДЕЛЁННЫМИ ВЫЧИСЛИТЕЛЬНЫМИ СИСТЕМАМИ

7.1. Необходимость интеллектуализации управления

В предыдущих главах было показано, что гетерогенные распределённые вычислительные системы представляют собой высокодинамичные среды, состояние которых непрерывно изменяется под воздействием вычислительной нагрузки, сетевых взаимодействий, доступности ресурсов и внешних факторов [12–15].

Состояние системы описывается функцией:

$$S(t),$$

а процесс управления определяется как отображение:

$$X(t) = A(S(t)).$$

При этом ограничения задачи управления также зависят от текущего состояния вычислительной среды:

$$g_i(X, S(t)) \leq b_i.$$

В главе 6 было показано, что управляющая подсистема, как правило, работает не с истинным состоянием $S(t)$, а с его приближённой оценкой $\hat{S}(t)$, сформированной на основе неполных, зашумлённых и задержанных данных:

$$S(t) = \hat{S}(t) + \varepsilon.$$

Следовательно, управление ГРВС осуществляется в условиях высокой размерности пространства состояний, динамического изменения параметров среды, неопределённости наблюдений, асинхронности процессов, неполной информации, конфликтующих критериев оптимизации и сложных нелинейных зависимостей между компонентами системы.

В подобных условиях классические детерминированные методы управления оказываются недостаточно эффективными. Жёстко заданные алгоритмы и фиксированные правила могут обеспечивать приемлемое поведение в заранее известных режимах, однако плохо адаптируются к изменению структуры нагрузки, особенностей вычислительной среды и поведения распределённых компонентов системы [14–17].

Поэтому эффективное управление ГРВС требует перехода от фиксированных алгоритмических схем к интеллектуальным методам, способным использовать данные, накопленный опыт и прогнозирование для изменения стратегий управления. Такая интеллектуализация позволяет системе не только реагировать на текущее состояние среды, но и выявлять скрытые закономерности, прогнозировать будущие изменения и

адаптировать параметры управления на основе результатов предыдущих решений.

Под интеллектуальными методами управления будем понимать совокупность алгоритмов и моделей, способных использовать данные для формирования адаптивного поведения системы управления. В отличие от классических методов, интеллектуальные алгоритмы не ограничиваются заранее заданными правилами, а могут изменять параметры своей работы в зависимости от наблюдаемого состояния среды, истории функционирования и результатов ранее выполненных управляющих воздействий.

Интеллектуальные методы позволяют учитывать сложные зависимости между параметрами распределённой вычислительной системы:

$$X(t) = A(S(t), H(t)),$$

где $H(t)$ — накопленная история наблюдений и управляющих воздействий.

Это особенно важно для ГРВС, поскольку многие зависимости между состоянием ресурсов, сетевой нагрузкой, временем выполнения задач, ограничениями размещения и эффективностью распределения вычислений являются нелинейными и трудно формализуемыми аналитически. В таких условиях интеллектуальные модели позволяют не только аппроксимировать сложные зависимости, но и использовать их для прогнозирования и выбора управляющих воздействий.

Одним из ключевых преимуществ интеллектуальных методов является способность работать с неопределённостью. Методы машинного обучения, вероятностные модели и нейросетевые архитектуры позволяют формировать устойчивые оценки состояния среды даже при наличии шумов, пропусков наблюдений, задержек и противоречивых данных [15–17].

Кроме того, интеллектуальные методы способны обучаться на накопленных данных. В общем виде процесс изменения параметров модели управления может быть представлен следующим образом:

$$\theta(t+1) = \Phi(\theta(t), S(t), X(t), S(t+1)),$$

где $\theta(t)$ — параметры модели управления, $S(t)$ — оценка состояние, $X(t)$ — выполненное управляющее воздействие, $S(t+1)$ — фактическое новое состояние после применения управления, а Φ — функция обучения или адаптации параметров.

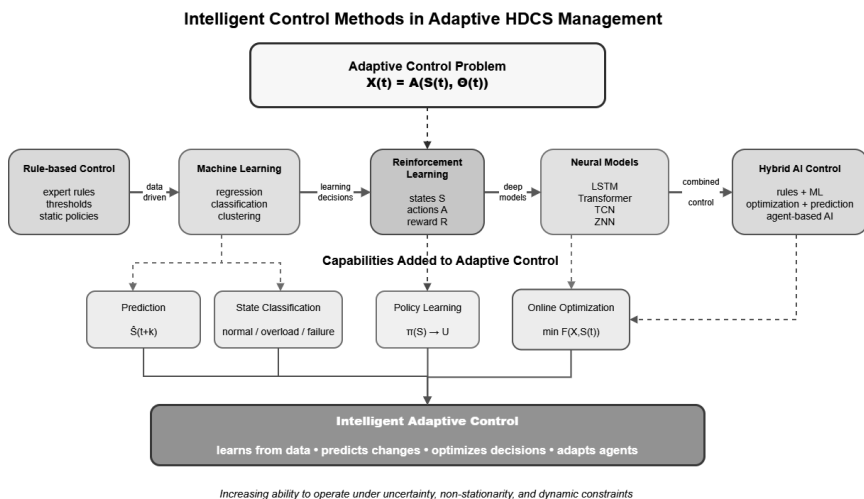
Благодаря этому система управления получает возможность адаптироваться к изменению нагрузки, выявлять повторяющиеся паттерны поведения, прогнозировать будущие состояния среды, улучшать качество распределения ресурсов, снижать вероятность перегрузок и отказов, а

также повышать устойчивость функционирования распределённой вычислительной инфраструктуры.

В современных распределённых вычислительных средах интеллектуальные методы применяются для прогнозирования нагрузки, адаптивной балансировки ресурсов, обнаружения аномалий, оптимизации маршрутизации, интеллектуального распределения задач, предиктивного масштабирования, анализа потоков телеметрии и координации мультиагентных систем управления.

Особое значение интеллектуализация приобретает в мультиагентных архитектурах. В таких системах отдельные агенты могут использовать специализированные модели машинного обучения, прогнозирования и принятия решений в пределах своей области ответственности. Это позволяет формировать распределённую интеллектуальную систему управления, способную к локальной адаптации и коллективному принятию решений [14–17].

Рисунок 8 иллюстрирует процесс интеллектуализации системы управления. В рамках данной схемы классические rule-based методы дополняются машинным обучением, reinforcement learning, нейросетевыми моделями и гибридными интеллектуальными механизмами. Эти методы расширяют возможности адаптивного управления за счёт прогнозирования состояния среды, классификации режимов функционирования, обучения стратегий поведения и онлайн-оптимизации управляющих воздействий.



Intelligent control methods extend adaptive management through prediction, classification, policy learning, and online optimization

Рисунок 8 – Интеллектуализация системы управления

Следует отметить, что интеллектуализация не означает полного отказа от классических методов управления. Наиболее эффективными для ГРВС являются гибридные архитектуры, сочетающие эвристические алгоритмы, методы оптимизации, прогнозные модели, машинное обучение, мультиагентное взаимодействие и механизмы адаптивной координации. В такой архитектуре классические методы обеспечивают устойчивость и интерпретируемость, а интеллектуальные модели — адаптивность, прогнозирование и способность учитывать сложные зависимости в данных.

Таким образом, интеллектуальные методы являются естественным развитием адаптивного и мультиагентного подходов к управлению ГРВС. Их применение позволяет перейти от реактивного управления, основанного на фиксированных правилах, к обучаемому и прогнозно-адаптивному управлению, способному повышать эффективность распределения ресурсов, устойчивость системы и способность вычислительной среды адаптироваться к изменению состояния и структуры нагрузки.

7.2. Общая модель интеллектуального управления

Интеллектуальное управление представляет собой развитие адаптивного управления, при котором управляющий контур не только реагирует на изменение состояния среды, но и изменяет собственные параметры на основе данных, накопленного опыта и результатов ранее принятых решений [12–15].

В классической адаптивной постановке управление задаётся как отображение состояния системы в управляющее воздействие:

$$X(t) = A(\hat{S}(t)),$$

где $\hat{S}(t)$ — оценка состояния системы, A — алгоритм управления, а $X(t)$ — формируемое управляющее воздействие.

В интеллектуальной системе управления данная модель расширяется за счёт введения изменяемых параметров модели управления:

$$X(t) = A(\hat{S}(t), \theta(t)),$$

где $\theta(t)$ — параметры интеллектуальной модели управления в момент времени t .

Параметры $\theta(t)$ могут включать веса критериев оптимизации, параметры прогнозных моделей, коэффициенты балансировки нагрузки, правила приоритизации задач, параметры маршрутизации, стратегии распределения ресурсов, характеристики обучаемых моделей и параметры интеллектуальных агентов.

Принципиальное отличие интеллектуального управления состоит в том, что параметры $\theta(t)$ не являются фиксированными. Они изменяются в процессе функционирования системы на основе наблюдаемых данных, результатов предыдущих управляющих воздействий и анализа поведения вычислительной среды. Тем самым система управления получает возможность не только применять заранее заданные правила, но и корректировать собственную стратегию управления.

Процесс обновления параметров модели может быть представлен следующим образом:

$$\theta(t+1) = \Phi(\theta(t), \hat{S}(t), X(t), \hat{S}(t+1)),$$

где Φ — алгоритм обучения или адаптации параметров; $\hat{S}(t)$ — состояние системы до применения управляющего воздействия; $X(t)$ — выполненное управляющее воздействие; $\hat{S}(t+1)$ — новое состояние системы после применения управления.

Данная запись подчёркивает, что интеллектуальная система использует не только текущее состояние среды, но и результат собственных действий. Если управляющее воздействие привело к улучшению состояния системы, соответствующая стратегия может усиливаться. Если же действие вызвало перегрузку, нарушение SLA, рост задержек или иные негативные эффекты, параметры модели должны корректироваться.

Таким образом, интеллектуальное управление становится двухуровневым процессом. На первом уровне формируются текущие управляющие воздействия:

$$X(t) = A(\hat{S}(t), \theta(t)).$$

На втором уровне изменяются параметры самой модели управления:

$$\theta(t+1) = \Phi(\theta(t), \hat{S}(t), X(t), \hat{S}(t+1)).$$

Такой подход позволяет системе адаптироваться к изменению нагрузки, учитывать особенности вычислительной среды, выявлять скрытые закономерности, повышать эффективность распределения ресурсов, уменьшать число неэффективных решений и улучшать качество управления по мере накопления опыта [14–17].

Для ГРВС интеллектуальная модель управления особенно важна, поскольку такие системы характеризуются нестационарной нагрузкой, неоднородностью ресурсов, изменяющимися сетевыми условиями, неполной наблюдаемостью и большим числом взаимосвязанных параметров. В подобных условиях использование фиксированных параметров управления приводит к постепенной деградации эффективности при изменении режима функционирования среды.

Интеллектуальные методы позволяют формировать модели, учитывающие историю наблюдений, результаты предыдущих управляющих воздействий, долгосрочные зависимости, временные паттерны нагрузки и вероятностные характеристики поведения системы. В качестве алгоритма обучения Φ могут использоваться методы машинного обучения, reinforcement learning, эволюционные алгоритмы, нейросетевые модели, байесовские методы, online learning и адаптивные статистические модели.

В мультиагентной системе интеллектуальное управление может реализовываться распределённо. Каждый агент a_i обладает собственными параметрами модели:

$$\theta_i(t),$$

которые обновляются на основе локального состояния и результатов локальных управляющих воздействий:

$$\theta_i(t+1) = \Phi_i(\theta_i(t), \hat{S}_i(t), X_i(t), \hat{S}_i(t+1)).$$

Это обеспечивает локальную адаптацию агентов, распределённое обучение, специализацию стратегий управления, коллективное накопление опыта и адаптацию к особенностям отдельных подсистем. Например, агент вычислительных ресурсов может обучаться локальной балансировке нагрузки, сетевой агент — прогнозированию перегрузок каналов, а агент оптимизации — выбору более эффективных стратегий распределения ресурсов.

Необходимым элементом интеллектуального управления является механизм оценки качества принимаемых решений. Для этого вводится функция качества или полезности:

$$J = Q(\hat{S}_i(t), X(t), S(t+1)),$$

которая позволяет оценить, насколько выполненное управляющее воздействие улучшило состояние системы. Такая функция может учитывать время выполнения задач, загрузку ресурсов, сетевые задержки, энергопотребление, число нарушений SLA, устойчивость системы и стоимость перераспределения ресурсов.

В современных распределённых вычислительных средах интеллектуальное управление используется для адаптивного распределения ресурсов, предиктивного масштабирования, интеллектуальной балансировки нагрузки, прогнозирования отказов, управления потоками данных, обнаружения аномалий и координации мультиагентных систем управления.

Таким образом, интеллектуальное управление представляет собой развитие адаптивного подхода, в котором система изменяет не только управляющие воздействия, но и собственные механизмы принятия решений. За счёт обучения, анализа данных и накопления опыта

формируется основа для построения самообучающихся распределённых систем управления гетерогенными вычислительными средами.

7.3. Машинное обучение в управлении

Методы машинного обучения являются одним из ключевых направлений интеллектуализации управления гетерогенными распределёнными вычислительными системами. Их применение позволяет выявлять зависимости между состоянием вычислительной среды, характеристиками нагрузки и эффективностью управляющих воздействий, а также адаптировать систему управления к изменению структуры распределённой инфраструктуры [14–17].

В отличие от классических алгоритмов, основанных на фиксированных правилах, методы машинного обучения формируют модель на основе анализа накопленных данных о функционировании системы. Для задач управления ГРВС это означает переход от заранее заданных эвристик к моделям, способным учитывать реальные режимы работы вычислительной среды, статистику выполнения задач, сетевые задержки, поведение очередей и результаты предыдущих управляющих решений.

7.3.1. Постановка задачи машинного обучения в контуре управления

В общем случае задача машинного обучения в контексте управления ГРВС может быть сформулирована следующим образом. Пусть задан набор наблюдений:

$$D = \{(\hat{S}_i, X_i, R_i)\},$$

где $\hat{S}_i$ — наблюдаемое состояние системы, X_i — управляющее воздействие или решение, принятое в данном состоянии, а R_i — результат выполнения этого решения, отражающий изменение состояния системы или значение показателя качества управления.

Требуется построить обучаемую модель:

$$Z(t) = M(\hat{S}(t), H(t)),$$

где $Z(t)$ — прогноз, класс состояния, оценка риска, рекомендация или управляющее решение.

В качестве состояния системы $\hat{S}$ могут использоваться загрузка вычислительных узлов, сетевые характеристики, параметры очередей задач, состояние потоков данных, показатели SLA, история изменения нагрузки и результаты предыдущих управляющих воздействий. Выходом модели могут быть распределение задач между узлами, выбор стратегии балансировки нагрузки, параметры масштабирования, маршруты передачи

данных, приоритеты обработки задач или решения о миграции вычислений.

Следует отметить, что в практических системах машинное обучение не всегда непосредственно формирует управляющее воздействие. Во многих случаях ML-модель решает вспомогательную, но критически важную задачу: прогнозирует нагрузку, оценивает риск перегрузки, классифицирует состояние среды, выявляет аномалию или определяет принадлежность задачи к определённому классу. Затем результат работы модели используется алгоритмами оптимизации, агентами согласования или эвристическими механизмами распределения ресурсов.

Таким образом, задача машинного обучения в управлении ГРВС заключается не только в аппроксимации функции $X = f(S)$, но и в построении интеллектуальных моделей, повышающих качество наблюдения, прогнозирования, выбора стратегии управления и адаптации распределённой вычислительной среды.

7.3.2. Регрессионные методы

Регрессионные методы используются для прогнозирования числовых параметров системы:

$$y^{\wedge} = f(S),$$

где $y^{\wedge}$ — прогнозируемое значение, зависящее от текущего или исторического состояния вычислительной среды.

В контексте управления ГРВС регрессионные модели применяются для прогнозирования нагрузки, оценки времени выполнения задач, предсказания использования CPU, памяти и GPU, оценки сетевых задержек, прогнозирования энергопотребления и вероятности перегрузки отдельных компонентов системы. Эти прогнозы позволяют системе управления принимать решения не только на основе текущего состояния, но и с учётом ожидаемой динамики среды.

Например, прогноз будущей загрузки узла может использоваться агентом вычислительных ресурсов для предотвращения перегрузки; прогноз сетевой задержки — сетевым агентом при выборе маршрута передачи данных; прогноз времени выполнения задачи — агентом качества обслуживания при оценке риска нарушения SLA.

В качестве регрессионных моделей могут использоваться линейная регрессия, деревья решений, случайные леса, градиентный бустинг, нейросетевые модели, LSTM и Transformer-архитектуры. Выбор конкретной модели зависит от объёма данных, требований к интерпретируемости, допустимой вычислительной сложности и характера временной динамики нагрузки.

Для адаптивного управления особенно важны модели, способные работать с временными зависимостями. В таких случаях прогноз может быть представлен как:

$$\hat{S}_i(t+k) = f(\hat{S}_i(t), \hat{S}_i(t-1), \dots, \hat{S}_i(t-n)),$$

где $\hat{S}_i(t+k)$ — прогнозируемое состояние системы через k шагов управления. Такая постановка позволяет компенсировать задержки наблюдения и формировать проактивные управляющие воздействия [15–17].

7.3.3. Классификационные методы

Классификационные методы используются для определения режима функционирования системы или принадлежности текущего состояния к одному из заранее заданных классов:

$$class = f(S).$$

В задачах управления ГРВС классификация применяется для выбора стратегии управления, определения режима функционирования системы, обнаружения аномальных состояний, классификации типов нагрузки, оценки риска перегрузки, выявления отказов и деградации производительности.

Например, состояние вычислительной среды может быть классифицировано как нормальное, перегруженное, критическое, аварийное или нестабильное. Такая классификация позволяет агентам управления выбирать соответствующие действия: продолжать штатное распределение задач, включать режим балансировки, запускать перераспределение ресурсов, ограничивать поступление новых задач или инициировать процедуры восстановления.

Классификационные модели особенно важны в event-driven системах управления, где необходимо быстро определить характер изменения состояния и выбрать реакцию управляющего контура. В таких системах классификация может выступать промежуточным звеном между наблюдением события и формированием управляющего воздействия:

$$e \rightarrow class(S) \rightarrow X(t).$$

Для классификации могут применяться логистическая регрессия, деревья решений, SVM, нейронные сети, ансамблевые методы и вероятностные классификаторы. При этом в критически важных системах значимым становится не только качество классификации, но и интерпретируемость результата, поскольку ошибочное отнесение состояния к безопасному или критическому классу может приводить к неэффективным управляющим решениям [14–17].

7.3.4. Кластеризация состояний, ресурсов и нагрузки

Кластеризация относится к методам обучения без учителя и используется для выявления скрытой структуры данных без заранее заданных меток классов:

$$S \rightarrow C_k,$$

где C_k — кластер состояний системы, вычислительных узлов, задач или режимов нагрузки.

В задачах управления ГРВС кластеризация применяется для группировки вычислительных узлов, сегментации нагрузки, выявления типовых режимов работы системы, обнаружения нетипичных состояний и формирования специализированных стратегий управления. Например, вычислительные узлы могут автоматически группироваться по производительности, энергоэффективности, сетевой близости, степени загрузки или роли в распределённой инфраструктуре.

Кластеризация нагрузки позволяет выявлять периодические паттерны, пики активности, устойчивые режимы функционирования и особенности поведения различных классов задач. Это особенно важно для систем, в которых потоки задач неоднородны и различаются по вычислительной сложности, объёму данных, требованиям к памяти, сетевому обмену и допустимому времени выполнения.

В распределённых системах кластеризация может использоваться также для построения иерархических архитектур управления и организации мультиагентных групп взаимодействия. Например, агенты, обслуживающие близкие по характеристикам узлы или схожие классы задач, могут использовать общие стратегии управления или обмениваться специализированными моделями прогнозирования.

На практике могут применяться k-means, hierarchical clustering, DBSCAN, spectral clustering и графовые методы кластеризации. Выбор метода зависит от структуры данных, необходимости учитывать топологию системы и требований к обнаружению нетипичных состояний.

Отдельное направление связано с повышением точности, устойчивости и воспроизводимости результатов кластеризации. Для этих целей применяются ансамбли алгоритмов кластеризации, методы автоматической группировки с повышенными требованиями к стабильности результата, самоадаптивный поиск с переменными окрестностями для задачи k-means, а также алгоритмы k-medoids с изменяемыми жадными эвристиками [75–78]. Такие подходы представляют интерес для управления ГРВС, поскольку позволяют устойчиво выделять однородные группы вычислительных узлов, задач и режимов нагрузки в условиях неоднородных данных и изменяющейся структуры системы.

Следует отметить, что кластеризация часто используется совместно с другими методами машинного обучения. Например, сначала выполняется выделение классов нагрузки, после чего для каждого класса строится собственная регрессионная модель прогнозирования или выбирается специализированная стратегия распределения ресурсов [15–17].

7.3.5. Машинное обучение в мультиагентном контуре управления

В мультиагентной системе отдельные агенты могут использовать различные ML-модели в зависимости от своей области ответственности.

Агенты прогнозирования применяют регрессионные модели и временные ряды для оценки будущей нагрузки; агенты обнаружения аномалий используют классификационные и вероятностные модели; агенты координации могут применять кластеризацию состояний и группировку ресурсов; агенты оптимизации используют результаты ML-моделей как входные данные для выбора управляющих воздействий.

Такое распределение ML-функций позволяет не строить единую универсальную модель управления для всей ГРВС, а формировать совокупность специализированных моделей, работающих в рамках единого адаптивного контура. Это соответствует природе гетерогенных распределённых систем, где разные подсистемы имеют различные характеристики, источники данных и требования к скорости принятия решений.

При этом машинное обучение не заменяет алгоритмы распределения ресурсов, рассмотренные в главе 5. Оно усиливает их за счёт прогнозирования, классификации состояний, выявления скрытых закономерностей и адаптации параметров управления. Например, greedy-алгоритм может использовать ML-прогноз нагрузки при выборе узла, алгоритм маршрутизации — прогноз сетевых задержек, а многокритериальная оптимизация — оценку риска нарушения SLA.

Таким образом, методы машинного обучения создают основу для построения интеллектуальных систем управления ГРВС, способных обучаться на данных, выявлять сложные закономерности и адаптироваться к изменению состояния распределённой вычислительной среды. Их основная роль заключается не в замене классических алгоритмов, а в повышении качества анализа состояния, прогнозирования и выбора управляющих воздействий в условиях неопределённости и динамической нагрузки.

7.4. Обучение с подкреплением (Reinforcement Learning)

Обучение с подкреплением является одним из перспективных направлений интеллектуального управления гетерогенными распределёнными вычислительными системами. В отличие от классических методов машинного обучения, ориентированных преимущественно на построение модели по заранее накопленным данным, reinforcement learning направлено на формирование стратегии поведения системы в процессе взаимодействия со средой [14–17].

Для задач управления ГРВС данный подход представляет особый интерес, поскольку качество управляющего воздействия определяется не только его немедленным эффектом, но и долгосрочными последствиями. Например, текущее перераспределение задач может снизить нагрузку одного узла, но одновременно увеличить сетевые задержки, вызвать

дополнительную миграцию данных или создать перегрузку в другой части системы. Поэтому в динамической распределённой среде важно учитывать не только локальный выигрыш, но и накопленный эффект последовательности решений.

7.4.1. Формализация

В reinforcement learning система управления рассматривается как агент, взаимодействующий со средой. В общем виде такая постановка может быть представлена кортежем:

$$E = \langle S, A, P, R \rangle,$$

где S — множество состояний среды, A — множество допустимых действий, P — модель переходов между состояниями, R — функция награды.

Состояние среды

$$S(t).$$

В контексте ГРВС под состоянием RL-агента фактически понимается наблюдаемое или восстановленное состояние среды, то есть $\hat{S}(t)$, а не полное истинное состояние системы. Фактически состояние $S(t)$ является тем информационным описанием вычислительной среды, на основе которого система управления выбирает дальнейшее действие.

Множество действий

$$A$$

соответствует возможным управляющим воздействиям: распределению задач, миграции вычислений, изменению маршрутов передачи данных, масштабированию ресурсов, изменению приоритетов обработки задач или корректировке параметров балансировки нагрузки.

Модель переходов

$$P(S_{t+1} | S_t, A_t)$$

описывает вероятность перехода системы в новое состояние S_{t+1} после выполнения действия A_t в состоянии S_t . Для ГРВС такая модель, как правило, является сложной, нелинейной и частично неопределённой, поскольку результат управляющего воздействия зависит от текущей нагрузки, состояния сети, поведения других агентов, задержек наблюдения и внешних факторов.

Функция награды

$$R_t = R(S_t, A_t, S_{t+1})$$

задаёт оценку качества принятого решения. В задачах управления ГРВС награда может учитывать время выполнения задач, загрузку ресурсов, соблюдение SLA, энергопотребление, устойчивость системы, число перегрузок, стоимость вычислений и объём накладных расходов на перераспределение задач.

Таким образом, reinforcement learning позволяет формировать стратегию управления, ориентированную не только на текущее улучшение

состояния среды, но и на долгосрочное повышение эффективности распределённой вычислительной системы.

7.4.2. Цель обучения

Целью обучения с подкреплением является построение такой стратегии поведения, которая максимизирует суммарную ожидаемую награду:

$$\max E[\sum_{t=0}^{\infty} \gamma^t R_t],$$

где R_t — награда на шаге t , γ — коэффициент дисконтирования, E — математическое ожидание.

Коэффициент дисконтирования

$$0 \leq \gamma < 1$$

определяет значимость будущих наград по сравнению с текущими. При больших значениях γ система ориентируется на долгосрочную эффективность управления, а при малых — преимущественно на краткосрочные результаты.

Для управления ГРВС это имеет принципиальное значение. Локально выгодное решение не всегда является системно эффективным. Например, агрессивная балансировка нагрузки может вызывать чрезмерную миграцию задач; минимизация текущей загрузки отдельного узла может приводить к росту сетевых задержек; локальная оптимизация может ухудшать глобальную устойчивость распределённой системы. Reinforcement learning позволяет учитывать подобные отложенные эффекты и формировать стратегии, ориентированные на устойчивое функционирование вычислительной среды [15–17].

Стратегия управления в RL может быть представлена как отображение:

$$\pi: S \rightarrow A,$$

где π определяет выбор действия в зависимости от состояния среды. В задачах распределения ресурсов такая стратегия может определять, на какой узел назначить задачу, когда инициировать миграцию, как изменить приоритеты очередей или когда запустить масштабирование вычислительных ресурсов.

7.4.3. Применение в управлении ГРВС

В задачах управления распределёнными вычислительными системами обучение с подкреплением может применяться для распределения задач, балансировки нагрузки, управления вычислительными ресурсами, адаптивной маршрутизации, масштабирования сервисов, управления потоками данных, энергосберегающего распределения вычислений и оптимизации мультиагентного взаимодействия.

Одним из наиболее показательных направлений является использование RL для интеллектуального распределения задач:

$$X(t) = \pi(S(t)),$$

где стратегия π формируется в процессе взаимодействия с вычислительной средой.

В такой постановке система постепенно обучается выбирать вычислительные узлы, избегать перегрузок, учитывать сетевые ограничения, минимизировать задержки выполнения и поддерживать баланс между производительностью, стоимостью и устойчивостью системы.

Для гетерогенных распределённых систем особенно важно, что RL позволяет учитывать последствия последовательности управляющих воздействий. Например, решение о миграции задачи оценивается не только по текущему снижению загрузки узла, но и по последующим изменениям сетевого трафика, времени выполнения задач, состоянию очередей и вероятности нарушения SLA.

В мультиагентных системах возможно использование multi-agent reinforcement learning, при котором различные агенты обучаются совместно и координируют свои стратегии управления. Локальная стратегия агента может быть представлена как:

$$\pi_i: S_i \rightarrow A_i,$$

где S_i — локальное состояние, наблюдаемое агентом, а A_i — множество доступных ему действий.

Такой подход позволяет реализовывать распределённое обучение, адаптировать локальные стратегии, формировать коллективное поведение агентов и повышать устойчивость управления. Например, агент вычислительных ресурсов может обучаться локальной балансировке нагрузки, сетевой агент — адаптивной маршрутизацией, а агент координации — согласованию локальных решений.

Особый интерес представляют методы deep reinforcement learning, сочетающие обучение с подкреплением и нейросетевые модели. Они позволяют работать с высокоразмерными пространствами состояний, сложными нелинейными зависимостями, потоковыми данными и частично наблюдаемыми средами. Для ГРВС это особенно важно, поскольку состояние вычислительной среды описывается большим числом взаимосвязанных метрик, а управляющие воздействия имеют отложенные и распределённые последствия.

7.4.4. Ограничения и практическое использование

Несмотря на высокий потенциал, применение reinforcement learning в реальных ГРВС связано с рядом ограничений. К ним относятся высокая вычислительная сложность обучения, необходимость большого объёма данных, проблема устойчивости обучения, сложность интерпретации стратегии и риск неустойчивого поведения в процессе исследования среды.

Особенно критичной является проблема безопасного обучения. RL-агент, исследующий новые действия непосредственно в работающей вычислительной инфраструктуре, может сформировать неудачные

управляющие воздействия, приводящие к перегрузке узлов, нарушению SLA, избыточной миграции задач или деградации производительности. Поэтому использование RL в реальных системах требует ограничений безопасности и механизмов контроля допустимости действий.

На практике reinforcement learning целесообразно применять в составе гибридных систем управления. В таких системах обучаемая стратегия дополняется эвристическими правилами, формальными ограничениями, прогнозными моделями, методами оптимизации и мультиагентной координацией [14–17]. Например, RL может выбирать стратегию распределения ресурсов, но окончательное решение проверяется на допустимость с точки зрения ресурсных, сетевых и временных ограничений.

Перспективным направлением является обучение RL-моделей на имитационных стендах, цифровых двойниках или исторических данных функционирования системы. Такой подход позволяет предварительно сформировать стратегию управления без риска нарушения работы реальной вычислительной инфраструктуры. После этого обученная политика может использоваться в рабочей системе в ограниченном или контролируемом режиме.

Таким образом, обучение с подкреплением создаёт основу для построения самообучающихся систем управления ГРВС, способных формировать стратегии распределения ресурсов на основе опыта взаимодействия с вычислительной средой. Однако наиболее рациональным является не изолированное применение RL, а его включение в гибридный контур интеллектуального управления, где обучаемые стратегии сочетаются с ограничениями безопасности, эвристическими механизмами, прогнозированием и мультиагентной координацией.

7.5. Нейросетевые модели

Нейросетевые модели являются важным инструментом интеллектуального управления гетерогенными распределёнными вычислительными системами. Их применение обусловлено способностью выявлять сложные нелинейные зависимости между параметрами вычислительной среды, работать с высокоразмерными потоковыми данными и адаптироваться к изменяющейся структуре нагрузки [14–17].

В отличие от классических аналитических моделей, нейронные сети не требуют явного задания функциональных зависимостей между состоянием системы и управляющими воздействиями. Модель формируется в процессе обучения на основе наблюдаемых данных:

$$\theta(t+1) = \Phi(\theta(t), Y(t)),$$

где $\theta(t)$ — параметры нейросетевой модели, $Y(t)$ — наблюдаемые данные, а Φ — алгоритм обучения или обновления параметров.

В задачах управления ГРВС нейросетевые модели могут выполнять две основные функции. Первая связана с анализом и прогнозированием состояния вычислительной среды. Вторая — с аппроксимацией функции управления, то есть с построением отображения от состояния системы к управляющему воздействию. В обоих случаях нейросетевые модели не заменяют весь контур управления, а встраиваются в него как интеллектуальный компонент, повышающий качество прогнозирования, оценки состояния и выбора решений.

Примером практического применения нейросетевых методов к обработке данных распределённых наблюдательных систем является модель ForestSAR-YOLO [79], предназначенная для обнаружения малых объектов на изображениях, получаемых с беспилотных летательных аппаратов. Этот результат показывает, что специализированные нейросетевые архитектуры могут использоваться как интеллектуальный слой предварительной обработки и интерпретации данных, формирующий событийную информацию для последующего анализа и принятия управляющих решений.

7.5.1. Прогнозирование состояния вычислительной среды

Одним из наиболее распространённых направлений применения нейросетевых моделей является прогнозирование состояния распределённой вычислительной среды:

$$\hat{S}(t+1) = NN(\hat{S}(t)),$$

где NN — нейросетевая модель прогнозирования.

В более общем случае прогноз может строиться не только по текущему состоянию, но и по истории наблюдений:

$$\hat{S}(t+k) = NN(\hat{S}(t), \hat{S}(t-1), \dots, \hat{S}(t-n)).$$

В качестве входных данных могут использоваться временные ряды загрузки CPU, использования памяти и GPU, сетевые метрики, параметры очередей задач, история распределения ресурсов, характеристики потоков данных, события мониторинга, логи и телеметрия.

Прогнозирование позволяет заранее выявлять рост нагрузки, предсказывать перегрузки ресурсов, оценивать вероятность нарушения SLA, выполнять предиктивное масштабирование и формировать проактивные управляющие воздействия. Для ГРВС это особенно важно, поскольку задержки наблюдения и обработки данных могут приводить к запаздывающим решениям. Нейросетевой прогноз позволяет частично компенсировать этот эффект и принимать решения с учётом ожидаемого состояния среды [15–17].

Например, прогноз роста очереди задач может использоваться агентом качества обслуживания для изменения приоритетов обработки,

прогноз перегрузки узла — агентом вычислительных ресурсов для перераспределения задач, а прогноз увеличения сетевой задержки — сетевым агентом для изменения маршрутов передачи данных.

7.5.2. Аппроксимация функций управления

Нейронные сети могут использоваться не только для прогнозирования, но и для аппроксимации функции управления:

$$Recommendation(t) = NN(\hat{S}(t), H(t)),$$

В данном случае нейросетевая модель выступает как интеллектуальный компонент, преобразующий состояние системы в управляющее воздействие или рекомендацию для алгоритма управления. Выходом модели могут быть распределение задач между узлами, параметры балансировки нагрузки, выбор маршрутов передачи данных, решения о миграции вычислений, параметры масштабирования или приоритеты выполнения задач.

Преимуществом такого подхода является способность учитывать сложные взаимосвязи между большим числом параметров системы, которые трудно описать аналитически. Для ГРВС это особенно важно, поскольку эффективность распределения ресурсов зависит одновременно от состояния вычислительных узлов, структуры сетевых потоков, динамики очередей, ограничений размещения, особенностей вычислительных задач и поведения других компонентов системы.

Однако непосредственное использование нейросетевой модели как управляющего компонента требует осторожности. Ошибочное решение модели может привести к перегрузке ресурсов, нарушению SLA или нестабильному перераспределению задач. Поэтому в практических системах нейросетевые модели часто используются не как полностью автономный управляющий механизм, а как часть гибридного контура: они формируют прогноз, оценку риска или рекомендацию, после чего решение проверяется с помощью эвристических правил, ограничений допустимости и механизмов оптимизации.

Таким образом, нейросетевая аппроксимация функции управления наиболее эффективна в сочетании с классическими алгоритмами распределения ресурсов, мультиагентной координацией и механизмами контроля устойчивости.

7.5.3. Нейросетевые архитектуры для временных и потоковых данных

Для различных задач управления распределёнными вычислительными системами могут использоваться специализированные нейросетевые архитектуры. Их выбор определяется типом данных, временной динамикой нагрузки, требованиями к скорости обработки и характером управляющей задачи.

LSTM-модели применяются для анализа временных рядов и прогнозирования динамики состояния системы. В общем виде прогноз может быть представлен следующим образом:

$$\hat{S}(t+1) = LSTM(\hat{S}(t-k), \dots, \hat{S}(t)).$$

Такие модели позволяют учитывать зависимость текущего состояния от предыдущих наблюдений и поэтому полезны для прогнозирования нагрузки, использования ресурсов, сетевых задержек и состояния очередей задач. Их применение особенно оправдано в системах, где нагрузка имеет выраженные временные паттерны и зависит от предшествующих состояний [15–17].

Transformer-архитектуры применяются для анализа сложных зависимостей между параметрами распределённой системы:

$$X(t) = Transformer(\hat{S}(t)).$$

Их важным свойством является механизм attention, позволяющий выделять наиболее значимые элементы состояния системы при формировании прогноза или управляющей рекомендации. Для ГРВС это полезно в ситуациях, когда состояние среды формируется большим числом взаимосвязанных параметров, распределённых между различными узлами, сервисами и агентами.

Transformer-модели могут использоваться для анализа многомерных потоков данных, обработки логов и событий, выявления значимых факторов перегрузки, поддержки интеллектуального распределения ресурсов и координации мультиагентного взаимодействия. При этом их применение требует учёта вычислительной стоимости и требований к объёму обучающих данных.

TCN-модели используются для обработки потоковых данных и временных последовательностей:

$$\hat{S}(t+1) = TCN(Y(t)).$$

Temporal Convolutional Networks хорошо подходят для анализа телеметрии, обработки потоков мониторинга, обнаружения аномалий, прогнозирования нагрузки и event-driven анализа. Их преимуществом является эффективная обработка длинных временных последовательностей при сравнительно низкой вычислительной сложности по сравнению с некоторыми рекуррентными архитектурами.

Таким образом, LSTM, Transformer и TCN следует рассматривать не как изолированные архитектуры, а как разные инструменты обработки временной и потоковой информации в контуре интеллектуального управления ГРВС.

7.5.4. Роль нейросетевых моделей в гибридном управлении

В интеллектуальных системах управления ГРВС нейросетевые модели чаще всего используются в составе гибридных архитектур. Это связано с тем, что нейронные сети хорошо выявляют сложные зависимости в данных, но требуют контроля устойчивости, интерпретируемости и допустимости принимаемых решений.

В гибридном контуре нейросетевая модель может выполнять несколько функций:

$$\begin{aligned}\hat{S}(t+1) &= NN(\hat{S}(t)), \\ risk(t) &= NN(\hat{Y}(t)), \\ \hat{X}(t) &= NN(\hat{S}(t)).\end{aligned}$$

В первом случае модель прогнозирует будущее состояние среды, во втором — оценивает риск перегрузки, отказа или нарушения SLA, в третьем — формирует рекомендацию по управляющему воздействию. Эти результаты могут использоваться агентами прогнозирования, оптимизации, согласования и исполнения решений.

Например, агент прогнозирования может использовать LSTM или TCN для оценки будущей нагрузки, агент мониторинга — autoencoder-модель для выявления аномалий, сетевой агент — графовую или attention-модель для анализа маршрутов, а агент оптимизации — нейросетевую оценку риска при выборе распределения ресурсов.

В исследованиях автора и представителей научной школы показано, что применение нейросетевых моделей позволяет повысить эффективность адаптивного управления распределёнными вычислительными системами, особенно в условиях высокой динамики нагрузки, неопределённости данных и сложной структуры ограничений [2–5].

Следует отметить, что специализированные нейросетевые методы динамической оптимизации, включая Zeroing Neural Networks, требуют отдельного рассмотрения, поскольку они ориентированы не только на прогнозирование или аппроксимацию зависимости, но и на решение задач адаптивной оптимизации в изменяющейся среде. Поэтому в настоящем разделе нейросетевые модели рассматриваются как общий класс интеллектуальных инструментов, а специализированные ZNN-подходы анализируются далее отдельно.

Таким образом, нейросетевые модели являются важным элементом интеллектуального управления ГРВС. Они обеспечивают прогнозирование состояния среды, анализ потоковой телеметрии, выявление сложных зависимостей и поддержку выбора управляющих воздействий. Наиболее эффективно их применение в составе гибридных архитектур, где нейросетевые прогнозы и рекомендации сочетаются с эвристическими правилами, оптимизационными методами, мультиагентной координацией и механизмами контроля устойчивости.

7.6. Гибридные методы

Практическое применение интеллектуальных методов в гетерогенных распределённых вычислительных системах показывает, что использование одного класса моделей редко обеспечивает одновременно высокое качество решений, устойчивость, интерпретируемость и допустимую вычислительную сложность. Методы машинного обучения и нейросетевые модели способны выявлять сложные зависимости и прогнозировать изменение состояния среды, однако они требуют данных, вычислительных ресурсов и механизмов контроля корректности решений. Классические эвристики и оптимизационные методы, напротив, обладают большей предсказуемостью, но хуже адаптируются к изменяющейся структуре нагрузки [14–17].

В связи с этим наиболее практичным направлением интеллектуализации управления ГРВС являются гибридные методы. В данной главе гибридный подход рассматривается не просто как объединение нескольких алгоритмов, а как архитектурный механизм, позволяющий включать обучаемые модели в контур адаптивного управления без потери устойчивости, управляемости и контроля допустимости решений.

Гибридные интеллектуальные методы могут объединять эвристические правила, методы многокритериальной оптимизации, прогнозные модели, алгоритмы машинного обучения, нейросетевые компоненты, reinforcement learning и механизмы мультиагентной координации. Их основная идея заключается в том, что различные компоненты системы управления выполняют разные функции: одни обеспечивают быстрые и интерпретируемые реакции, другие — прогнозирование, третьи — поиск более качественных решений, а координационные механизмы согласуют локальные действия с глобальными ограничениями.

В общем виде гибридная модель управления может быть представлена следующим образом:

$$X(t) = \alpha(t)f_{rule}(\hat{S}(t)) + \beta(t)f_{opt}(\hat{S}(t)) + \gamma(t)f_{ML}(\hat{S}(t), H(t)),$$

где f_{rule} — компонент, основанный на правилах и эвристиках; f_{opt} — оптимизационный компонент; f_{ML} — обучаемая интеллектуальная модель; $H(t)$ — история наблюдений и управляющих воздействий; $\alpha(t)$, $\beta(t)$, $\gamma(t)$ — коэффициенты значимости отдельных компонентов управления.

Компонент правил

$$f_{rule}(\hat{S}(t))$$

обеспечивает базовую устойчивость и предсказуемость системы управления. Он может включать экспертные правила, пороговые механизмы, жадные алгоритмы, эвристики балансировки нагрузки,

простые стратегии маршрутизации и правила аварийного реагирования. Преимуществом такого компонента является низкая вычислительная сложность, высокая скорость реакции и интерпретируемость решений. Поэтому rule-based механизмы особенно важны в критических ситуациях, когда система должна быстро сформировать безопасное управляющее воздействие.

Оптимизационный компонент

$$f_{opt}^{\wedge}(S(t))$$

ориентирован на поиск эффективного распределения ресурсов с учётом ограничений:

$$g_i(X, S(t)) \leq b_i.$$

Он может включать многокритериальную оптимизацию, локальный поиск, эволюционные методы, генетические алгоритмы и распределённую оптимизацию. Его задача состоит в повышении качества решений по сравнению с простыми эвристиками. Однако оптимизационные методы могут требовать значительного времени вычислений, поэтому в высокодинамичных ГРВС они часто используются не для мгновенного реагирования, а для периодической корректировки стратегий, уточнения параметров или согласования решений между агентами.

Интеллектуальный компонент

$$f_{ML}^{\wedge}(S(t), H(t))$$

использует методы машинного обучения, прогнозирования, нейросетевые модели и обучаемые стратегии управления. Его задача состоит в выявлении сложных зависимостей, прогнозировании нагрузки, обнаружении аномалий, оценке риска нарушения SLA, адаптации к изменению среды и выборе более эффективных стратегий управления. Например, ML-модель может прогнозировать перегрузку узла, нейросетевая модель — оценивать будущую сетевую задержку, а RL-компонент — предлагать стратегию перераспределения задач.

Коэффициенты

$$\alpha(t), \beta(t), \gamma(t)$$

могут изменяться во времени в зависимости от режима функционирования системы. В критических ситуациях может возрастать роль rule-based компонента, обеспечивающего безопасную и быструю реакцию. В стабильных режимах может усиливаться оптимизационный компонент, позволяющий улучшить качество распределения ресурсов. При наличии достаточного объёма качественных данных может возрастать роль ML-компонента, отвечающего за прогнозирование и адаптацию.

Такой подход позволяет избежать двух крайностей. С одной стороны, система не сводится к жёстким правилам, плохо работающим в новых режимах нагрузки. С другой стороны, управление не передаётся

полностью обучаемой модели, решения которой могут быть нестабильными, трудно интерпретируемыми или недопустимыми с точки зрения ограничений ГРВС.

Для практического управления распределённой вычислительной средой особенно важна схема, в которой интеллектуальная модель не принимает окончательное решение автономно, а формирует прогноз, оценку риска или рекомендацию. Затем это решение проверяется с помощью ограничений допустимости, эвристических правил и механизмов оптимизации. В таком случае управляющее воздействие можно представить как:

$$X(t) = \text{Validate}(f_{ML}(\hat{S}(t), H(t)), \hat{GS}((t))),$$

где $\hat{G}(\hat{S}(t))$ — множество ограничений, определяющих допустимость управляющих воздействий, а *Validate* — процедура проверки и согласования решения.

В мультиагентной архитектуре гибридные методы могут распределяться между различными агентами. Monitoring Agent может использовать простые правила и ML-модели для выявления аномалий; Prediction Agent — нейросетевые и статистические модели прогнозирования; Optimization Agent — методы многокритериальной оптимизации и локального поиска; Coordination Agent — правила согласования и проверку ограничений; Compute Agent и Network Agent — быстрые эвристики исполнения и локальной адаптации.

В результате формируется распределённая интеллектуальная система управления, в которой гибридность проявляется не только на уровне алгоритма, но и на уровне архитектуры. Отдельные агенты используют разные методы в соответствии со своей ролью, а глобальное поведение системы возникает как результат их согласованного взаимодействия.

В исследованиях автора и представителей научной школы показано, что комбинирование интеллектуальных, оптимизационных и эвристических методов позволяет повысить устойчивость и эффективность управления распределёнными вычислительными системами в условиях динамической нагрузки и неопределённости среды [2–5].

Таким образом, гибридные интеллектуальные методы являются наиболее реалистичным способом внедрения искусственного интеллекта в управление ГРВС. Они позволяют объединить адаптивность обучаемых моделей, качество оптимизационных методов, быстрое действие эвристик и устойчивость мультиагентной координации. В контексте настоящей монографии гибридный подход выступает как архитектурная основа перехода от отдельных интеллектуальных моделей к целостной системе адаптивного интеллектуального управления распределённой вычислительной средой.

7.7. Интеллектуализация агентов

Рассмотренная в главе 4 мультиагентная архитектура управления обеспечивает декомпозицию функций управления между специализированными агентами. Однако сама по себе мультиагентная организация ещё не гарантирует интеллектуального поведения системы. Для реализации адаптивного и самообучающегося управления агенты должны быть дополнены механизмами анализа данных, прогнозирования, машинного обучения, оценки состояния и интеллектуального принятия решений [14–17].

Интеллектуализация агентов означает, что каждый агент перестаёт быть только исполнителем фиксированных правил и становится автономным компонентом распределённой системы управления, способным анализировать локальное состояние среды, использовать обучаемые модели, корректировать параметры поведения и взаимодействовать с другими агентами при формировании общесистемного решения.

В общем виде интеллектуальный агент может быть представлен следующим образом:

$$a_i^{AI} = \langle Obs_i, Model_i, Dec_i, Act_i, K_i, \theta_i \rangle,$$

где Obs_i — функция наблюдения агента; $Model_i$ — интеллектуальная модель анализа, прогнозирования или принятия решений; Dec_i — механизм выбора управляющего воздействия; Act_i — множество допустимых действий; θ_i — параметры обучаемой модели агента.

В такой постановке агент не только получает информацию о состоянии среды, но и преобразует её в управленчески значимое представление:

$$\hat{S}_i(t) = Model_i(O_i(\hat{S}(t))).$$

На основе этого представления агент формирует локальное решение:

$$X_i(t) = Dec_i(\hat{S}_i(t), \theta_i(t)).$$

Таким образом, интеллектуализация агентов связывает три ранее рассмотренных элемента монографии: мультиагентную декомпозицию управления, данные наблюдаемости и обучаемые модели интеллектуального анализа.

Для систематизации ролей интеллектуальных агентов целесообразно представить их функции в виде таблицы.

Таблица 7.1 — Интеллектуальные функции агентов в мультиагентной системе управления ГРВС

Класс агента	Интеллектуальная функция	Возможные модели	Управленческий результат
Monitoring Agent	анализ телеметрии и выявление отклонений	статистические модели, anomaly detection, autoencoder	обнаружение аномалий и формирование оценки состояния
Compute Agent	локальная балансировка и прогноз перегрузок	регрессия, нейросетевые модели, RL	адаптивное распределение задач между вычислительными узлами
Network Agent	интеллектуальная маршрутизация и прогноз задержек	графовые модели, ML, RL	снижение сетевых задержек и предотвращение перегрузки каналов
Prediction Agent	прогноз состояния среды	LSTM, TCN, Transformer, временные модели	переход от реактивного к проактивному управлению
Optimization Agent	выбор согласованного распределения ресурсов	многокритериальная оптимизация, ML, гибридные методы	формирование глобально эффективного решения
Coordination Agent	согласование локальных решений	правила согласования, multi-agent learning, federated learning	предотвращение конфликтов между агентами

7.7.1. Интеллектуализация Compute Agent

Агент вычислительных ресурсов отвечает за локальное управление вычислительными узлами, контроль загрузки и балансировку вычислительной нагрузки. В интеллектуальной архитектуре он использует не только текущие значения загрузки CPU, памяти или GPU, но и историю изменения нагрузки, параметры очередей задач, характеристики выполняемых задач и прогноз локальных перегрузок.

Принятие решений таким агентом может быть представлено следующим образом:

$$X_i(t) = Model_{compute}(S_i^{local}(t)),$$

где $S_i^{local}(t)$ — локальное состояние вычислительного узла или группы узлов, а $Model_{compute}$ — обучаемая модель принятия решений.

Интеллектуальный Compute Agent может прогнозировать локальные перегрузки, выбирать допустимые узлы для размещения задач, адаптировать стратегии балансировки, выявлять аномальное поведение и учитывать особенности конкретного вычислительного ресурса. Это особенно важно для гетерогенных сред, где вычислительные узлы существенно различаются по архитектуре, производительности, доступной памяти, поддержке ускорителей и режимам эксплуатации [15–17].

При этом обучаемая модель не обязательно должна самостоятельно принимать окончательное решение. В практической системе она может формировать прогноз или рекомендацию, которая затем проверяется с учётом ресурсных ограничений, правил безопасности и решений других агентов.

7.7.2. Интеллектуализация Network Agent

Network Agent управляет сетевыми потоками, маршрутизацией и распределением коммуникационной нагрузки. В интеллектуальной системе он должен учитывать не только текущую загрузку каналов, но и структуру сетевой топологии, прогноз изменения потоков данных, историю задержек и взаимосвязь сетевых решений с размещением вычислительных задач.

Интеллектуальный выбор маршрутов может быть представлен следующим образом:

$$Path(t) = Model_{net}(G_{net}(t), Load(t), H(t)),$$

где $G_{net}(t)$ — граф сети, $Load(t)$ — состояние сетевой нагрузки, $H(t)$ — история наблюдений, а $Model_{net}$ — модель машинного обучения или интеллектуальной оптимизации маршрутов.

Такой агент способен прогнозировать перегрузки каналов, адаптивно изменять маршруты передачи данных, выявлять узкие места сети, оптимизировать задержки передачи и предотвращать деградацию сетевой производительности. Для этого могут использоваться графовые нейронные сети, reinforcement learning, прогнозные модели и методы анализа потоковых данных.

Особое значение интеллектуальная маршрутизация приобретает в высоконагруженных распределённых системах с интенсивным межузловым обменом данными. В таких системах неудачное размещение задачи или выбор маршрута может привести к тому, что сетевые ограничения станут главным фактором деградации производительности [14–17].

7.7.3. Интеллектуализация Optimization Agent

Optimization Agent выполняет глобальную координацию и оптимизацию распределения ресурсов. В отличие от локальных агентов, ориентированных на отдельные подсистемы, данный агент должен

учитывать глобальные цели управления, конфликтующие критерии и ограничения, формируемые другими агентами.

В интеллектуальной архитектуре глобальное решение может быть представлено как результат гибридной оптимизации:

$$X^*(t) = \text{Hybrid}(Opt, ML, \hat{S}(t)),$$

где Opt — оптимизационный компонент, ML — обучаемая модель, а $\hat{S}(t)$ — оценка состояния вычислительной среды.

Интеллектуальный Optimization Agent может адаптировать веса критериев оптимизации, учитывать долгосрочные последствия решений, использовать прогноз состояния среды, согласовывать локальные решения агентов и формировать глобально устойчивое распределение ресурсов. Он объединяет многокритериальную оптимизацию, эвристики, прогнозные модели, reinforcement learning и нейросетевые оценки риска.

Использование гибридных методов позволяет сочетать устойчивость классической оптимизации, адаптивность ML-моделей и скорость эвристических алгоритмов. В результате система управления становится способной работать в условиях динамической нагрузки, неопределённости данных и конфликтующих ограничений [15–17].

7.7.4. Интеллектуализация Prediction Agent

Prediction Agent является одним из ключевых компонентов интеллектуального контура управления. Его задача состоит в прогнозировании будущего состояния вычислительной среды:

$$\hat{S}(t + 1) = \text{Model}_{pred}(S(t), H(t)).$$

В качестве модели прогнозирования могут использоваться LSTM, Transformer, TCN, графовые нейросети, статистические модели и гибридные прогнозные архитектуры. Выбор модели зависит от характера данных, временной динамики нагрузки, масштаба системы и требований к скорости прогноза.

Prediction Agent выполняет прогнозирование нагрузки, оценку вероятности перегрузок, предсказание нарушения SLA, прогнозирование сетевых задержек, анализ временных паттернов и выявление долгосрочных тенденций изменения состояния системы.

Результаты прогнозирования используются другими агентами для перехода от реактивного управления к проактивному:

$$X(t) = A(\hat{S}(t + 1)).$$

Это позволяет принимать решения с учётом ожидаемого состояния среды, а не только текущих наблюдений. Именно Prediction Agent связывает слой наблюдаемости, рассмотренный в главе 6, с интеллектуальными методами адаптивного управления.

7.7.5. Коллективная интеллектуализация мультиагентной системы

Интеллектуализация отдельных агентов приводит к формированию распределённой интеллектуальной системы управления:

$$MAS_{AI} = \{a_1^{AI}, a_2^{AI}, \dots, a_n^{AI}\}.$$

В такой системе агенты не только выполняют локальные функции управления, но и обучаются, обмениваются знаниями, координируют стратегии, адаптируют параметры поведения и совместно формируют глобальное управляющее решение.

Особое значение приобретает распределённое обучение агентов:

$$\theta_i(t+1) = \Phi_i(\theta_i(t), \hat{S}_i(t), X_i(t), \hat{S}_i(t+1)),$$

где каждый агент обновляет собственную модель на основе локальных наблюдений, выполненных действий и результата взаимодействия со средой.

В современных исследованиях развиваются подходы multi-agent reinforcement learning, federated learning, distributed intelligence, cooperative AI systems и graph-based multi-agent architectures. Для ГРВС они особенно перспективны, поскольку позволяют масштабировать интеллектуальное управление, уменьшать нагрузку на центральные компоненты, повышать устойчивость системы, сохранять локальную автономность агентов и адаптироваться к распределённой структуре вычислительной среды [14–17].

При этом коллективная интеллектуализация требует механизмов согласования. Если каждый агент обучается только на локальной информации, его стратегия может оказаться неэффективной с точки зрения системы в целом. Поэтому интеллектуальные агенты должны не только обучаться локально, но и обмениваться агрегированными знаниями, согласовывать решения и учитывать глобальные ограничения управления.

Таким образом, интеллектуализация агентов является естественным этапом развития мультиагентных систем управления ГРВС. Интеграция методов машинного обучения, прогнозирования и интеллектуальной оптимизации позволяет превратить мультиагентную архитектуру в самообучающуюся распределённую систему управления, способную адаптироваться к изменению состояния вычислительной среды и повышать эффективность распределения ресурсов на основе накопленного опыта.

7.8. Адаптация моделей

Одним из ключевых свойств интеллектуальных систем управления гетерогенными распределёнными вычислительными системами является способность моделей изменяться в процессе функционирования среды. В

условиях динамической нагрузки, изменения состава вычислительных ресурсов, нестабильных потоков данных и нестационарного поведения пользователей использование статических моделей управления приводит к постепенному снижению качества принимаемых решений [14–17].

Данная проблема связана с деградацией модели при изменении среды. Модель, обученная на одних режимах нагрузки, может становиться менее эффективной при появлении новых типов задач, изменении сетевой топологии, изменении характеристик вычислительных узлов или смещении распределения входных данных. В результате прогнозы становятся менее точными, а управляющие воздействия — менее эффективными.

Поэтому интеллектуальные компоненты системы управления должны обладать механизмами адаптации параметров:

$$\theta(t+1) = \theta(t) + \Delta\theta,$$

где $\theta(t)$ — текущие параметры модели, а $\Delta\theta$ — изменение параметров на основе новых данных, результатов функционирования системы и оценки качества ранее принятых решений.

Адаптация моделей позволяет учитывать изменение структуры нагрузки, подстраиваться под новые режимы функционирования среды, компенсировать деградацию качества прогнозов, улучшать стратегии распределения ресурсов и повышать устойчивость интеллектуального управления.

Для ГРВС это особенно важно, поскольку распределение задач изменяется во времени, характеристики вычислительных узлов могут меняться, структура сетевых потоков не является стационарной, а параметры среды подвержены случайным воздействиям. Следовательно, модель управления должна рассматриваться не как однажды обученный статический объект, а как компонент адаптивного управляющего контура.

В интеллектуальной мультиагентной системе различные агенты могут использовать собственные адаптируемые модели:

$$\theta_i(t+1) = \theta_i(t) + \Delta\theta_i.$$

Это обеспечивает локальную специализацию и позволяет агентам адаптироваться к особенностям своей области ответственности. Например, агент вычислительных ресурсов может уточнять модель локальной перегрузки, сетевой агент — модель задержек передачи данных, а агент прогнозирования — модель временной динамики нагрузки.

7.8.1. Online learning

Online learning предполагает обновление модели непосредственно в процессе поступления новых данных:

$$\theta(t+1) = \Phi(\theta(t), y_t),$$

где y_t — новое наблюдение, событие мониторинга или результат выполнения управляющего воздействия.

В отличие от пакетного обучения, online learning позволяет адаптироваться в реальном времени, учитывать текущую динамику нагрузки, быстро реагировать на изменение состояния среды и уменьшать задержки переобучения моделей. Данный подход особенно эффективен для потоковой обработки данных, прогнозирования нагрузки, адаптивной балансировки, обнаружения аномалий и event-driven систем управления.

В ГРВС online learning позволяет интеллектуальным агентам постепенно корректировать свои модели по мере изменения состояния распределённой вычислительной среды [15–17]. Например, агент прогнозирования может обновлять параметры модели при изменении суточного профиля нагрузки, а агент мониторинга — уточнять границы нормального поведения при изменении состава задач.

Однако online learning связано с рисками. Слишком частое или неконтролируемое обновление модели может приводить к переобучению на локальных изменениях, накоплению ошибок, нестабильности параметров и чрезмерной чувствительности к шуму или аномалиям. Поэтому на практике online learning должен сопровождаться сглаживанием обновлений, ограничением скорости адаптации, регуляризацией модели и контролем устойчивости обучения.

7.8.2. Incremental learning

Incremental learning ориентировано на постепенное накопление знаний без полного переобучения модели:

$$\theta(t + n) = \Phi(\theta(t), D_{new}),$$

где D_{new} — новая порция обучающих данных.

Основное преимущество incremental learning заключается в способности расширять модель новыми знаниями, сохранять ранее накопленный опыт, уменьшать вычислительные затраты и адаптироваться к новым типам нагрузки или поведения системы. Для распределённых вычислительных систем это особенно важно, поскольку полное переобучение сложных моделей требует значительных вычислительных ресурсов, создаёт дополнительные задержки и может быть затруднено в режиме реального времени.

Incremental learning может использоваться для обновления моделей прогнозирования, адаптации систем обнаружения аномалий, интеллектуального распределения задач и анализа потоков телеметрии. Например, при появлении нового класса задач система может не перестраивать всю модель управления, а постепенно включать новые данные в уже существующую модель.

При этом важной проблемой является сохранение ранее накопленных знаний. Если модель адаптируется только к новым данным, она может ухудшить качество работы на ранее известных режимах функционирования. Поэтому incremental learning должен учитывать баланс

между усвоением новых закономерностей и сохранением устойчивости ранее сформированных стратегий управления.

7.8.3. Transfer learning

Transfer learning основано на переносе знаний между различными моделями, задачами или вычислительными средами:

$$\theta_{target} = Transfer(\theta_{source}).$$

Данный подход позволяет использовать опыт, накопленный в одной распределённой системе, для ускоренного обучения модели в другой среде. В контексте ГРВС transfer learning может применяться для переноса моделей между вычислительными кластерами, адаптации управления к новым типам нагрузки, обучения агентов в новых вычислительных средах и ускорения настройки интеллектуальных систем управления.

Преимуществом transfer learning является уменьшение объёма данных, необходимого для обучения новой модели. Это особенно важно для новых распределённых систем, edge-сред, специализированных вычислительных платформ и гетерогенных архитектур с ограниченным объёмом данных.

В мультиагентной архитектуре transfer learning позволяет обмениваться знаниями между агентами, ускорять адаптацию локальных моделей, формировать коллективный опыт управления и уменьшать время обучения системы. Например, агент, накопивший опыт управления определённым типом узлов или задач, может передать часть параметров модели агенту, работающему в сходной подсистеме.

Однако перенос знаний должен выполняться осторожно. Модель, эффективная в одной вычислительной среде, может оказаться неадекватной в другой из-за различий в топологии сети, типах задач, характеристиках ресурсов или профиле нагрузки. Поэтому transfer learning должен сопровождаться дообучением, валидацией и контролем качества решений в новой среде.

7.8.4. Контроль качества адаптации

Адаптация моделей должна сопровождаться постоянной оценкой качества интеллектуального управления. Для этого используется функция качества:

$$J(t) = Q(\hat{S}(t), X(t), \hat{S}(t+1)),$$

которая позволяет оценить, насколько выполненное управляющее воздействие улучшило состояние системы.

Без механизма контроля непрерывное обновление моделей может приводить к деградации качества решений, нестабильности поведения, накоплению ошибок и потере интерпретируемости моделей. Поэтому современные интеллектуальные системы управления используют валидацию, контроль качества прогнозов, адаптивную регуляризацию,

оценку устойчивости обучения и мониторинг эффективности управляющих воздействий [14–17].

Особое значение имеет контроль изменения качества во времени. Если модель начинает хуже прогнозировать нагрузку, чаще ошибаться при классификации состояний или формировать менее эффективные рекомендации, система должна зафиксировать деградацию и инициировать корректирующие действия: переобучение, откат к предыдущей версии модели, изменение скорости адаптации или временное усиление rule-based компонента управления.

Таким образом, адаптация моделей является необходимым условием интеллектуального управления ГРВС. Использование online learning, incremental learning и transfer learning позволяет формировать самообучающиеся распределённые системы управления, способные изменять свои стратегии по мере накопления опыта. Однако адаптация должна быть управляемой и контролируемой, поскольку некорректное обновление моделей может само стать источником нестабильности управляющего контура.

7.9. Ограничения интеллектуальных методов

Несмотря на значительный потенциал интеллектуальных методов управления, их применение в гетерогенных распределённых вычислительных системах связано с рядом ограничений, которые необходимо учитывать при проектировании архитектуры управления [14–17]. Эти ограничения связаны не только с качеством моделей машинного обучения, но и с устойчивостью всего управляющего контура.

Первым существенным ограничением является вычислительная стоимость интеллектуальных методов. Обучение и применение сложных моделей машинного обучения, нейросетевых архитектур и методов reinforcement learning требует дополнительных вычислительных ресурсов. В условиях ГРВС это особенно важно, поскольку ресурсы, используемые интеллектуальной подсистемой управления, конкурируют с ресурсами, необходимыми для выполнения прикладных задач. Поэтому применение интеллектуальных методов должно быть соотнесено с допустимыми накладными расходами на управление.

Вторым ограничением является зависимость от данных. Эффективность ML- и нейросетевых моделей напрямую определяется полнотой, репрезентативностью и достоверностью обучающих данных. Если данные содержат шум, пропуски, систематические искажения или отражают только ограниченный набор режимов функционирования системы, качество интеллектуального управления может существенно снижаться. Для ГРВС эта проблема особенно значима, поскольку данные

мониторинга являются распределёнными, асинхронными и часто неполными.

Третьим ограничением является риск деградации модели при изменении среды. Модель может хорошо работать на исторических данных, но демонстрировать низкую эффективность при изменении структуры нагрузки, появлении новых типов задач, изменении сетевой топологии или конфигурации вычислительных ресурсов. В условиях нестационарной среды ранее обученная модель может перестать адекватно отражать реальные зависимости между состоянием системы и эффективными управляющими воздействиями [15, 16].

Четвёртым ограничением является риск переобучения. Интеллектуальная модель может слишком точно подстроиться под отдельные режимы функционирования или локальные особенности обучающей выборки, но плохо обобщать решения на новые состояния вычислительной среды. Это особенно опасно для адаптивного управления, поскольку внешне эффективная модель может формировать неустойчивые решения при появлении новых типов нагрузки или ранее не наблюдавшихся комбинаций ограничений.

Пятым ограничением является проблема интерпретируемости. Многие интеллектуальные модели, особенно глубокие нейронные сети, формируют решения, причины которых сложно объяснить в терминах понятных правил или критериев управления. Это затрудняет верификацию решений, анализ ошибок, сопровождение системы и применение интеллектуальных методов в критически важных вычислительных инфраструктурах.

Дополнительную сложность создаёт влияние интеллектуальных методов на устойчивость управляющего контура. Некорректно обученная модель, ошибочный прогноз, чрезмерно агрессивная адаптация параметров или неудачная стратегия reinforcement learning могут привести к неустойчивому перераспределению задач, избыточной миграции сервисов, росту сетевой нагрузки или возникновению колебаний в распределении ресурсов.

Следовательно, интеллектуальные методы не должны внедряться в контур управления как полностью автономная замена классическим алгоритмам. Более рациональным является их применение в составе гибридных архитектур, где обучаемые модели сочетаются с формальными ограничениями, эвристическими правилами, механизмами обратной связи, процедурами контроля качества решений и проверкой допустимости управляющих воздействий [14–17].

В такой архитектуре интеллектуальная модель может формировать прогноз, оценку риска или рекомендацию, однако окончательное управляющее воздействие должно проходить проверку на соответствие ресурсным, сетевым, временным и организационным ограничениям. Это

позволяет использовать преимущества интеллектуальных методов, сохраняя управляемость, устойчивость и предсказуемость поведения системы.

Таким образом, интеллектуальные методы не заменяют полностью классические подходы к управлению, а расширяют их возможности. Их эффективное применение требует баланса между качеством решений, вычислительными затратами, устойчивостью моделей, доступностью данных, интерпретируемостью и безопасностью управляющих воздействий.

7.10. Связь с адаптивным управлением

Интеллектуальные методы являются функциональным расширением контура адаптивного управления, рассмотренного в главе 3. Если базовый адаптивный контур включает наблюдение, анализ, планирование, исполнение и обратную связь, то интеллектуальные методы усиливают прежде всего аналитические, прогностические и решающие компоненты этого контура [14–17].

В структуре адаптивного управления интеллектуальные методы выполняют три ключевые функции: анализ состояния системы, прогнозирование её динамики и поддержку выбора управляющих воздействий.

На этапе анализа интеллектуальные методы используются для интерпретации состояния вычислительной среды, выявления скрытых закономерностей, классификации режимов функционирования и обнаружения аномалий. В этом случае ML-модели, статистические методы и нейросетевые алгоритмы позволяют перейти от анализа отдельных метрик к комплексной оценке состояния ГРВС:

$$\hat{S}(t) = AI(Y(t), H(t)),$$

где $Y(t)$ — наблюдаемые данные, а AI — совокупность интеллектуальных методов анализа состояния среды.

На этапе прогнозирования интеллектуальные методы обеспечивают оценку будущего состояния системы:

$$\hat{S}(t+1) = M(\hat{S}(t), H(t)),$$

где M — прогнозная модель, а $H(t)$ — история наблюдений и управляющих воздействий.

Прогнозные модели позволяют учитывать ожидаемое изменение нагрузки, состояние сетевых каналов, вероятность перегрузки ресурсов и риск нарушения SLA. Это создаёт основу для проактивного управления, при котором управляющее воздействие формируется не после наступления критического состояния, а до его возникновения.

На этапе планирования интеллектуальные методы используются для выбора управляющего воздействия:

$$X(t) = A(S(t), \theta(t)),$$

где $\theta(t)$ — параметры интеллектуальной модели управления.

На данном этапе могут применяться методы оптимизации, reinforcement learning, нейросетевые модели принятия решений и гибридные алгоритмы, объединяющие правила, прогнозирование и многокритериальную оптимизацию. Интеллектуальная модель позволяет учитывать не только текущее состояние системы, но и накопленный опыт, прогнозируемую динамику и результаты ранее выполненных управляющих воздействий.

Таким образом, интеллектуальные методы не заменяют адаптивное управление, а повышают его эффективность за счёт более точного анализа состояния, прогноза динамики системы и выбора рациональных управляющих воздействий. В этом смысле интеллектуализация является естественным развитием адаптивного контура управления ГРВС.

В мультиагентной архитектуре связь интеллектуальных методов с адаптивным управлением реализуется через специализированных агентов. Monitoring Agent формирует данные наблюдения, Prediction Agent выполняет анализ и прогнозирование, Optimization Agent формирует план управляющих воздействий, а Compute Agent и Network Agent реализуют принятые решения. Такая организация обеспечивает распределённую реализацию интеллектуального адаптивного управления в условиях динамической и частично наблюдаемой вычислительной среды [15–17].

В результате адаптивный контур приобретает интеллектуальные свойства: способность обучаться на данных, прогнозировать изменение состояния среды, корректировать параметры моделей, учитывать долгосрочные последствия управляющих решений и повышать качество управления по мере накопления опыта.

Таким образом, связь интеллектуальных методов с адаптивным управлением заключается в том, что они превращают адаптивный контур из реактивного механизма корректировки состояния в обучаемую, прогнозную и самоулучшающуюся систему управления гетерогенной распределённой вычислительной средой.

7.11. Ключевой вывод

Проведённый анализ показывает, что интеллектуальные методы являются необходимым компонентом современного адаптивного управления гетерогенными распределёнными вычислительными системами. Их роль заключается не только в повышении качества отдельных управляющих решений, но и в обеспечении способности

системы обучаться, прогнозировать изменение состояния среды и адаптировать собственные стратегии управления в условиях неопределённости [14–17].

Интеллектуализация управления позволяет перейти от фиксированных правил и заранее заданных эвристик к обучаемому контуру управления, использующему данные, накопленный опыт и прогнозирование. Это особенно важно для ГРВС, где состояние системы изменяется во времени, ограничения зависят от текущего режима функционирования, а наблюдаемая информация может быть неполной, зашумлённой и задержанной.

Ключевое значение интеллектуальных методов состоит в том, что они обеспечивают анализ сложных зависимостей между параметрами системы, прогнозирование будущих состояний, адаптацию параметров моделей управления, обучение на исторических и потоковых данных, поддержку принятия решений в условиях неопределённости и повышение эффективности мультиагентной координации.

Таким образом, интеллектуальные методы формируют аналитическую и прогностическую основу адаптивного управления ГРВС. В сочетании с мультиагентной архитектурой они позволяют построить распределённую систему управления, способную не только реагировать на изменение состояния среды, но и прогнозировать его, обучаться на результатах собственных действий и постепенно повышать качество управления вычислительной инфраструктурой.

Выводы по главе 7

В главе рассмотрены интеллектуальные методы управления гетерогенными распределёнными вычислительными системами и показана их роль в построении адаптивных мультиагентных систем управления вычислительной инфраструктурой [14–17].

Установлено, что динамический характер ГРВС, зависимость ограничений от состояния среды, неполнота наблюдаемых данных и высокая сложность процессов распределения ресурсов делают недостаточным применение исключительно классических алгоритмов управления. Эффективное управление распределённой вычислительной средой требует методов, способных обучаться, прогнозировать, адаптироваться к изменению состояния системы и учитывать результаты принятых решений.

Формализована общая модель интеллектуального управления:

$$X(t) = A(\hat{S}(t), \theta(t)),$$

где $\theta(t)$ — параметры модели управления, изменяемые в процессе функционирования системы. Обновление параметров может быть представлено следующим образом:

$$\theta(t+1) = \Phi(\theta(t), \hat{S}(t), X(t), S(t+1)).$$

Показано, что интеллектуальное управление представляет собой развитие адаптивного подхода, в котором система изменяет не только управляющие воздействия, но и собственные стратегии принятия решений.

Рассмотрены методы машинного обучения, используемые в задачах управления ГРВС. Показано, что регрессионные методы применяются для прогнозирования нагрузки, времени выполнения задач, сетевых задержек и энергопотребления; классификационные методы — для определения режима функционирования системы, выявления аномалий и оценки риска перегрузки; кластеризация — для группировки вычислительных узлов, сегментации нагрузки и выделения типовых режимов функционирования среды.

Особое внимание уделено обучению с подкреплением. Формализована модель взаимодействия интеллектуального агента со средой:

$$E = \langle S, A, P, R \rangle,$$

а также целевая функция максимизации накопленной награды:

$$\max E[\sum_{t=0}^{\infty} \gamma^t R_t].$$

Показано, что reinforcement learning позволяет формировать стратегии распределения ресурсов и балансировки нагрузки на основе опыта взаимодействия с распределённой вычислительной средой. При этом отмечено, что практическое применение RL в ГРВС требует гибридных и безопасных схем, ограничивающих риск неустойчивых управляющих воздействий.

Рассмотрены нейросетевые модели, применяемые для интеллектуального управления ГРВС. Показано, что они используются для прогнозирования состояния системы, аппроксимации функций управления, анализа потоковых данных и обработки временных рядов. Проанализированы LSTM, Transformer и TCN как инструменты работы с временной и потоковой информацией в контуре управления.

Обоснована целесообразность применения гибридных интеллектуальных методов:

$$X(t) = \alpha(t)f_{rule}(\hat{S}(t)) + \beta(t)f_{opt}(\hat{S}(t)) + \gamma(t)f_{ML}(\hat{S}(t), H(t)).$$

Показано, что гибридные архитектуры являются наиболее практичным способом внедрения интеллектуальных моделей в управление ГРВС, поскольку позволяют объединить устойчивость классических методов, качество оптимизационных алгоритмов, адаптивность обучаемых моделей и контроль допустимости управляющих воздействий.

Выполнена интеллектуализация мультиагентной архитектуры управления. Показано, что специализированные агенты могут использовать обучаемые модели в соответствии со своей областью ответственности: Monitoring Agent — для анализа телеметрии и обнаружения аномалий, Compute Agent — для локальной балансировки и прогноза перегрузок, Network Agent — для интеллектуальной маршрутизации, Prediction Agent — для прогноза состояния среды, Optimization Agent — для выбора согласованного распределения ресурсов, Coordination Agent — для согласования локальных решений.

Рассмотрены механизмы адаптации моделей:

$$\theta(t + 1) = \theta(t) + \Delta\theta,$$

а также подходы online learning, incremental learning и transfer learning. Показано, что адаптация моделей позволяет интеллектуальной системе управления изменять стратегии распределения ресурсов по мере накопления опыта функционирования вычислительной среды. При этом подчёркнуто, что адаптация должна сопровождаться контролем качества, поскольку некорректное обновление моделей может стать источником нестабильности управляющего контура.

Проанализированы ограничения интеллектуальных методов, включая вычислительную стоимость, зависимость от качества данных, риск деградации модели при изменении среды, переобучение, недостаточную интерпретируемость и возможное влияние на устойчивость управления. Установлено, что эффективное применение интеллектуальных методов требует их интеграции с механизмами адаптивного управления, многокритериальной оптимизации, эвристическими правилами и мультиагентной координацией.

Показана связь интеллектуальных методов с контуром адаптивного управления. Интеллектуальные алгоритмы усиливают этапы анализа состояния среды, прогнозирования и планирования управляющих воздействий, превращая адаптивный контур из реактивного механизма корректировки состояния в обучаемую, прогнозную и самоулучшающуюся систему управления.

Таким образом, результаты главы формируют методологическую основу построения интеллектуальных мультиагентных систем управления ГРВС, способных обучаться, адаптироваться и принимать решения в условиях неопределённости, неполной наблюдаемости и динамического изменения состояния распределённой вычислительной среды.

ГЛАВА 8. LLM-АГЕНТЫ В АДАПТИВНОМ УПРАВЛЕНИИ И РАЗРАБОТКЕ РАСПРЕДЕЛЁННЫХ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ

8.1. Эволюция интеллектуальных методов управления

В предыдущей главе были рассмотрены интеллектуальные методы управления гетерогенными распределёнными вычислительными системами, основанные на машинном обучении, нейросетевых моделях, reinforcement learning и гибридных алгоритмах [14–17]. Эти подходы позволяют повысить эффективность распределения ресурсов, прогнозирования нагрузки, обнаружения аномалий и адаптации управляющего контура к изменяющемуся состоянию вычислительной среды.

Классические интеллектуальные методы в основном ориентированы на обработку числовых данных и могут быть представлены как модели:

$$X(t) = A(S(t), \theta(t)),$$

где $S(t)$ — оценка состояния системы, $\theta(t)$ — параметры интеллектуальной модели, а $X(t)$ — управляющее воздействие или рекомендация.

В рамках данного подхода интеллектуализация управления базируется преимущественно на анализе временных рядов, обучении на числовых данных, аппроксимации сложных зависимостей, прогнозировании состояния среды, обнаружении аномалий и оптимизации распределения ресурсов. Такие методы эффективны в задачах, где состояние системы может быть представлено в виде формализованного набора метрик, а целевая функция и ограничения достаточно явно заданы.

Однако по мере усложнения распределённых вычислительных систем становится очевидным, что одних числовых моделей недостаточно. Современная ГРВС содержит не только метрики загрузки, сетевые задержки и параметры очередей, но и логи, трассировки, текстовые сообщения, конфигурационные файлы, эксплуатационные политики, документацию, описания инцидентов и команды оператора. Существенная часть информации о состоянии системы имеет семантический, контекстный или организационный характер и не сводится к набору числовых признаков.

Первым ограничением классических интеллектуальных методов является слабая интерпретируемость решений. Многие ML- и нейросетевые модели функционируют как «чёрный ящик»: они могут формировать прогноз или рекомендацию, но не всегда позволяют объяснить, почему было выбрано конкретное управляющее воздействие. Это затрудняет анализ поведения системы, верификацию решений,

сопровождение моделей и применение подобных методов в критически важных вычислительных инфраструктурах.

Второе ограничение связано с узкой специализацией моделей. Как правило, отдельная модель решает ограниченный класс задач: прогнозирование нагрузки, классификацию состояний, маршрутизацию, обнаружение аномалий или распределение ресурсов. При изменении типа задачи, структуры среды или требований управления может потребоваться переобучение модели, изменение архитектуры или построение нового специализированного компонента.

Третье ограничение состоит в отсутствии универсального механизма рассуждения. Традиционные ML-модели хорошо выявляют статистические зависимости, но хуже работают с логическими связями, абстрактными правилами, контекстной интерпретацией событий, объяснением решений, анализом документации и сложными сценариями координации. Именно эти функции становятся особенно значимыми в мультиагентных системах управления, где требуется не только вычислить локальное решение, но и согласовать действия множества агентов, интерпретировать неоднозначный контекст и учитывать высокоуровневые политики управления.

Развитие больших языковых моделей (Large Language Models, LLM) приводит к формированию нового класса интеллектуальных систем, в которых обработка информации осуществляется не только на уровне числовых признаков, но и на семантическом, логическом и контекстном уровнях [18–21].

В отличие от специализированных ML-моделей, LLM способны работать с естественным языком, интерпретировать текстовые описания состояния системы, анализировать логи и сообщения, использовать документацию, учитывать историю взаимодействий, формировать объяснимые выводы и строить планы действий. Это позволяет рассматривать LLM не просто как средство генерации текста, а как когнитивный компонент интеллектуальной системы управления.

В результате происходит переход от отдельных обучаемых моделей к агентно-ориентированным интеллектуальным системам, в которых языковая модель дополняется памятью, механизмами планирования и доступом к внешним инструментам:

$$a_{LLM} = \langle \textit{Obs}, \textit{Reason}, \textit{Plan}, \textit{Act}, \textit{Memory}, \textit{Tools} \rangle.$$

Такая агентная модель объединяет семантический анализ, рассуждение, накопление опыта, планирование действий, обращение к инструментам мониторинга и взаимодействие с другими агентами. В контексте управления ГРВС это означает появление интеллектуальных компонентов, способных не только прогнозировать числовые параметры среды, но и анализировать контекст функционирования системы,

объяснять причины отклонений, координировать распределённые решения и взаимодействовать с человеком-оператором на естественном языке.

Особенно важным является то, что LLM-агенты могут выступать связующим слоем между различными компонентами интеллектуальной архитектуры управления. Числовые ML-модели продолжают выполнять прогнозирование и классификацию, методы оптимизации — решать формальные задачи распределения ресурсов, мультиагентная система — обеспечивать распределённость управления, а LLM-агенты добавляют уровень рассуждения, интерпретации, планирования, объяснения и координации.

Кроме непосредственного управления вычислительной инфраструктурой, LLM-агенты начинают использоваться в процессах разработки, сопровождения и эксплуатации распределённых систем. Они применяются для генерации программного кода, автоматизации DevOps, анализа логов и телеметрии, подготовки тестов, работы с инфраструктурой как кодом, формирования документации и поддержки операторов. Это расширяет область применения LLM от runtime-управления вычислительной средой к более широкому жизненному циклу распределённых информационных систем.

Таким образом, появление больших языковых моделей формирует новый этап эволюции интеллектуальных методов управления ГРВС. Происходит переход от узкоспециализированных моделей машинного обучения к LLM-агентам как когнитивному слою адаптивной мультиагентной системы управления, способному выполнять семантический анализ, логическое рассуждение, планирование, объяснение решений и координацию распределённых процессов управления вычислительной инфраструктурой.

8.2. Понятие LLM-агента

Под LLM-агентом будем понимать интеллектуального агента, использующего большую языковую модель в качестве ядра семантического анализа, рассуждения, планирования и поддержки принятия решений. Такой агент способен интерпретировать входной контекст, формировать план действий, обращаться к внешним инструментам, использовать память и формировать объяснимые рекомендации, планы или стратегии управляющих действий [18–21].

В отличие от классического интеллектуального агента, основанного на специализированной модели машинного обучения, LLM-агент обладает более универсальным механизмом обработки информации. Он может работать не только с числовыми признаками, но и с текстовыми описаниями состояния системы, логами, трассировками, эксплуатационной

документацией, сообщениями других агентов, результатами инструментов мониторинга и командами оператора.

Для задач управления ГРВС это имеет принципиальное значение. Состояние распределённой вычислительной среды не всегда может быть полностью сведено к набору числовых метрик. Существенная часть информации о работе системы содержится в логах, событиях, сообщениях об ошибках, архитектурных описаниях, правилах эксплуатации, политике безопасности и экспертных комментариях. LLM-агент позволяет включить эти данные в контур интеллектуального управления.

Формально LLM-агент может быть представлен следующим кортежем:

$$a_{LLM} = \langle Obs, Reason, Plan, Act, Memory, Tools \rangle,$$

где *Obs* — функция наблюдения; *Reason* — механизм рассуждения на основе LLM; *Plan* — функция планирования действий; *Act* — множество допустимых действий агента; *Memory* — память агента; *Tools* — множество внешних инструментов, доступных агенту.

Компонент

Obs

обеспечивает получение и структурирование входной информации. В контексте управления ГРВС источниками наблюдений могут быть метрики, логи, трассировки, события мониторинга, состояние задач, сообщения других агентов, результаты выполнения диагностических инструментов и команды оператора. На этом этапе разнородные данные преобразуются в контекст, пригодный для анализа языковой моделью.

Компонент

Reason

реализует семантический и логический анализ полученного контекста. Именно на этом уровне большая языковая модель выполняет интерпретацию состояния системы, выявление возможных причин отклонений, сопоставление наблюдений с политиками управления, анализ конфликтующих ограничений и формирование обоснования возможных действий.

Компонент

Plan

отвечает за построение последовательности действий, направленных на достижение цели управления. План может включать запрос дополнительных данных, запуск инструментов диагностики, обращение к агентам прогнозирования или оптимизации, перераспределение задач, изменение параметров конфигурации, формирование рекомендации оператору или передачу решения на согласование координирующему агенту.

Компонент

Act

определяет допустимые действия агента. В зависимости от уровня автономности LLM-агент может выполнять действия непосредственно через внешние инструменты и API управления инфраструктурой либо формировать предложения для последующего подтверждения человеком, специализированным агентом исполнения или системой валидации. Для критически значимых действий *Act* должен рассматриваться как множество кандидатных действий, проходящих проверку ограничений, политик безопасности и, при необходимости, подтверждение оператора.

Компонент

Memory

обеспечивает хранение и использование накопленного опыта: предыдущих состояний системы, выполненных действий, результатов решений, типовых сценариев отказов, политик управления, ограничений безопасности, эксплуатационной документации и истории взаимодействия с другими агентами. Использование памяти позволяет LLM-агенту учитывать прошлый опыт и повышать согласованность принимаемых решений.

Компонент

Tools

определяет набор внешних инструментов, к которым может обращаться агент. К ним могут относиться системы мониторинга, оптимизационные модули, симуляторы, базы знаний, API управления ресурсами, DevOps-инструменты, средства анализа логов и механизмы проверки ограничений. Наличие инструментов позволяет LLM-агенту не только рассуждать о состоянии системы, но и получать дополнительные данные, проверять гипотезы и инициировать управляющие процедуры.

Таким образом, LLM-агент представляет собой не просто языковую модель, а управляющий интеллектуальный компонент, объединяющий наблюдение, рассуждение, планирование, действия, память и инструментальное взаимодействие. Его использование позволяет дополнить классические методы адаптивного управления семантическим анализом, объяснимостью, работой с неоднородной информацией и способностью к координации распределённых решений в вычислительной среде.

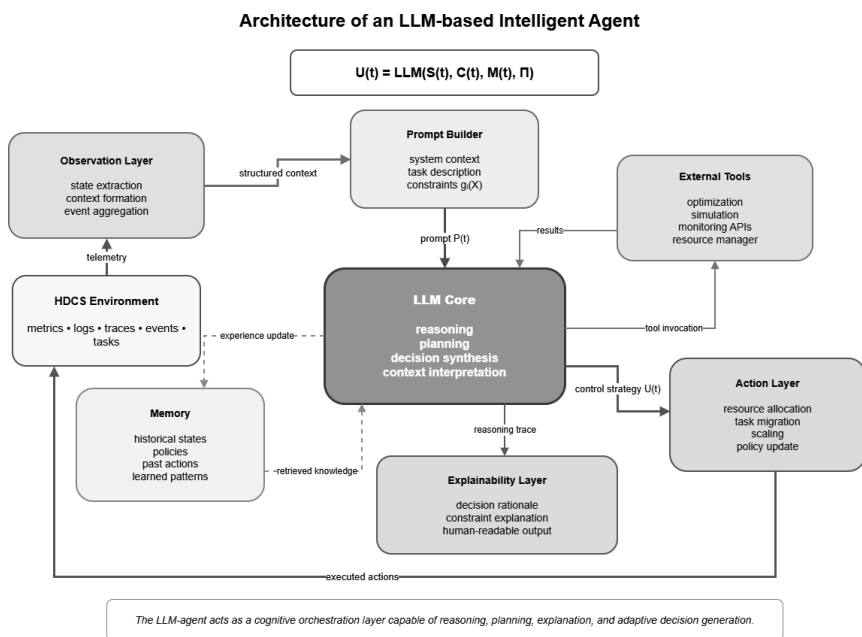
8.3. Архитектура LLM-агента

Функционирование LLM-агента в контуре адаптивного управления может быть представлено как последовательное преобразование наблюдаемых данных и контекста вычислительной среды в план, рекомендацию или согласованное управляющее воздействие через этапы наблюдения, формирования контекста, рассуждения, планирования, валидации, обращения к инструментам и исполнения действий:

$$Y(t) \rightarrow Context(t) \rightarrow Reasoning \rightarrow Plan(t) \rightarrow Validation \rightarrow X^*(t).$$

Данная запись подчёркивает, что LLM-агент не принимает решение непосредственно по исходным метрикам системы. Сначала состояние вычислительной среды преобразуется в контекст, затем интерпретируется языковой моделью с учётом памяти и доступных инструментов, после чего формируется план действий и выполняется управляющее воздействие [18–21].

Рисунок 9 иллюстрирует архитектуру LLM-агента, включающую управляемую среду ГРВС, слой наблюдения, построитель контекста, ядро рассуждения на основе LLM, память, внешние инструменты, слой действий и слой объяснения решений.



Architecture of an LLM-based intelligent agent integrating observation, reasoning, memory, tools, and adaptive actions

Рисунок 9 – Архитектура LLM-агента

Первым компонентом архитектуры является слой наблюдения. Его задача состоит в преобразовании состояния системы в форму, пригодную для последующей обработки языковой моделью:

$$Obs: S(t) \rightarrow Context(t).$$

На этом этапе метрики, события, логи, трассировки, сообщения агентов и результаты мониторинга преобразуются в структурированное

контекстное представление. Такое представление может включать описание текущей нагрузки, отклонений от нормального режима, выявленных аномалий, ограничений, доступных действий и целей управления.

Вторым компонентом является построитель контекста или *prompt builder*. Его функция состоит в том, чтобы объединить наблюдаемые данные, цели управления, ограничения, политики безопасности и релевантную информацию из памяти в единое входное представление для LLM:

$$P(t) = \text{BuildPrompt}(\text{Context}(t), \text{Goals}, \text{Constraints}, \text{Memory}).$$

Качество построения контекста имеет принципиальное значение, поскольку от полноты и корректности представления состояния системы зависит качество последующего рассуждения и планирования. Некорректно сформированный контекст может привести к ошибочной интерпретации состояния среды или генерации недопустимых действий.

Третьим компонентом является ядро рассуждения на основе большой языковой модели:

$$\text{Decision} = \text{LLM}(P(t)).$$

На этом уровне LLM выполняет анализ контекста, выявляет возможные причины проблемного состояния, сопоставляет наблюдения с ограничениями, формирует гипотезы и определяет возможные направления действий. В отличие от классических ML-моделей, языковая модель способна работать с неоднородной информацией, включая числовые данные, текстовые сообщения, логи, политики управления, документацию и описания инфраструктуры.

Четвёртым компонентом является механизм планирования:

$$\text{Plan} = f(\text{Decision}, \text{Goals}, \text{Constraints}).$$

Планирование связывает результаты рассуждения с целями управления и системой ограничений. План может включать последовательность диагностических, оптимизационных или управляющих действий: запрос дополнительных данных, проверку состояния узлов, запуск оптимизационного модуля, перераспределение задач, изменение маршрутов передачи данных, масштабирование ресурсов или передачу решения на согласование другим агентам.

Пятым компонентом является слой внешних инструментов:

$$\text{ToolResult} = \text{Tool}(\text{Plan}, \text{Context}).$$

LLM-агент может обращаться к инструментам мониторинга, оптимизации, симуляции, анализа логов, проверки ограничений и управления ресурсами. Это позволяет агенту не ограничиваться внутренним рассуждением, а проверять гипотезы, получать актуальные данные и использовать специализированные алгоритмы для расчёта формальных решений.

Шестым компонентом является слой действий:

$$Recommendation(t) = Execute(Plan).$$

На этом этапе сформированный план преобразуется в конкретное управляющее воздействие. В зависимости от уровня автономности системы действие может выполняться непосредственно через API управления инфраструктурой, передаваться специализированным агентам исполнения или оформляться как рекомендация оператору.

$$X^*(t) = Validate(Plan(t), G, Policies, Goals).$$

Для критически важных систем выполнение действий должно сопровождаться проверкой ограничений, контролем безопасности и, при необходимости, подтверждением со стороны человека.

Седьмым компонентом является память:

$$Memory = \{History, Knowledge, Policies, PastActions\}.$$

Память хранит историю состояний системы, ранее выполненные действия, результаты решений, типовые сценарии отказов, эксплуатационную документацию, политики управления и накопленные знания о вычислительной среде. Использование памяти позволяет LLM-агенту учитывать опыт предыдущих взаимодействий, сопоставлять текущую ситуацию с ранее наблюдавшимися сценариями и повышать согласованность принимаемых решений.

Восьмым компонентом является слой объяснения решений. Его задача состоит в формировании человеко-понятного обоснования принятого решения:

$$Explanation = Explain(U(t), Context(t), Constraints).$$

Такое объяснение может включать причины выбора управляющего воздействия, указание на значимые факторы состояния системы, описание конфликтующих ограничений, ожидаемые последствия действия и возможные риски. Наличие слоя объяснения особенно важно для систем с участием оператора, критически важных инфраструктур и задач аудита автономных управляющих решений.

Архитектура LLM-агента обеспечивает расширение классического адаптивного контура управления за счёт семантической интерпретации данных, объяснимости решений, работы с текстовым контекстом, использования памяти и взаимодействия с внешними инструментами. При этом LLM-агент должен рассматриваться не как замена формальным методам оптимизации и классическим ML-моделям, а как когнитивный слой, дополняющий мультиагентную систему управления средствами рассуждения, планирования, объяснения и координации.

8.4. Интеграция LLM-агентов в систему управления

Рассмотренная в главе 4 мультиагентная архитектура управления ГРВС включает специализированных агентов мониторинга,

прогнозирования, оптимизации, координации и исполнения управляющих воздействий. Интеграция LLM-агентов не заменяет данную архитектуру, а расширяет её за счёт появления когнитивного уровня, обеспечивающего семантический анализ, рассуждение, планирование, объяснение решений и взаимодействие с человеком-оператором [18–21].

В отличие от специализированных агентов, работающих преимущественно с формализованными параметрами среды, LLM-агент способен обрабатывать неоднородный контекст: метрики, логи, трассировки, сообщения агентов, описания инцидентов, эксплуатационную документацию, ограничения безопасности и высокоуровневые цели управления. Поэтому его целесообразно рассматривать не как ещё один локальный исполнительный агент, а как интеллектуальный слой координации и интерпретации.

С учётом введения LLM-компонентов мультиагентная система управления может быть представлена следующим образом:

$$MAS_{LLM} = \langle A, a_{LLM}, Comm, Tools, Policies, Human \rangle,$$

где $A = \{a_1, \dots, a_p\}$ — множество специализированных агентов управления; a_{LLM} — LLM-агент, реализующий семантический анализ, рассуждение и интеллектуальную координацию; *Comm* — коммуникационные связи между агентами; *Tools* — внешние инструменты мониторинга, оптимизации, диагностики и управления инфраструктурой; *Policies* — политики управления, ограничения безопасности и правила допустимости решений; *Human* — оператор или эксперт, участвующий в контуре управления при необходимости.

Принципиально важно, что LLM-агент не должен функционировать изолированно. Он включается в существующий контур адаптивного управления и взаимодействует с другими агентами через обмен контекстом, анализ результатов наблюдения, формирование гипотез, совместное планирование, согласование решений и генерацию объяснений. При этом выполнение критически важных управляющих воздействий должно оставаться под контролем специализированных исполнительных агентов, механизмов проверки ограничений и, при необходимости, человека-оператора.

Наиболее естественная область интеграции LLM-агентов связана с агентами прогнозирования, оптимизации и координации. Именно эти компоненты требуют не только обработки числовых данных, но и интерпретации контекста, анализа причинно-следственных связей, выбора стратегии и объяснения принимаемых решений.

8.4.1. Интеграция с Prediction Agent

Prediction Agent отвечает за анализ состояния среды и прогнозирование её динамики:

$$\hat{S}(t+1) = Model(S(t), H(t)).$$

Традиционно такой агент использует статистические модели, методы машинного обучения, временные ряды или нейросетевые архитектуры. Интеграция LLM-компонента позволяет дополнить числовое прогнозирование семантической интерпретацией состояния системы.

LLM-агент может анализировать результаты прогнозных моделей, сопоставлять их с логами, событиями и историческими сценариями, выявлять возможные причины прогнозируемых перегрузок и формировать объяснения ожидаемого изменения состояния среды. Например, если прогнозная модель указывает на рост задержек, LLM-агент может сопоставить это с недавними изменениями конфигурации, логами сетевых ошибок или историей аналогичных инцидентов.

В этом случае Prediction Agent не превращается в языковую модель, а получает дополнительный когнитивный слой:

$$\hat{S}^{expl}(t+1) = LLM(\hat{S}(t+1), Logs, Events, History, Policies).$$

Здесь $\hat{S}^{expl}(t+1)$ представляет собой не только прогноз состояния, но и его интерпретацию: возможные причины, уровень риска, связь с ограничениями и рекомендации по дальнейшему анализу.

Такая интеграция особенно полезна в ситуациях, когда числовой прогноз сам по себе недостаточен для принятия решения. Прогноз перегрузки указывает на ожидаемое изменение состояния, но не всегда объясняет, связано ли оно с ростом нагрузки, деградацией сети, отказом компонента, ошибкой конфигурации или изменением поведения приложений.

8.4.2. Интеграция с Optimization Agent

Optimization Agent выполняет задачу формирования управляющего воздействия:

$$X^*(t) = \arg \min_{\hat{X}} F(X, \hat{S}(t))$$

при ограничениях

$$\hat{g}_i(X, \hat{S}(t)) \leq b_i, g_i \in G.$$

Традиционные методы оптимизации ориентированы преимущественно на работу с формальными критериями, ограничениями и числовыми параметрами состояния системы. LLM-агент не должен заменять математическую оптимизацию, поскольку не обеспечивает строгих гарантий оптимальности и допустимости решения. Его роль состоит в поддержке оптимизационного процесса на уровне интерпретации контекста, выбора стратегии и объяснения результатов.

В гибридной архитектуре LLM-компонент может выполнять функции мета-управления оптимизацией:

$$Strategy(t) = LLM(Context(t), Goals, Policies),$$

после чего выбранная стратегия передаётся оптимизационному агенту:

$$X^*(t) = Opt(\hat{S}(t), Strategy(t), G, Goals).$$

В такой постановке LLM-агент может помогать выбирать метод оптимизации, уточнять приоритеты критериев, интерпретировать конфликтующие ограничения, формировать гипотезы о допустимых компромиссах и объяснять полученное решение оператору или другим агентам.

Например, при конфликте между минимизацией времени выполнения задач и снижением энергопотребления LLM-агент может проанализировать текущий режим функционирования системы, политики SLA, критичность задач и историю предыдущих решений. После этого он формирует высокоуровневую стратегию, но само распределение ресурсов рассчитывается оптимизационным модулем с проверкой ограничений.

Такой подход особенно важен в условиях конфликтующих критериев, изменяющихся ограничений, неполной наблюдаемости и необходимости взаимодействия с человеком-оператором. Он позволяет использовать сильные стороны LLM — контекстный анализ и объяснение — без отказа от строгих формальных методов оптимизации.

8.4.3. Интеграция с Coordination Agent

Наиболее значимой областью применения LLM-агентов является координация мультиагентной системы управления. Coordination Agent отвечает за согласование локальных решений, разрешение конфликтов, распределённое планирование, синхронизацию действий агентов и формирование глобальной стратегии управления.

В мультиагентной системе управления разные агенты могут формировать решения, корректные в пределах их локальной области ответственности, но конфликтующие на уровне системы в целом. Например, Compute Agent может предложить перенос задачи на менее загруженный узел, Network Agent может указать на перегрузку маршрута к этому узлу, а QoS Agent — на риск нарушения временного ограничения при миграции.

В таких ситуациях LLM-агент может использоваться как интеллектуальный координационный слой:

Coordination = LLM(Context, LocalDecisions, Goals, Policies).

Он анализирует сообщения агентов, сопоставляет локальные решения с глобальными целями, выявляет конфликтующие ограничения, формирует компромиссные планы и объясняет причины выбранной стратегии. В отличие от формального механизма согласования, LLM-компонент способен работать с текстовыми описаниями, политиками управления, историей инцидентов и не полностью формализованными требованиями.

При этом LLM-агент координации не должен единолично утверждать критические управляющие воздействия. Его решение должно проходить через механизмы проверки допустимости, формальные ограничения и, при необходимости, подтверждение оператора. Таким

образом, LLM выполняет функцию когнитивного координатора, а не неконтролируемого автономного управляющего центра.

8.4.4. Роль LLM-агентов в интеллектуальной архитектуре управления

Интеграция LLM-агентов приводит к формированию гибридной интеллектуальной архитектуры, в которой различные классы методов выполняют взаимодополняющие функции. Классические ML-модели обеспечивают прогнозирование, классификацию и обработку числовых данных. Методы оптимизации решают формальные задачи распределения ресурсов. Мультиагентная система обеспечивает распределённость управления и локальную адаптацию. LLM-агенты реализуют когнитивный и семантический уровень управления.

В такой архитектуре LLM-агенты выполняют функции анализа контекста, интерпретации состояния системы, логического рассуждения, интеллектуального планирования, координации распределённых решений, объяснения действий и взаимодействия с человеком. Их основная ценность состоит в работе с теми аспектами управления, которые трудно полностью описать числовыми моделями или формальными правилами.

Особенно перспективным направлением является использование LLM как интерфейса между человеком-оператором и распределённой системой управления. В этом случае оператор формулирует цель или ограничение на естественном языке, а LLM-агент преобразует его в контекст для мультиагентной системы, инициирует анализ состояния среды, запрашивает результаты у специализированных агентов и формирует объяснимый план действий.

Таким образом, интеграция LLM-агентов расширяет возможности адаптивного мультиагентного управления ГРВС за счёт добавления когнитивного слоя. Этот слой обеспечивает семантический анализ, рассуждение, планирование, объяснение решений и интеллектуальную координацию, но должен работать совместно с формальными методами оптимизации, ML-моделями, специализированными агентами исполнения и механизмами проверки допустимости управляющих воздействий.

8.5. Роли LLM-агентов в управлении

Интеграция больших языковых моделей в мультиагентную архитектуру управления приводит к появлению новых ролей интеллектуальных агентов. В отличие от специализированных агентов, ориентированных преимущественно на обработку числовых метрик, прогнозирование или выполнение локальных управляющих воздействий, LLM-агенты выполняют функции когнитивного уровня: интерпретируют состояние системы, анализируют контекст, формируют гипотезы, согласуют решения и объясняют действия управляющего контура [18–21].

В задачах управления ГРБС LLM-агенты не должны рассматриваться как замена агентам мониторинга, прогнозирования, оптимизации или исполнения. Их роль состоит в расширении мультиагентной архитектуры за счёт семантического анализа, рассуждения, планирования и взаимодействия с человеком-оператором. Поэтому функции LLM-агентов целесообразно рассматривать через их вклад в общий контур управления.

Таблица 8.1 — Основные роли LLM-агентов в управлении ГРБС

Роль LLM-агента	Основная функция	Входные данные	Управленческий результат
Агент анализа состояния	интерпретация состояния вычислительной среды	метрики, логи, трассировки, события, контекст	объяснение состояния, гипотезы о причинах отклонений
Агент поддержки принятия решений	выбор стратегии или формирование рекомендации	состояние, цели, ограничения, прогнозы	план действий или рекомендация для оптимизационного контура
Агент координации	согласование локальных решений агентов	сообщения агентов, локальные решения, политики, SLA	согласованный план управления
Агент объяснения	формирование интерпретируемого обоснования	решение, контекст, ограничения, результаты действий	объяснение для оператора, аудит решения
Агент взаимодействия с оператором	перевод целей оператора в действия системы	естественно-языковые команды, политики, состояние среды	формализованная задача для MAS или рекомендация

8.5.1. Агент анализа состояния

Одной из ключевых ролей LLM-агентов является интеллектуальный анализ состояния распределённой вычислительной среды. В данной роли языковая модель используется как механизм семантической интерпретации наблюдаемого состояния системы:

$$Insight(t) = LLM(Context(t), Logs(t), Metrics(t), Traces(t)).$$

В отличие от классических моделей обнаружения аномалий, которые чаще всего фиксируют отклонение метрик от нормы, LLM-агент может анализировать более широкий контекст: историю событий, текстовые сообщения, логи, документацию, правила эксплуатации, особенности архитектуры вычислительной среды и результаты работы других агентов.

Агент анализа состояния может выявлять возможные причины деградации производительности, сопоставлять аномалии с недавними изменениями конфигурации, анализировать взаимосвязи между сетевыми и вычислительными факторами, формировать гипотезы о причинах проблемного поведения среды и указывать, какие дополнительные данные необходимо запросить для уточнения диагноза.

Результатом работы такого агента является не только факт обнаружения отклонения, но и его интерпретация:

$$Reason(t) = LLM(Context(t), Logs(t), Metrics(t)).$$

Например, рост времени выполнения задач может быть связан не только с перегрузкой CPU, но и с увеличением сетевых задержек, изменением маршрутов передачи данных, ростом очереди на отдельном сервисе или ошибкой конфигурации. LLM-агент позволяет рассматривать подобные ситуации как комплексные сценарии, а не как изолированные отклонения отдельных метрик.

8.5.2. Агент поддержки принятия решений

LLM-агент может использоваться как компонент поддержки принятия решений в контуре управления ГПВС:

$$Recommendation(t) = LLM(S(t), Goals, Constraints, History).$$

В этом случае языковая модель анализирует текущее состояние среды, цели управления, ограничения, прогнозы и историю предыдущих действий, после чего формирует рекомендацию, стратегию или высокоуровневый план управляющих воздействий.

Такой агент может выступать в роли мета-оптимизатора или интеллектуального оркестратора. Он не обязан самостоятельно рассчитывать точное распределение ресурсов, но может выбирать подходящую стратегию оптимизации, определять последовательность действий, предлагать изменение приоритетов, инициировать обращение к специализированному оптимизационному модулю или формировать набор альтернативных сценариев.

Принципиально важно, что LLM в этой роли не заменяет математические методы оптимизации. Более корректной является постановка, в которой LLM формирует стратегию применения формальных методов:

$$Strategy(t) = LLM(Context(t), Goals, Policies),$$

а конкретное управляющее воздействие вычисляется специализированным оптимизационным или эвристическим механизмом:

$$X^*(t) = Decision(\hat{S}(t), Strategy(t), G, Goals).$$

Такой подход позволяет объединять контекстный анализ, логическое рассуждение, формальную оптимизацию, прогнозирование и эвристические алгоритмы. Он особенно полезен в ситуациях, когда

система функционирует в условиях неопределённости, конфликтующих критериев и неполной наблюдаемости.

8.5.3. Агент координации

Одной из наиболее перспективных ролей LLM-агентов является координация мультиагентной системы управления. В распределённых вычислительных системах различные агенты могут формировать решения, которые локально выглядят рациональными, но конфликтуют на уровне всей системы.

Например, агент вычислительных ресурсов может предложить миграцию задачи на менее загруженный узел, сетевой агент может указать на рост задержек при таком переносе, а агент качества обслуживания — на риск нарушения SLA. Формально подобные ситуации могут быть представлены как конфликт ограничений:

$$g_i(X, S(t)) \cap g_j(X, S(t)).$$

Задача LLM-агента координации заключается в анализе таких конфликтов и формировании согласованного решения:

$$\textit{Coordination}(t) = \textit{LLM}(\textit{LocalDecisions}, \textit{Policies}, \textit{Goals}, \textit{Context}).$$

Такой агент сопоставляет локальные цели, выявляет противоречия, учитывает высокоуровневые политики управления, требования безопасности, SLA и приоритеты оператора. В результате он может предложить компромиссный план: например, отложить миграцию, изменить маршрут передачи данных, временно повысить приоритет задачи или запросить дополнительные ресурсы.

Преимущество LLM-агента координации заключается в способности работать не только с формальными ограничениями, но и с текстовыми политиками, эксплуатационными правилами, описаниями инцидентов и сообщениями других агентов. Это позволяет использовать языковую модель как когнитивный механизм согласования распределённых решений.

8.5.4. Агент объяснения

Одним из ключевых преимуществ LLM-агентов является способность формировать интерпретируемые объяснения принятых решений:

$$\textit{Explanation}(t) = \textit{LLM}(X(t), S(t), \textit{Context}(t), \textit{Constraints}).$$

Классические нейросетевые модели и reinforcement learning-системы часто обладают низкой объяснимостью. В результате оператору сложно определить, почему было принято конкретное решение, какие факторы оказались наиболее значимыми, какие ограничения повлияли на выбор стратегии и каковы ожидаемые последствия управляющего воздействия.

LLM-агент объяснения позволяет представить результат работы управляющего контура в человеко-понятной форме. Он может описывать причины перераспределения задач, основания изменения маршрутов,

факторы, вызвавшие масштабирование, причины выявления аномалии, риски выбранной стратегии и ожидаемые последствия действия.

Такая роль особенно важна для критически важных вычислительных инфраструктур, систем с участием человека в контуре управления, аудита решений, сопровождения интеллектуальных систем и повышения доверия к автономным механизмам управления.

При этом следует учитывать, что объяснение, сформированное LLM, также должно проверяться на соответствие фактическим данным. Языковая модель может убедительно сформулировать интерпретацию, которая не полностью соответствует реальной причинно-следственной структуре инцидента. Поэтому объяснение должно опираться на наблюдаемые данные, результаты инструментов мониторинга и формальные ограничения системы.

8.5.5. Агент взаимодействия с оператором

Отдельной важной ролью LLM-агента является взаимодействие с человеком-оператором. В традиционных системах управления оператор взаимодействует с инфраструктурой через панели мониторинга, команды, конфигурационные файлы и специализированные интерфейсы. LLM-агент позволяет добавить естественно-языковой слой взаимодействия:

$$Task = LLM(Command_{human}, Context, Policies).$$

В этом случае оператор может формулировать цель управления на естественном языке, а LLM-агент преобразует её в формализованную задачу для мультиагентной системы: запросить диагностику, проверить состояние узлов, оценить риск перегрузки, подготовить план масштабирования или сформировать рекомендацию по перераспределению ресурсов.

Такой подход особенно полезен в сложных распределённых инфраструктурах, где оператору трудно вручную анализировать множество метрик, логов, ограничений и зависимостей между компонентами. LLM-агент может выполнять роль интеллектуального интерфейса между человеком и системой управления, снижая когнитивную нагрузку и повышая оперативность принятия решений.

Однако взаимодействие через LLM не должно отменять контроль безопасности. Команды оператора, интерпретированные языковой моделью, должны проходить проверку допустимости, особенно если они приводят к изменению конфигурации, масштабированию ресурсов, миграции задач или другим критически значимым действиям.

Таким образом, LLM-агенты расширяют мультиагентную архитектуру управления ГРВС за счёт появления когнитивного уровня обработки информации. Их основные роли связаны с анализом состояния, поддержкой принятия решений, координацией агентов, объяснением управляющих воздействий и взаимодействием с человеком-оператором. В совокупности эти функции создают основу для построения

интеллектуальных распределённых систем управления нового поколения, в которых формальные методы оптимизации и ML-модели дополняются семантическим анализом, рассуждением и объяснимым планированием.

8.6. Мультиагентные LLM-системы

Развитие больших языковых моделей приводит к переходу от отдельных LLM-агентов к распределённым мультиагентным LLM-системам, в которых несколько специализированных интеллектуальных агентов совместно решают задачи анализа, планирования, критики, исполнения и координации управления распределённой вычислительной средой [18–21].

Такой подход особенно важен для гетерогенных распределённых вычислительных систем, поскольку управление ими связано с высокой сложностью состояния среды, распределённым характером ресурсов, множеством конфликтующих ограничений, необходимостью согласования локальных решений и многоэтапным процессом принятия управляющих воздействий.

В этом случае единая языковая модель заменяется системой взаимодействующих специализированных LLM-агентов:

$$LLM_MAS = \{a_{LLM}^1, a_{LLM}^2, \dots, a_{LLM}^k\},$$

где каждый агент выполняет определённую функцию в общем контуре интеллектуального управления.

В отличие от классической мультиагентной системы, где взаимодействие агентов часто ограничено обменом формализованными сигналами, параметрами или управляющими сообщениями, LLM-агенты способны обмениваться более сложными информационными объектами: контекстом, объяснениями, гипотезами, планами, критикой решений, промежуточными рассуждениями и интерпретациями состояния системы. Это позволяет реализовать не только распределённое принятие решений, но и коллективное рассуждение.

Принципиально важно, что мультиагентная LLM-система не должна рассматриваться как неконтролируемая группа языковых моделей. В задачах управления ГРВС такая система должна быть организована как управляемая архитектура с разделением ролей, общей памятью, формальными ограничениями, контролем безопасности, доступом к инструментам и механизмом проверки решений.

8.6.1. Распределение ролей в LLM-MAS

В мультиагентной LLM-системе могут выделяться несколько специализированных ролей, каждая из которых отвечает за отдельный этап формирования управляющего решения.

Planner Agent отвечает за формирование кандидатной стратегии управления и построение плана действий:

$$Plan = LLM(Goals, Constraints, S(t), Context).$$

Данный агент анализирует цели управления, интерпретирует ограничения, декомпозирует задачу, определяет возможные управляющие действия и формирует первоначальный план. В контексте ГРВС такой план может включать перераспределение задач, запуск диагностики, обращение к оптимизационному модулю, изменение маршрутов передачи данных, масштабирование ресурсов или подготовку рекомендации оператору.

Planner Agent функционирует как когнитивный уровень генерации решения, но его результат не должен автоматически становиться управляющим воздействием. Сформированный план должен пройти проверку, согласование и валидацию.

Critic Agent выполняет анализ и оценку предложенного плана:

$$Critique = LLM(Plan, Policies, Risks, Constraints).$$

Его функции включают выявление ошибок, поиск противоречий, анализ ограничений, оценку рисков, проверку устойчивости решения, анализ безопасности и обнаружение потенциальных конфликтов между локальными и глобальными целями управления.

Critic Agent играет роль внутреннего механизма валидации. Он позволяет уменьшить вероятность некорректных управляющих воздействий, особенно в ситуациях, когда решение принимается в условиях неполной информации, конфликтующих критериев или высокой критичности вычислительной инфраструктуры.

Supervisor Agent выполняет функции координации и принятия итогового решения о допустимости плана:

$$Supervisor = LLM(Context, Plan, Critique, Policies, Results).$$

Данный агент координирует взаимодействие других LLM-агентов, контролирует соответствие решения целям управления, учитывает политики безопасности, управляет уточнением плана, разрешает конфликты между агентами и принимает решение о передаче плана на исполнение либо о необходимости его доработки.

Supervisor Agent можно рассматривать как мета-уровень интеллектуального управления, обеспечивающий согласованность работы всей LLM-MAS. Именно этот агент связывает коллективное рассуждение с требованиями управляемости, безопасности и соответствия политике системы.

Executor Agent отвечает за реализацию утверждённого плана:

$$U(t) = Execute(Plan_{approved}).$$

Его задача состоит в преобразовании согласованного плана в конкретные действия: вызов внешних инструментов, запуск диагностических процедур, изменение конфигурации, обращение к API управления инфраструктурой, инициирование масштабирования,

перераспределение задач или передача команд специализированным исполнительным агентам.

Executor Agent связывает когнитивный уровень LLM-системы с реальными механизмами управления вычислительной средой. В практических системах он может взаимодействовать с системами оркестрации, DevOps-инструментами, API облачных платформ, системами мониторинга, оптимизационными модулями и средствами управления ресурсами.

При этом исполнительный агент должен быть наиболее жёстко ограничен правилами безопасности. Именно на этом уровне особенно важны проверка допустимости действий, контроль прав доступа, журналирование операций и возможность human-in-the-loop для критически значимых изменений.

8.6.2. Цикл взаимодействия агентов

Взаимодействие LLM-агентов может быть организовано как итеративный цикл:

$$Plan \rightarrow Critique \rightarrow Refine \rightarrow Execute \rightarrow Learn.$$

На первом этапе Planner Agent формирует кандидатный план действий на основе состояния системы, целей управления, ограничений и доступного контекста:

$$Plan(t) = Planner(Context(t), Goals, Policies),$$

На втором этапе Critic Agent выполняет проверку предложенного плана:

$$Critique(t) = Critic(Plan(t), G, Policies),$$

Если критика выявляет нарушение ограничений, недостаточную обоснованность, риск нестабильности или конфликт с политиками управления, план уточняется:

$$Plan^*(t) = Supervisor(Plan(t), Critique(t), Policies),$$

После согласования Supervisor Agent принимает решение о допустимости плана:

$$X^*(t) = Executor(Plan^*(t)).$$

Затем Executor Agent реализует утверждённый план:

$$U(t) = Executor(Plan_{approved}).$$

После выполнения действия система переходит в новое состояние:

$$S(t + 1) = f(S(t), U(t)).$$

Результаты исполнения фиксируются в общей памяти и используются для последующего улучшения поведения системы:

$$Memory(t + 1) = Update(Memory(t), Context(t), Plan^*(t), S(t + 1)).$$

Такой цикл обеспечивает внутреннюю проверку решений, повышение устойчивости управления, уменьшение числа ошибок, возможность объяснения действий и накопление опыта функционирования системы. В отличие от одиночного LLM-агента, мультиагентная LLM-система может разделить генерацию плана, критику, авторизацию и

исполнение между разными компонентами, что повышает контролируемость архитектуры.

Рисунок 10 иллюстрирует архитектуру мультиагентной LLM-системы. Управляемая ГРВС формирует наблюдения, которые поступают в общий контекст. Planner Agent генерирует кандидатный план, Critic Agent выполняет проверку ограничений и оценку рисков, Supervisor Agent координирует уточнение и авторизацию решения, а Executor Agent обращается к инструментам и реализует утверждённые действия. Общая память сохраняет опыт, а слой Human / Policy задаёт цели, ограничения и правила безопасности.

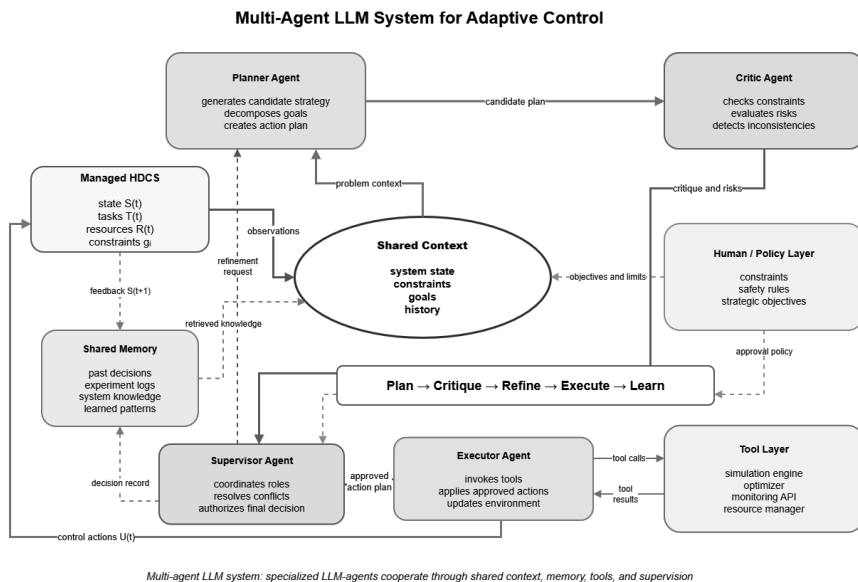


Рисунок 10 – Мультиагентная LLM-система

8.6.3. Особенности применения LLM-MAS в ГРВС

Мультиагентные LLM-системы особенно перспективны для управления облачной инфраструктурой, DevOps-автоматизации, интеллектуальной оркестрации вычислений, анализа распределённых логов, автономного сопровождения систем, управления edge computing и самоорганизующихся вычислительных сред.

Их основными преимуществами являются распределённость интеллектуального управления, способность к коллективному рассуждению, работа с неоднородной информацией, объяснимость решений, адаптивность, возможность интеграции с внешними инструментами и поддержка взаимодействия с человеком-оператором.

Однако применение LLM-MAS связано и с существенными ограничениями. К ним относятся высокая вычислительная стоимость, сложность координации агентов, риск генерации некорректных решений, возможность накопления ошибок в коллективном рассуждении, необходимость контроля безопасности и проблема устойчивости поведения системы при автономном взаимодействии агентов.

$$Plan^*(t) = Validate(Plan(t), G, Policies)$$

Поэтому в практических архитектурах LLM-MAS должны интегрироваться с формальными ограничениями, системами верификации, rule-based механизмами, мониторингом действий агентов, журналированием решений и механизмами human-in-the-loop. Особенно важно, чтобы критические действия не выполнялись только на основании рассуждения LLM, а проходили проверку допустимости и соответствия политике управления.

$$X^*(t) = Execute(Plan^*(t)).$$

Таким образом, мультиагентные LLM-системы формируют новый этап развития интеллектуального управления распределёнными вычислительными системами. Они объединяют мультиагентный подход, языковые модели, коллективное рассуждение, внутреннюю критику решений, работу с инструментами и кооперативное принятие решений. Для ГРВС такой подход создаёт основу построения когнитивных систем управления, способных не только реагировать на изменения состояния среды, но и анализировать контекст, согласовывать стратегии, объяснять действия и накапливать опыт управления вычислительной инфраструктурой.

8.7. LLM в разработке информационных систем

Большие языковые модели становятся не только инструментом анализа и управления распределёнными вычислительными системами, но и важным компонентом инженерного процесса их разработки, сопровождения и эксплуатации. В этом смысле LLM-агенты расширяют контур управления ГРВС за пределы runtime-режима: они могут участвовать в проектировании компонентов, генерации кода, тестировании, документировании, DevOps-автоматизации и сопровождении распределённых платформ [18–21].

Для рассматриваемой предметной области это имеет принципиальное значение. Гетерогенные распределённые вычислительные системы являются не только объектом управления, но и сложными программно-инфраструктурными системами, которые постоянно развиваются: изменяются сервисы, конфигурации, сценарии оркестрации, механизмы мониторинга, правила масштабирования и политики

безопасности. Поэтому интеллектуальная поддержка разработки и сопровождения становится частью общей задачи обеспечения устойчивого функционирования вычислительной инфраструктуры.

В отличие от классических средств автоматизации разработки, ориентированных преимущественно на формальные преобразования программного кода, LLM способны работать с естественным языком, архитектурными описаниями, программными интерфейсами, логикой взаимодействия компонентов, конфигурационными файлами и эксплуатационной документацией. Это позволяет использовать языковые модели как когнитивный слой инженерной среды, связывающий требования, архитектуру, код, тесты, инфраструктуру и документацию.

Одним из наиболее активно развивающихся направлений является автоматизированная генерация программного кода:

Code = LLM(Specification, Context).

В данном случае языковая модель преобразует спецификацию, описание архитектуры или формулировку требований в программную реализацию компонентов системы. Для распределённых вычислительных платформ это особенно важно, поскольку современные cloud-native, микросервисные и мультиагентные архитектуры требуют значительного объёма инфраструктуры, orchestration- и integration-кода.

LLM могут применяться для генерации backend-компонентов, API, микросервисов, конфигураций контейнерной инфраструктуры, сценариев оркестрации, DevOps-конвейеров, средств мониторинга, адаптеров к внешним сервисам и вспомогательных инструментов анализа телеметрии. При этом языковая модель способна учитывать архитектурный контекст проекта, используемые технологии, стиль разработки, требования к масштабируемости и ограничения распределённой вычислительной среды.

Не менее важным направлением является интеллектуальное тестирование программных систем:

Tests = LLM(Code, Requirements, Architecture).

LLM способны анализировать программный код и автоматически формировать тестовые сценарии, включая проверки корректности взаимодействия компонентов, обработку исключительных ситуаций, анализ граничных условий, проверку API-контрактов и сценарии отказоустойчивости. Для распределённых систем данная задача особенно сложна, поскольку число возможных состояний среды и сценариев взаимодействия между компонентами чрезвычайно велико.

В отличие от традиционных генераторов тестов, языковые модели способны учитывать не только синтаксическую структуру программы, но и её предполагаемую семантику, архитектурные зависимости, роли компонентов и логику функционирования системы. Это позволяет использовать LLM для формирования более содержательных тестовых

сценариев, связанных с реальными эксплуатационными ситуациями: деградацией сетевого канала, отказом узла, ростом очередей задач, изменением конфигурации или нарушением SLA.

Существенное значение языковые модели приобретают и в процессах сопровождения программного обеспечения. Одним из проявлений этого является интеллектуальный рефакторинг программного кода:

$$Code' = LLM(Code, TargetArchitecture).$$

В данном случае LLM выполняет преобразование существующей реализации с целью повышения читаемости, модульности, сопровождаемости, масштабируемости или адаптации системы к новым архитектурным требованиям. Для распределённых вычислительных платформ это особенно актуально, поскольку такие системы постоянно эволюционируют, изменяют структуру взаимодействия компонентов и переходят к новым моделям организации вычислений.

Использование LLM позволяет поддерживать реструктуризацию кода, адаптацию к микросервисной архитектуре, преобразование legacy-компонентов, оптимизацию API-взаимодействий, модернизацию cloud-native инфраструктуры и выявление потенциальных архитектурных проблем. Кроме того, языковые модели могут обнаруживать дублирование функциональности, неявные зависимости между компонентами и неэффективные схемы взаимодействия распределённых сервисов.

Отдельное направление связано с автоматизированным формированием и актуализацией технической документации:

$$Doc = LLM(Code, Config, Architecture, RuntimeState).$$

Поддержание актуальной документации традиционно является одной из наиболее сложных задач сопровождения распределённых информационных систем. В условиях постоянного изменения архитектуры, конфигураций и инфраструктурных компонентов документация быстро теряет актуальность, что затрудняет эксплуатацию, аудит, развитие и восстановление системы после отказов.

LLM позволяют формировать и обновлять технические описания, API-документацию, эксплуатационные инструкции, комментарии к программному коду, архитектурные обзоры, описания конфигураций и инфраструктурные регламенты. При этом документация может генерироваться на основе анализа исходного кода, конфигурационных файлов, сценариев развертывания, логов изменений и фактического состояния распределённой вычислительной среды. Это повышает согласованность между реальной системой и её описанием.

Особую роль LLM могут играть в DevOps- и infrastructure-as-code процессах. В распределённых вычислительных системах значительная часть управления инфраструктурой реализуется через конфигурационные

файлы, сценарии развертывания, CI/CD-конвейеры, политики масштабирования, мониторинг и автоматизированные процедуры восстановления. LLM-агенты могут анализировать такие артефакты, выявлять противоречия, предлагать изменения, формировать сценарии развертывания и помогать оператору интерпретировать последствия конфигурационных решений.

Таким образом, LLM в разработке информационных систем выполняют не только функцию генерации кода, но и функцию интеллектуального сопровождения жизненного цикла распределённой системы. Они связывают проектирование, реализацию, тестирование, эксплуатацию, мониторинг и документирование в единую инженерную среду.

При этом необходимо учитывать ограничения такого подхода. Код, тесты, конфигурации и документация, сформированные LLM, должны проходить проверку, ревью, тестирование и валидацию. Языковая модель может генерировать синтаксически корректные, но архитектурно неудачные или небезопасные решения. Поэтому LLM-агенты в инженерном процессе должны использоваться как ассистирующий и когнитивный компонент, а не как полностью автономный источник изменений без контроля.

В контексте настоящей работы использование LLM в разработке информационных систем важно потому, что процессы разработки, сопровождения и управления распределёнными вычислительными системами постепенно объединяются в единую интеллектуальную среду. LLM-агенты могут участвовать как в runtime-управлении ГРВС, так и в инженерной эволюции самой управляющей и вычислительной инфраструктуры. Это создаёт основу для перехода к инженерии программного обеспечения, управляемой методами искусственного интеллекта, и автономным AI-разработчикам, способным поддерживать развитие сложных распределённых платформ в связке с методами адаптивного управления и мультиагентной координации.

8.8. LLM в адаптивном управлении

Интеграция больших языковых моделей в контур адаптивного управления приводит к формированию гибридных интеллектуальных архитектур, объединяющих формальные методы оптимизации, модели машинного обучения, мультиагентную координацию и когнитивные механизмы рассуждения на основе LLM [18–21]. Такой подход является естественным развитием концепции интеллектуального управления, рассмотренной в главе 7.

В традиционных системах адаптивного управления основное внимание уделяется обработке числовых параметров среды, прогнозированию состояния системы и оптимизации распределения ресурсов. Однако по мере усложнения распределённых вычислительных платформ возрастает значение задач, связанных с интерпретацией контекста, согласованием целей, анализом неоднозначных ситуаций, объяснением решений и взаимодействием с человеком-оператором.

В этих условиях перспективной становится гибридная модель управления, в которой различные интеллектуальные компоненты выполняют взаимодополняющие функции:

$$X^*(t) = \text{Validate}(\text{Combine}(X_{opt}(t), Z_{ML}(t), Z_{LLM}(t)), G, \text{Policies}, \text{Goals}).$$

Итоговое управляющее воздействие формируется после проверки ограничений, политик безопасности и согласования с формальными алгоритмами управления.

Данная модель отражает принципиально важную особенность современных интеллектуальных систем управления: большие языковые модели не заменяют методы оптимизации и машинного обучения, а дополняют их, формируя когнитивный уровень обработки информации и принятия решений. Поэтому $X_{LLM}(t)$ не следует понимать только как прямое управляющее воздействие. Во многих случаях результатом работы LLM является стратегия, объяснение, план, рекомендация, уточнение целей или выбор метода оптимизации, который затем проходит проверку допустимости и реализуется специализированными агентами.

В такой архитектуре методы оптимизации обеспечивают формальное решение задач распределения ресурсов с учётом ограничений:

$$g_i(X, S(t)) \leq b_i,$$

модели машинного обучения отвечают за прогнозирование и адаптацию к динамике среды:

$$S^{\wedge}(t + 1) = ML(S(t), H(t)),$$

а LLM-компоненты выполняют интерпретацию контекста, координацию агентов и формирование высокоуровневых стратегий управления:

$$Z_{LLM}(t) = LLM(\text{Context}(t), \text{Goals}, \text{Policies}),$$

где $Z_{LLM}(t)$ — стратегия, объяснение, план или рекомендация.

Таким образом, адаптивное управление приобретает многоуровневую структуру. На нижнем уровне специализированные алгоритмы и агенты выполняют мониторинг, прогнозирование, оптимизацию и исполнение действий. На когнитивном уровне LLM-агент анализирует контекст, сопоставляет результаты разных компонентов, выявляет противоречия, формирует стратегию и объясняет принятое решение.

Одной из ключевых ролей LLM в адаптивном управлении является поддержка высокоуровневого выбора стратегии. В отличие от

специализированных ML-моделей, ориентированных на локальные зависимости между параметрами системы, языковые модели способны анализировать глобальный контекст функционирования вычислительной среды, учитывать историю взаимодействий, интерпретировать ограничения и работать с целями, заданными на естественном языке.

Это особенно важно в ситуациях, когда ограничения конфликтуют между собой, система функционирует в условиях неполной наблюдаемости, требуется координация большого числа агентов, необходимо учитывать организационные политики и требования безопасности, а строгая формальная модель среды не описывает всех значимых факторов. В подобных условиях LLM выступает как когнитивный уровень адаптивного управления, обеспечивающий семантическую интерпретацию состояния системы и согласование локальных решений.

Не менее важной функцией является работа со сложным неоднородным контекстом. Современные ГРВС включают множество взаимосвязанных сервисов, потоков данных, вычислительных узлов, механизмов оркестрации и эксплуатационных политик. Многие зависимости между компонентами системы трудно выразить только через аналитические модели или числовые признаки.

LLM позволяют объединять в едином контексте метрики, логи, трассировки, архитектурные описания, сообщения агентов, эксплуатационную документацию, политики управления и историю предыдущих решений. Благодаря этому языковые модели могут выявлять возможные причинно-следственные связи и формировать более целостное представление о состоянии распределённой вычислительной среды.

Особое значение LLM приобретают в задачах объяснения решений. Одним из фундаментальных ограничений классических интеллектуальных методов является низкая интерпретируемость. Нейросетевые модели и reinforcement learning-системы могут формировать эффективные решения, но их причины часто трудно объяснить оператору или разработчику системы.

Использование LLM позволяет формировать человеко-понятные описания управляющих воздействий:

$$Explanation(t) = LLM(X(t), S(t), Context(t), Constraints).$$

В результате система способна объяснять причины перераспределения ресурсов, основания изменения стратегии управления, факторы, повлиявшие на выбор маршрутов, причины масштабирования сервисов и ожидаемые последствия управляющих воздействий. Это особенно важно для критически значимых распределённых вычислительных инфраструктур, где требуется аудит решений, контроль

корректности управления, участие человека в контуре принятия решений и анализ поведения интеллектуальной системы.

Следует отметить, что LLM-компонент не должен использоваться как единственный источник управляющих воздействий. Решения, планы или рекомендации, сформированные LLM, должны проверяться с использованием формальных ограничений, механизмов безопасности, процедур валидации и специализированных алгоритмов оптимизации. В этом смысле LLM выполняет роль когнитивного координатора, а не неконтролируемого автономного управляющего центра.

Таким образом, интеграция LLM в адаптивное управление приводит к появлению нового класса интеллектуальных систем управления ГРВС. В таких системах формальные методы оптимизации и ML-модели дополняются когнитивным уровнем рассуждения, планирования, объяснения и координации. Это создаёт основу для построения автономных, но контролируемых интеллектуальных распределённых систем управления вычислительной инфраструктурой нового поколения.

8.9. Ограничения LLM-агентов

Несмотря на значительный потенциал LLM-агентов, их применение в задачах адаптивного управления гетерогенными распределёнными вычислительными системами связано с рядом принципиальных ограничений [18–21]. Эти ограничения особенно важны потому, что в контуре управления вычислительной инфраструктурой ошибка рассуждения или некорректное действие агента может привести не только к неверному ответу, но и к деградации производительности, нарушению SLA, перегрузке ресурсов или отказу компонентов системы.

Первое ограничение связано с недетерминированностью поведения больших языковых моделей. При одинаковом или близком входном контексте LLM может формировать различающиеся ответы, планы действий или объяснения. Для задач управления вычислительной инфраструктурой это обстоятельство имеет существенное значение, поскольку управляющие решения должны быть воспроизводимыми, проверяемыми и согласованными с заданными ограничениями.

Второе ограничение определяется зависимостью результата от prompt и контекста. Качество решений LLM-агента существенно зависит от способа представления состояния системы, формулировки целей, полноты описания ограничений, структуры входного сообщения и состава доступной памяти. Некорректно сформированный контекст может привести к неверной интерпретации состояния среды, выбору неадекватной стратегии или генерации недопустимого плана действий.

Третьим ограничением является риск генерации недостоверных утверждений, ошибочных гипотез или внешне убедительных, но фактически неверных объяснений. Для LLM-агентов это особенно опасно в задачах анализа инцидентов и объяснения управляющих воздействий. Языковая модель может предложить логически связное объяснение деградации производительности, которое не подтверждается фактическими метриками, логами или трассировками. Поэтому выводы LLM должны проверяться по данным мониторинга, результатам инструментов диагностики и формальным ограничениям системы.

Четвёртое ограничение связано с высокой вычислительной стоимостью. Использование LLM требует значительных ресурсов на выполнение запросов, хранение и обработку контекста, работу с памятью, вызов внешних инструментов и интеграцию с системами мониторинга или управления инфраструктурой. В контуре адаптивного управления это может увеличивать задержки принятия решений, особенно если LLM используется не как аналитический или координирующий уровень, а как компонент оперативного управления.

Пятое ограничение связано с отсутствием гарантий оптимальности. LLM-агент способен рассуждать, обобщать контекст и предлагать стратегии, однако сам по себе не обеспечивает строгого решения оптимизационной задачи:

$$\min F(X, S(t))$$

при ограничениях

$$g_i(X, S(t)) \leq b_i.$$

Поэтому решения, сформированные LLM, должны рассматриваться как рекомендации, гипотезы или высокоуровневые планы, требующие проверки формальными методами оптимизации, правилами допустимости и механизмами валидации.

Шестое ограничение связано с безопасностью доступа к инструментам и API управления инфраструктурой. LLM-агент, имеющий возможность вызывать внешние инструменты, изменять конфигурацию, запускать процедуры масштабирования или инициировать миграцию задач, становится потенциально опасным компонентом управляющего контура. Ошибка интерпретации команды, неверное рассуждение или некорректный вызов инструмента могут привести к нежелательным изменениям в вычислительной среде.

Поэтому инструментальные действия LLM-агентов должны выполняться в контролируемом режиме. Необходимы разграничение прав доступа, проверка допустимости команд, журналирование действий, предварительная симуляция критических изменений, ограничение области воздействия, механизмы отката и подтверждение оператора для операций, способных существенно изменить состояние инфраструктуры.

Седьмое ограничение связано с устойчивостью мультиагентного взаимодействия. В LLM-MAS несколько языковых агентов могут обмениваться гипотезами, критикой, планами и объяснениями. При отсутствии строгих правил координации возможно накопление ошибок, усиление неверных предположений, возникновение противоречивых планов или избыточных итераций согласования. Поэтому мультиагентные LLM-системы требуют контроля коллективного поведения, фиксированных ролей, процедур валидации и механизма остановки итеративного рассуждения.

Следовательно, LLM-агенты не должны использоваться как единственный источник управляющих решений в критически важных контурах управления. Их рациональная роль состоит в выполнении функций анализа контекста, объяснения, координации, генерации вариантов решений, поддержки оператора и выбора высокоуровневых стратегий. Непосредственное применение управляющих воздействий должно сопровождаться проверкой ограничений, контролем безопасности, механизмами обратной связи и, при необходимости, участием человека в контуре принятия решений.

Близкий подход к обеспечению управляемости ИИ-агента в организационно-производственном процессе рассмотрен в работе [120], где предложен алгоритм управления качеством изменений с участием ИИ-агента, основанный на фиксации действий агента, контрольных точках качества, проверке фактического состояния процесса и принятии управленческого решения. Рассмотренный подход согласуется с необходимостью включения LLM-агентов в контролируемую архитектуру с журналированием, валидацией, проверкой ограничений и участием оператора при критически значимых воздействиях.

Таким образом, LLM-агенты расширяют возможности адаптивного и мультиагентного управления ГРВС, но требуют осторожной интеграции в архитектуру системы. Наиболее обоснованным является гибридный подход, при котором LLM-компоненты дополняют формальные методы оптимизации, ML-модели и rule-based механизмы, не подменяя собой строгую проверку корректности, допустимости и безопасности управляющих решений.

8.10. Ключевой научный результат

Проведённый анализ показывает, что интеграция больших языковых моделей в мультиагентные системы управления приводит к качественному изменению архитектуры интеллектуального управления гетерогенными распределёнными вычислительными системами [18–21].

Классические подходы к управлению ГРВС, включая методы оптимизации, машинного обучения, нейросетевые модели и reinforcement

learning, ориентированы преимущественно на алгоритмическую обработку числовых данных, прогнозирование состояния среды и поиск решений в формализованном пространстве состояний и ограничений:

$$X^*(t) = A(\hat{S}(t)).$$

Такие методы обеспечивают высокую эффективность при решении задач распределения ресурсов, балансировки нагрузки, прогнозирования и оптимизации. Однако они ограничены в задачах, требующих интерпретации сложного контекста, логического рассуждения, объяснения решений, анализа неоднородной информации, работы с текстовыми источниками, учёта эксплуатационных политик и координации действий множества агентов.

Интеграция LLM-агентов позволяет перейти к новому классу интеллектуальных систем, в которых управление осуществляется не только на алгоритмическом, но и на когнитивном уровне:

$$X^*(t) = \textit{Decision}(\hat{S}(t), \textit{Context}(t), \textit{MAS}, \textit{AI}, \textit{LLM}, \textit{G}, \textit{Policies}, \textit{Goals}).$$

В данной модели принятие решений основывается не только на числовой оптимизации, но и на комплексном анализе контекста функционирования системы, накопленного опыта, целей управления, политик безопасности, сообщений агентов, логов, документации и семантических связей между событиями распределённой вычислительной среды.

Принципиально важным является то, что LLM-агенты обеспечивают функции, которые слабо представлены в традиционных интеллектуальных методах управления: интерпретацию состояния системы, формирование гипотез, логическое рассуждение, контекстное планирование, объяснение решений, координацию мультиагентного взаимодействия, взаимодействие с человеком-оператором и интеграцию с внешними инструментами.

Тем самым система управления получает способность не только вычислять управляющее воздействие, но и анализировать причины происходящих процессов, сопоставлять альтернативные стратегии, объяснять выбор действий, учитывать высокоуровневые цели и адаптировать поведение с учётом контекста эксплуатации.

В результате происходит переход от локальной оптимизации к когнитивной координации, от реактивного управления к рассуждающему управлению, от отдельных специализированных моделей к агентным системам, способным объединять числовой анализ, семантическую интерпретацию, планирование и объяснимое принятие решений.

При этом ключевым условием практического применения LLM-агентов является их включение в гибридную и контролируемую архитектуру. LLM-компонент не должен подменять формальную оптимизацию, проверку ограничений и механизмы безопасности. Его

рациональная роль состоит в выполнении функций когнитивного слоя: анализа контекста, генерации гипотез, выбора стратегии, объяснения решений и координации агентов. Непосредственное применение управляющих воздействий должно проходить через проверку допустимости, формальные ограничения, rule-based механизмы, специализированные агенты исполнения и, при необходимости, human-in-the-loop.

Таким образом, ключевым научным результатом главы является обоснование перехода от алгоритмического интеллектуального управления распределёнными вычислительными системами к когнитивному управлению, в котором LLM-агенты выступают как интеллектуальный слой анализа, рассуждения, планирования, объяснения и координации в адаптивной мультиагентной вычислительной среде.

Выводы по главе 8

В главе рассмотрены большие языковые модели и LLM-агенты как новый класс интеллектуальных компонентов адаптивного управления гетерогенными распределёнными вычислительными системами. Показано, что развитие LLM приводит к переходу от преимущественно алгоритмических методов управления к когнитивным мультиагентным архитектурам, способным выполнять семантический анализ, логическое рассуждение, планирование, координацию и объяснение решений [18–21].

Установлено, что классические интеллектуальные методы, основанные на машинном обучении, нейросетевых моделях, оптимизации и reinforcement learning, эффективны при обработке числовых данных, прогнозировании состояния среды и решении формализованных задач распределения ресурсов. Однако они ограничены в задачах интерпретации сложного контекста, работы с неоднородной информацией, объяснения решений и высокоуровневой координации мультиагентного взаимодействия.

Введена модель LLM-агента:

$$a_{LLM} = \langle Obs, Reason, Plan, Act, Memory, Tools \rangle,$$

в которой выделены функции наблюдения, рассуждения, планирования, исполнения действий, памяти и инструментального взаимодействия. Показано, что LLM-агент представляет собой не просто языковую модель, а управляющий интеллектуальный компонент, включаемый в контур адаптивного мультиагентного управления.

Рассмотрена архитектура LLM-агента и формализован процесс его функционирования:

$$X^*(t) = Act(Plan(Reason(Obs(S(t))), Memory, Tools))).$$

Установлено, что работа такого агента включает преобразование состояния вычислительной среды в контекст, семантический анализ этого

контекста, формирование плана действий, обращение к внешним инструментам, передачу действий на исполнение или формирование рекомендации, а также объяснение принятого решения.

Показано, что интеграция LLM-агентов в мультиагентную систему управления не заменяет существующих агентов мониторинга, прогнозирования, оптимизации, координации и исполнения, а расширяет их за счёт когнитивного слоя:

$$MAS_{LLM} = \langle A, a_{LLM}, Comm, Tools, Policies, Human \rangle.$$

Такой слой обеспечивает анализ контекста, интерпретацию результатов прогнозирования, выбор стратегии оптимизации, согласование локальных решений и взаимодействие с человеком-оператором.

Определены основные роли LLM-агентов в управлении ГРБС: агент анализа состояния, агент поддержки принятия решений, агент координации, агент объяснения и агент взаимодействия с оператором. Показано, что данные роли обеспечивают переход от обработки отдельных метрик к анализу целостного контекста функционирования распределённой вычислительной среды.

Рассмотрены мультиагентные LLM-системы:

$$LLM_MAS = \{a_{LLM}^1, a_{LLM}^2, \dots, a_{LLM}^k\},$$

в которых специализированные LLM-агенты совместно выполняют функции планирования, критики, уточнения, исполнения и обучения. Показано, что цикл

$$Plan \rightarrow Critique \rightarrow Refine \rightarrow Execute \rightarrow Learn$$

позволяет повысить контролируемость решений, организовать внутреннюю проверку действий и обеспечить накопление опыта управления.

Проанализировано применение LLM в разработке, сопровождении и эксплуатации распределённых информационных систем. Показано, что языковые модели могут использоваться для генерации кода, формирования тестов, рефакторинга, актуализации документации, DevOps-автоматизации и работы с инфраструктурой как кодом. Это позволяет рассматривать LLM не только как инструмент runtime-управления, но и как компонент жизненного цикла распределённых вычислительных систем.

Рассмотрена интеграция LLM в контур адаптивного управления:

$$X(t) = \alpha X_{opt}(t) + \beta X_{ML}(t) + \gamma X_{LLM}(t).$$

Показано, что в такой архитектуре методы оптимизации обеспечивают формальное решение задач распределения ресурсов, ML-модели выполняют прогнозирование и адаптацию к динамике среды, а LLM-компоненты реализуют когнитивные функции анализа контекста, планирования, объяснения и координации.

Проанализированы ограничения LLM-агентов: недетерминированность поведения, зависимость от prompt и контекста, риск генерации недостоверных объяснений, высокая вычислительная стоимость, отсутствие гарантий оптимальности, риски безопасности при доступе к инструментам и сложность устойчивой координации в LLM-

MAS. Установлено, что эффективное применение LLM требует гибридной архитектуры, формальных ограничений, процедур валидации, контроля действий, журналирования и, при необходимости, участия человека в контуре принятия решений.

Ключевым результатом главы является обоснование перехода от алгоритмического управления к когнитивному управлению распределёнными вычислительными системами. Использование LLM-агентов позволяет дополнить адаптивное мультиагентное управление функциями семантического анализа, рассуждения, планирования, объяснения решений и координации распределённых процессов.

Таким образом, результаты главы формируют концептуальную основу интеллектуальной архитектуры управления ГРВС нового поколения, объединяющей адаптивное управление, мультиагентные системы, методы машинного обучения, формальную оптимизацию и LLM-агентов в единую когнитивную среду управления распределённой вычислительной инфраструктурой.

ГЛАВА 9. АРХИТЕКТУРА АДАПТИВНОЙ СИСТЕМЫ УПРАВЛЕНИЯ ГЕТЕРОГЕННЫМИ РАСПРЕДЕЛЁННЫМИ ВЫЧИСЛИТЕЛЬНЫМИ СИСТЕМАМИ

9.1. Общая концепция архитектуры

Результаты, полученные в главах 2–8, позволяют перейти от рассмотрения отдельных моделей и методов к построению целостной архитектуры адаптивной системы управления гетерогенными распределёнными вычислительными системами. В предыдущих главах были последовательно рассмотрены формальная модель ГРВС, контур адаптивного управления, мультиагентная декомпозиция, методы распределения ресурсов, процессы наблюдаемости, интеллектуальные методы управления и LLM-агенты как когнитивный уровень системы [12–21].

Современная ГРВС представляет собой высокодинамичную вычислительную среду, в которой состояние ресурсов, структура нагрузки, сетевые характеристики, ограничения размещения и требования качества обслуживания изменяются во времени. В таких условиях управление не может быть сведено к статическому алгоритму распределения ресурсов или единому централизованному управляющему компоненту. Необходима архитектура, способная объединить наблюдение за состоянием среды, распределённое принятие решений, интеллектуальный анализ данных, адаптацию параметров управления и координацию действий множества компонентов.

Предлагаемая архитектура строится как интеграция нескольких взаимосвязанных уровней и механизмов. Формальная модель ГРВС задаёт объект управления и систему ограничений. Подсистема наблюдаемости формирует информационную основу управления на базе метрик, логов, трассировок и потоковых данных. Мультиагентная система обеспечивает распределение функций управления между специализированными агентами. Адаптивный контур связывает наблюдение, анализ, принятие решений и исполнение управляющих воздействий. Интеллектуальные модели используются для прогнозирования, оптимизации и адаптации поведения системы. LLM-агенты формируют верхний когнитивный слой, обеспечивающий семантический анализ, рассуждение, объяснение решений и высокоуровневую координацию.

С учётом этого архитектура адаптивной системы управления может быть представлена следующим кортежем:

$$ARCH = \langle E, D, MAS, AI, LLM, C \rangle,$$

где E — управляемая гетерогенная распределённая вычислительная среда; D — подсистема данных и наблюдаемости; MAS — мультиагентная система управления; AI — совокупность интеллектуальных моделей и

алгоритмов управления; *LLM* — когнитивный слой LLM-агентов; *C* — контур адаптивного управления и координации компонентов архитектуры.

Компонент

E

представляет управляемую распределённую вычислительную среду. Он включает вычислительные узлы, сетевую инфраструктуру, потоки данных, очереди задач, сервисы распределённой обработки, облачные, гибридные и edge-компоненты. Именно данный компонент является объектом управления, состояние которого изменяется под воздействием нагрузки, отказов, внешних условий и управляющих решений.

Компонент

D

формирует data-driven уровень архитектуры. Он включает механизмы сбора, фильтрации, агрегации, синхронизации и хранения информации о состоянии системы. На данном уровне наблюдаемые данные преобразуются в оценку состояния вычислительной среды:

$$\hat{S}(t) = O(Y(t)),$$

где $Y(t)$ — поток наблюдаемых данных, включающий метрики, логи, трассировки, события мониторинга и телеметрию инфраструктуры, а O — функция обработки и интерпретации наблюдений.

Компонент

MAS

представляет распределённую систему взаимодействующих агентов, реализующих функции мониторинга, прогнозирования, оптимизации, координации, управления ресурсами и исполнения управляющих воздействий. Мультиагентная организация обеспечивает декомпозицию сложной задачи управления на совокупность локальных подзадач, позволяет принимать решения ближе к управляемым ресурсам и снижает зависимость системы от единого центра управления.

Компонент

AI

объединяет интеллектуальные модели и алгоритмы управления, включая методы оптимизации, машинного обучения, прогнозирования, reinforcement learning, нейросетевые архитектуры и гибридные методы. Данный компонент отвечает за анализ состояния среды, прогнозирование её динамики, выбор стратегий распределения ресурсов и адаптацию параметров управления:

$$X(t) = A(\hat{S}(t), \theta(t)),$$

где $X(t)$ — управляющее воздействие, $\hat{S}(t)$ — оценка состояния системы, а $\theta(t)$ — параметры интеллектуальной модели управления.

Компонент

LLM

формирует когнитивный уровень архитектуры. LLM-агенты не заменяют формальные методы оптимизации, ML-модели и специализированных агентов управления, а дополняют их функциями семантического анализа, логического рассуждения, планирования, объяснения решений, работы с эксплуатационным контекстом и взаимодействия с человеком-оператором. На данном уровне формируются высокоуровневые стратегии, интерпретируются результаты работы других компонентов и выполняется координация сложных сценариев управления.

Компонент

C

описывает замкнутый контур адаптивного управления, связывающий все элементы архитектуры:

$$S(t) \rightarrow Y(t) \rightarrow \hat{S}(t) \rightarrow X(t) \rightarrow S(t+1).$$

Данный контур обеспечивает непрерывное наблюдение состояния среды, обработку данных, формирование управляющих воздействий, применение решений к вычислительной инфраструктуре и получение обратной связи. За счёт этого архитектура функционирует не как статическая схема, а как динамическая система управления, способная изменять своё поведение по мере изменения состояния ГРВС.

Принципиальной особенностью предлагаемой архитектуры является её многоуровневый и гибридный характер. Нижние уровни обеспечивают взаимодействие с вычислительной инфраструктурой и сбор данных. Средние уровни реализуют распределённое мультиагентное управление и адаптивное распределение ресурсов. Верхние уровни обеспечивают интеллектуальный анализ, прогнозирование, стратегическую координацию и объяснение принимаемых решений.

Таким образом, предлагаемая архитектура объединяет формальную модель ГРВС, data-driven наблюдаемость, мультиагентное распределённое управление, интеллектуальные методы анализа и LLM-когнитивный слой в единую адаптивную систему управления. Такая система рассматривается не как фиксированный набор алгоритмов, а как динамическая интеллектуальная среда, способная наблюдать состояние распределённой инфраструктуры, адаптироваться к его изменению, обучаться на накопленных данных и координировать процессы управления в условиях высокой сложности, неопределённости и гетерогенности вычислительной среды.

9.2. Уровневая структура системы

Предлагаемая архитектура адаптивной системы управления гетерогенными распределёнными вычислительными системами имеет многоуровневую структуру. Каждый уровень выполняет собственную функцию в общем контуре управления: инфраструктурный уровень задаёт объект управления, уровень наблюдаемости формирует данные о состоянии среды, уровень управления реализует алгоритмы адаптации, агентный уровень обеспечивает распределённое выполнение функций, интеллектуальный уровень выполняет прогнозирование и когнитивный анализ, а уровень принятия решений формирует итоговое управляющее воздействие [12–21].

Такое разделение необходимо для того, чтобы избежать смешения разнородных процессов: сбора данных, анализа состояния, локального управления, глобальной координации, интеллектуального прогнозирования и исполнения решений. В результате архитектура рассматривается не как единый монолитный управляющий модуль, а как совокупность взаимосвязанных слоёв, образующих замкнутый адаптивный контур управления распределённой вычислительной средой.

Структура архитектуры представлена на рисунке 11.

9.2.1. Уровень инфраструктуры

Нижний уровень архитектуры образует управляемая вычислительная инфраструктура. Он включает вычислительные узлы, сетевую подсистему, хранилища данных, очереди задач, сервисы распределённой обработки, облачные, гибридные и edge-компоненты. Данный уровень является непосредственным объектом управления и определяет физическую и логическую структуру ГРВС.

Формально инфраструктурный уровень может быть описан графовой моделью:

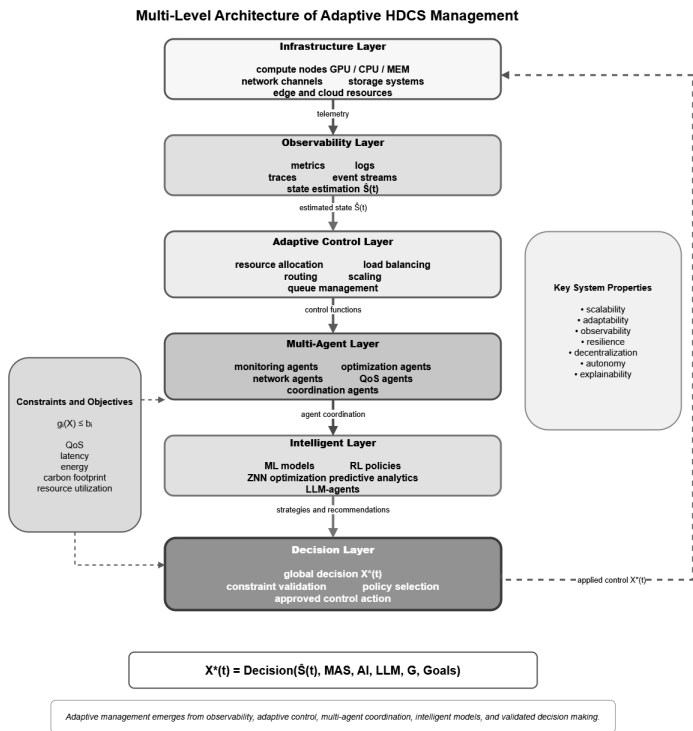
$$G_{net} = (N, L),$$

где N — множество вычислительных узлов, а L — множество каналов связи между ними.

Состояние инфраструктуры определяется характеристиками вычислительных ресурсов, параметрами сетевых соединений, текущей загрузкой узлов, доступностью памяти и ускорителей, состоянием очередей задач, потоками данных и ограничениями размещения. Именно на этом уровне фактически выполняются вычисления, передаются данные и реализуются последствия управляющих воздействий.

Особенностью инфраструктурного уровня является его гетерогенный характер. Узлы могут различаться по производительности, архитектуре, набору доступных ресурсов, сетевой близости, энергетическим характеристикам и допустимым режимам эксплуатации. Поэтому система управления должна учитывать не только наличие ресурсов, но и их

неоднородность, динамическую доступность и взаимосвязь с другими компонентами вычислительной среды.



Multi-level architecture integrating infrastructure, observability, adaptive control, agents, intelligent models, and decision making

Рисунок 11 – Многоуровневая архитектура системы управления

9.2.2. Уровень данных и наблюдаемости

Следующий уровень архитектуры связан с наблюдением состояния вычислительной среды и обработкой данных мониторинга. Его основная задача заключается в преобразовании распределённых, зашумлённых и асинхронно поступающих данных в оценку состояния системы, пригодную для последующего анализа и управления.

Формально процесс наблюдения может быть представлен следующим образом:

$$\hat{S}(t) = O(Y(t)),$$

где $Y(t)$ — поток наблюдаемых данных, а O — функция обработки, фильтрации, агрегации и интерпретации информации.

На данном уровне осуществляется сбор метрик, логов, трассировок, событий мониторинга, телеметрии инфраструктуры, сообщений агентов и данных о выполнении задач. Эти данные поступают от различных компонентов распределённой среды и могут отличаться частотой обновления, точностью, полнотой и степенью актуальности.

Уровень наблюдаемости выполняет фильтрацию шумов, агрегацию распределённых наблюдений, синхронизацию данных, восстановление пропущенной информации, выявление аномалий и формирование признакового описания состояния системы. Результатом его работы является оценка состояния:

$$\hat{S}(t),$$

которая используется агентами управления, интеллектуальными моделями и уровнем принятия решений. Тем самым данный слой формирует информационную основу всей архитектуры.

9.2.3. Уровень адаптивного управления

Уровень адаптивного управления отвечает за преобразование оценки состояния вычислительной среды в управляющие воздействия:

$$X(t) = A(\hat{S}(t)).$$

На данном уровне реализуются алгоритмы распределения ресурсов, балансировки нагрузки, управления потоками данных, маршрутизации, масштабирования вычислений, изменения приоритетов обработки задач и адаптации параметров системы.

В отличие от инфраструктурного уровня, данный слой работает не с физическими компонентами непосредственно, а с моделями и алгоритмами управления. Он определяет, какие действия должны быть выполнены для улучшения состояния среды, предотвращения перегрузок, соблюдения SLA, повышения устойчивости или оптимизации использования ресурсов.

На уровне адаптивного управления используются эвристические алгоритмы, методы оптимизации, алгоритмы распределения задач, механизмы управления очередями, прогнозные модели и правила адаптивного изменения параметров. Этот уровень формирует операционную основу управления, связывая данные наблюдаемости с конкретными действиями, применяемыми к вычислительной инфраструктуре.

9.2.4. Агентный уровень

Агентный уровень задаёт организационный механизм распределённого управления. Если уровень адаптивного управления определяет функции и алгоритмы, то агентный уровень определяет, какими специализированными компонентами эти функции выполняются и как они взаимодействуют между собой.

Мультиагентная система управления может быть представлена как:

$$MAS = \{a_1, \dots, a_p\},$$

где a_i — специализированный агент, отвечающий за определённую область состояния системы, класс ограничений или тип управляющих воздействий.

Агентный уровень обеспечивает локальность принятия решений, параллельную обработку информации, распределение ответственности, координацию локальных действий, устойчивость к отказам отдельных компонентов и масштабируемость системы управления. Каждый агент может иметь собственную область наблюдения, локальную модель состояния, набор допустимых действий и механизм взаимодействия с другими агентами.

В рамках данного уровня могут функционировать агенты вычислительных ресурсов, сетевые агенты, агенты качества обслуживания, агенты размещения, агенты прогнозирования, агенты оптимизации, агенты координации и агенты исполнения. Их совместная работа позволяет избежать перегрузки единого управляющего центра и обеспечить управление крупномасштабной распределённой средой на основе локальных и глобальных решений.

9.2.5. Интеллектуальный и когнитивный уровень

Интеллектуальный уровень объединяет методы машинного обучения, reinforcement learning, нейросетевые модели, гибридные алгоритмы и LLM-агентов. Его задача заключается в повышении качества анализа состояния среды, прогнозирования, выбора стратегий управления и объяснения принимаемых решений.

На данном уровне выполняются прогнозирование нагрузки, выявление аномалий, интеллектуальная оптимизация, оценка риска перегрузок, адаптация параметров моделей, анализ контекста, логическое рассуждение, координация агентов и объяснение решений.

Особенностью интеллектуального уровня является использование обучаемых моделей, параметры которых могут изменяться в процессе функционирования системы:

$$\hat{\theta}(t+1) = L(\theta(t), S(t), X(t), S(t+1)).$$

Это позволяет системе учитывать накопленный опыт, изменять стратегии распределения ресурсов и адаптироваться к новым режимам работы вычислительной среды.

В составе данного уровня LLM-компоненты формируют когнитивный слой архитектуры. Они обеспечивают семантический анализ логов, событий, политик управления, сообщений агентов и эксплуатационного контекста. В отличие от ML-моделей, ориентированных преимущественно на числовые данные, LLM-агенты позволяют работать с неоднородной информацией, формировать объяснения, поддерживать взаимодействие с оператором и выполнять высокоуровневую координацию сложных сценариев управления.

9.2.6. Уровень принятия решений

Верхний уровень архитектуры отвечает за формирование итогового управляющего решения на основе данных, прогнозов, локальных решений агентов, интеллектуальных рекомендаций, ограничений и глобальных целей управления.

Итоговое решение может быть представлено следующим образом:

$$X^*(t) = \underset{\wedge}{\text{Decision}}(S(t), MAS, AI, LLM, G, Goals),$$

где G — система ограничений, а $Goals$ — множество целей управления.

В частном случае задача принятия решений может быть сведена к оптимизационной постановке:

$$X^*(t) = \underset{\wedge}{\text{argmin}}_X F(X, S(t))$$

при ограничениях

$$g_i(X, S(t)) \leq b_i, g_i \in G.$$

На данном уровне осуществляется согласование различных критериев управления: производительности, загрузки ресурсов, сетевых задержек, энергопотребления, устойчивости, стоимости вычислений и соблюдения SLA. Здесь же выполняется проверка допустимости решений, разрешение конфликтов между локальными действиями агентов и выбор итогового управляющего воздействия.

В отличие от нижних уровней, уровень принятия решений ориентирован не на управление отдельным узлом или подсистемой, а на достижение глобальных целей функционирования ГРВС. Его задача состоит в том, чтобы объединить результаты всех уровней архитектуры в согласованное решение, которое может быть применено к вычислительной среде.

Таким образом, многоуровневая архитектура обеспечивает разделение функций между инфраструктурой, подсистемой наблюдаемости, адаптивным управлением, агентной организацией, интеллектуальными моделями и уровнем принятия решений. Такое разделение позволяет интегрировать процессы мониторинга, анализа, прогнозирования, координации и исполнения управляющих воздействий в единую адаптивную систему управления гетерогенной распределённой вычислительной средой.

9.3. Контур управления в архитектуре

Предложенная многоуровневая архитектура формирует единый замкнутый контур адаптивного управления гетерогенной распределённой вычислительной средой. В отличие от централизованных схем, где наблюдение, анализ и принятие решений сосредоточены в одном управляющем компоненте, в рассматриваемой архитектуре эти функции

распределены между подсистемой наблюдаемости, мультиагентной системой, интеллектуальными моделями, LLM-агентами и уровнем принятия решений [12–21].

Контур управления может быть представлен как последовательность преобразований:

$$S(t) \rightarrow Y(t) \rightarrow \hat{S}(t) \rightarrow MAS \rightarrow AI/LLM \rightarrow X^*(t) \rightarrow S(t+1),$$

где $S(t)$ — фактическое состояние вычислительной среды в момент времени t ; $Y(t)$ — поток наблюдаемых данных; $\hat{S}(t)$ — оценка состояния системы; MAS — мультиагентная система управления; AI/LLM — интеллектуальные и когнитивные компоненты анализа и принятия решений; $X^*(t)$ — согласованное управляющее воздействие; $S(t+1)$ — новое состояние системы после применения управления.

Данная модель отражает не линейную однократную процедуру, а непрерывный циклический процесс. Каждое управляющее воздействие изменяет состояние вычислительной среды, после чего обновлённое состояние вновь наблюдается, анализируется и используется для формирования следующих решений.

На первом этапе контур управления начинается с фактического состояния распределённой вычислительной среды:

$$S(t).$$

Данное состояние включает загрузку вычислительных узлов, состояние памяти и ускорителей, параметры сетевых каналов, очереди задач, потоки данных, доступность ресурсов, ограничения размещения, состояние сервисов и показатели качества обслуживания.

Поскольку полное состояние ГРВС непосредственно недоступно управляющей системе, оно наблюдается через поток данных:

$$Y(t) = Obs(S(t)).$$

Поток наблюдений включает метрики, события, логи, трассировки, телеметрию инфраструктуры, сообщения агентов и результаты выполнения управляющих действий. Эти данные поступают от распределённых источников, могут быть неполными, зашумлёнными и задержанными, поэтому требуют предварительной обработки.

На следующем этапе подсистема данных и наблюдаемости формирует оценку состояния вычислительной среды:

$$\hat{S}(t) = O(Y(t)),$$

где O — функция фильтрации, агрегации, синхронизации и интерпретации наблюдаемых данных. Полученная оценка состояния $\hat{S}(t)$ используется всеми последующими компонентами архитектуры: агентами управления, прогнозными моделями, оптимизационными алгоритмами и уровнем принятия решений.

Далее оценка состояния поступает в мультиагентную систему управления:

$$MAS = \{a_1, \dots, a_p\}.$$

На данном этапе специализированные агенты выполняют локальный анализ состояния среды, контролируют отдельные классы ограничений, формируют локальные решения и обмениваются информацией с другими агентами. Compute Agent анализирует загрузку вычислительных ресурсов, Network Agent оценивает состояние каналов связи, QoS Agent контролирует временные ограничения, Prediction Agent формирует прогноз состояния среды, Optimization Agent подготавливает варианты распределения ресурсов, а Coordination Agent согласует локальные решения.

Мультиагентная часть контура обеспечивает декомпозицию задачи управления. Вместо единого центра, анализирующего всё пространство состояний и ограничений, система распределяет функции между агентами:

$$X_i(t) = a_i(\hat{S}_i(t)),$$

где $\hat{S}_i(t)$ — локальная оценка состояния, доступная агенту a_i , а $X_i(t)$ — локальное управляющее решение или рекомендация.

Локальные решения агентов передаются на уровень интеллектуального анализа и согласования. На данном этапе используются методы оптимизации, машинного обучения, прогнозирования, reinforcement learning, нейросетевые модели, гибридные алгоритмы и LLM-агенты. Их задача состоит в том, чтобы преобразовать множество локальных наблюдений, прогнозов и рекомендаций в согласованное управляющее воздействие:

$$X^*(t) = Decision(X_1(t), \dots, X_p(t), \hat{S}(t), AI, LLM, G, Goals).$$

где G — система ограничений, а $Goals$ — цели управления.

Интеллектуальные модели обеспечивают прогнозирование изменения нагрузки, оценку риска перегрузок, выбор стратегии распределения ресурсов, адаптацию параметров управления и поиск компромиссных решений. LLM-агенты дополняют этот процесс когнитивными функциями: интерпретацией сложного контекста, объяснением решений, анализом сообщений агентов, согласованием высокоуровневых целей и формированием стратегий управления.

Особенностью предлагаемого контура является то, что решение формируется не только на основе текущей оценки состояния среды, но и с учётом истории функционирования системы, прогнозируемых изменений нагрузки, ограничений инфраструктуры, глобальных целей управления, политик безопасности, SLA и результатов предыдущих управляющих воздействий:

$$X^*(t) = A(\hat{S}(t), H(t), \theta(t), G, Goals).$$

Здесь $H(t)$ — история наблюдений и действий, а $\theta(t)$ — параметры интеллектуальных моделей управления.

После формирования и проверки управляющего воздействия оно применяется к вычислительной среде:

$$S(t+1) = f(S, X^*(t)).$$

Управляющее воздействие может включать перераспределение задач, изменение маршрутов передачи данных, масштабирование ресурсов, миграцию сервисов, изменение приоритетов обработки, запуск восстановительных процедур или корректировку конфигурации системы.

Результаты применения управления вновь поступают в подсистему наблюдаемости, замыкая контур:

$$S(t+1) \rightarrow Y(t+1) \rightarrow \hat{S}(t+1).$$

Именно наличие обратной связи позволяет архитектуре адаптироваться к изменению состояния среды, оценивать последствия принятых решений и корректировать дальнейшие управляющие воздействия.

Принципиально важным является то, что в рассматриваемой архитектуре адаптируются не только управляющие воздействия, но и параметры моделей управления:

$$\theta(t+1) = L(\theta(t), \hat{S}(t), X^*(t), S(t+1)).$$

Это означает, что система способна изменять стратегии распределения ресурсов, уточнять прогнозные модели, корректировать параметры агентов и повышать качество управления по мере накопления опыта.

Таким образом, контур управления в предлагаемой архитектуре имеет одновременно адаптивный, мультиагентный и когнитивный характер. Адаптивность обеспечивается обратной связью и изменением управляющих воздействий во времени. Мультиагентность обеспечивает распределённость анализа и локальность принятия решений. Интеллектуальные модели обеспечивают прогнозирование и оптимизацию. LLM-агенты добавляют семантический анализ, объяснение, планирование и высокоуровневую координацию.

В результате предложенный контур объединяет наблюдение, обработку данных, мультиагентное взаимодействие, интеллектуальный анализ, когнитивную координацию и исполнение решений в единую архитектуру адаптивного управления гетерогенной распределённой вычислительной средой.

9.4. Агентная структура системы

Ключевым элементом предлагаемой архитектуры является мультиагентная организация системы управления. Именно агентный уровень обеспечивает переход от централизованной схемы принятия решений к распределённой архитектуре, в которой функции управления декомпозируются между специализированными компонентами, работающими на разных уровнях вычислительной среды [12–21].

Необходимость такой организации определяется высокой сложностью ГРВС, распределённым характером ресурсов, изменяющимся состоянием инфраструктуры и невозможностью эффективного анализа всех параметров среды единым управляющим компонентом. В результате управление реализуется как взаимодействие множества агентов:

$$MAS = \{a_1, a_2, \dots, a_p\},$$

где каждый агент a_i отвечает за определённую область состояния системы, класс ограничений или тип управляющих воздействий.

В рамках предлагаемой архитектуры агенты целесообразно разделить на три основных класса: локальные, глобальные и интеллектуальные. Такое разделение отражает масштаб принимаемых решений и место агента в общем контуре управления.

Первый класс образуют локальные агенты, функционирующие непосредственно на уровне вычислительных узлов, сетевых сегментов, очередей задач и отдельных подсистем инфраструктуры. Их основная задача состоит в оперативной реакции на локальные изменения состояния среды. Локальные агенты работают с ограниченной областью наблюдения и принимают решения вблизи управляемых ресурсов, что уменьшает задержки управления и снижает нагрузку на глобальные компоненты архитектуры.

К локальным агентам относятся Compute Agent, Network Agent, QoS Agent и Placement Agent. Compute Agent отвечает за контроль загрузки вычислительных ресурсов, анализ доступности CPU, памяти и ускорителей, а также за локальную балансировку задач. Network Agent управляет сетевыми потоками, маршрутизацией и предотвращением перегрузки каналов связи. QoS Agent контролирует временные ограничения, SLA и приоритеты обработки задач. Placement Agent определяет допустимость размещения задач и сервисов на конкретных узлах с учётом требований совместимости, локализации данных и доступности ресурсов.

Второй класс образуют глобальные агенты, выполняющие функции координации и стратегического управления на уровне всей распределённой вычислительной среды. Их задача состоит не в оперативном управлении отдельным узлом, а в согласовании локальных

решений, формировании глобальной стратегии распределения ресурсов и предотвращении конфликтов между ограничениями.

К глобальным агентам относятся Optimization Agent, Prediction Agent и Coordination Agent. Optimization Agent формирует согласованные решения по распределению ресурсов с учётом множества критериев и ограничений. Prediction Agent оценивает будущую динамику нагрузки, состояние ресурсов и риск возникновения перегрузок. Coordination Agent согласует локальные решения агентов, разрешает конфликты между ограничениями и обеспечивает целостность поведения системы.

Третий класс образуют интеллектуальные агенты, реализующие обучаемые и когнитивные механизмы анализа состояния среды, прогнозирования, адаптации и поддержки принятия решений. В отличие от локальных и глобальных агентов, ориентированных на конкретные функции управления, интеллектуальные агенты усиливают архитектуру за счёт методов машинного обучения, нейросетевых моделей, reinforcement learning и LLM-компонентов.

ML Agent использует обучаемые модели для анализа телеметрии, прогнозирования состояния системы, обнаружения аномалий и адаптации параметров управления:

$$\theta(t+1) = L(\theta(t), S(t), X(t), S(t+1)).$$

LLM Agent формирует когнитивный уровень архитектуры. Он выполняет семантический анализ логов, сообщений агентов, эксплуатационных политик и документации, участвует в объяснении решений, высокоуровневом планировании и взаимодействии с оператором:

$$Decision = LLM(Context, Goals, Policies, History).$$

Принципиально важно, что указанные классы агентов не функционируют изолированно. Локальные агенты передают данные о состоянии своих подсистем, глобальные агенты согласуют локальные решения, интеллектуальные агенты формируют прогнозы, рекомендации и объяснения, а LLM-компоненты обеспечивают работу с контекстом и высокоуровневую координацию. В результате агентная структура системы образует распределённую интеллектуальную среду управления, способную сочетать локальную автономность, глобальную согласованность и когнитивную поддержку принятия решений.

Таким образом, агентная структура архитектуры обеспечивает организационную основу адаптивного управления ГРВС. Она распределяет функции между локальными, глобальными и интеллектуальными агентами, связывая управление ресурсами, прогнозирование, оптимизацию, координацию и когнитивный анализ в единую мультиагентную систему.

9.5. Распределение функций управления

Распределение функций управления между специализированными агентами является одним из основных механизмов построения масштабируемой адаптивной архитектуры. В отличие от централизованной схемы, где единый управляющий компонент вынужден одновременно учитывать все параметры среды и ограничения, мультиагентный подход позволяет закрепить отдельные функции за агентами, наиболее близкими к соответствующим данным, ресурсам и управляющим воздействиям [12–21].

Распределение функций непосредственно связано с формальной моделью ограничений, введенной ранее:

$$g_i(X, S(t)) \leq b_i.$$

Каждый класс ограничений или управленческих задач сопоставляется специализированному агенту, который отвечает за контроль соответствующего аспекта функционирования системы. Это позволяет декомпозировать общую задачу управления на совокупность локальных и глобальных подзадач, сохраняя при этом согласованность итогового поведения ГРВС.

Для систематизации распределения функций управления целесообразно представить соответствие между функциями, агентами, контролируемыми ограничениями и управленческими результатами в виде таблицы 9.1.

Compute Agent реализует управление вычислительными ресурсами. Он контролирует загрузку процессоров, памяти и ускорителей, оценивает допустимость назначения задач и формирует локальные решения по балансировке нагрузки:

$$\begin{aligned}\sum_i x_{ij} c_i &\leq CPU_j, \\ \sum_i x_{ij} m_i &\leq MEM_j.\end{aligned}$$

Результатом работы данного агента является поддержание допустимой загрузки вычислительных узлов и предотвращение локальных перегрузок.

Network Agent отвечает за управление сетевыми ресурсами и потоками данных. Его область ответственности связана с контролем пропускной способности каналов связи:

$$Load_{ij} \leq B_{ij}.$$

Данный агент участвует в маршрутизации потоков, выборе альтернативных путей передачи данных и предотвращении сетевых перегрузок.

QoS Agent контролирует временные ограничения и параметры качества обслуживания:

$$t_i(X) \leq \tau_i.$$

Таблица 9.1 — Распределение функций управления между агентами

Функция управления	Ответственный агент	Контролируемые ограничения	Управленческий результат
Управление вычислительными ресурсами	Compute Agent	$\sum_i x_{ij} c_i \leq CPU_j$, $\sum_i x_{ij} m_i \leq MEM_j$	локальная балансировка нагрузки, предотвращение перегрузки узлов
Управление сетевыми потоками	Network Agent	$Load_{ij} \leq B_{ij}$	маршрутизация потоков данных, снижение сетевых задержек
Контроль качества обслуживания	QoS Agent	$t_i(X) \leq \tau_i$	соблюдение SLA, приоритизация задач
Размещение задач и сервисов	Placement Agent	$X(t_i) \in S_i$	выбор допустимых узлов, соблюдение совместимости
Прогнозирование состояния среды	Prediction Agent	прогноз риска нарушения ограничений	проактивное управление нагрузкой и ресурсами
Глобальная оптимизация	Optimization Agent	$g_i(X, S(t)) \leq b_i$	согласованное распределение ресурсов
Координация решений	Coordination Agent	конфликты между g_i и g_j	разрешение противоречий между локальными решениями
Интеллектуальный анализ	ML Agent	качество прогноза и адаптации модели	выявление закономерностей, обнаружение аномалий
Когнитивная координация	LLM Agent	политики, цели, контекст, ограничения безопасности	объяснение, планирование, поддержка оператора

Он обеспечивает соблюдение SLA, формирует приоритеты обработки задач и инициирует корректирующие действия при риске нарушения допустимого времени выполнения.

Placement Agent отвечает за логические и инфраструктурные ограничения размещения:

$$X(t_i) \in S_i.$$

Его задача состоит в том, чтобы исключить недопустимые назначения задач на узлы, не удовлетворяющие требованиям совместимости, доступности данных, безопасности или программной среды.

Prediction Agent реализует прогнозирование будущего состояния вычислительной среды:

$$\hat{S}^{\wedge}(t+1) = M(\hat{S}(t), H(t)).$$

Он оценивает будущую нагрузку, вероятность перегрузок, изменение очередей задач, состояние сетевых каналов и риск нарушения SLA. Результаты прогнозирования используются для перехода от реактивного к проактивному управлению.

Optimization Agent формирует глобально согласованное распределение ресурсов:

$$X^* = \operatorname{argmin}_X F(X, \hat{S}(t))$$

при ограничениях

$$g_i(X, \hat{S}(t)) \leq b_i.$$

В отличие от локальных агентов, он учитывает состояние системы в целом и обеспечивает поиск компромиссного решения между производительностью, загрузкой ресурсов, сетевыми задержками, энергопотреблением и устойчивостью.

Coordination Agent обеспечивает согласование решений различных агентов:

$$\operatorname{Resolve}(g_i, g_j).$$

Необходимость данного компонента связана с тем, что локальные решения могут конфликтовать между собой. Например, решение Compute Agent о переносе задачи может увеличить сетевую нагрузку, а стратегия минимизации времени выполнения может повысить энергопотребление. Coordination Agent обеспечивает согласование таких решений с глобальными целями системы.

ML Agent усиливает архитектуру за счёт интеллектуального анализа данных. Он используется для выявления закономерностей, прогнозирования, обнаружения аномалий, оценки рисков и адаптации параметров моделей управления. Его функция состоит не в замене специализированных агентов, а в повышении качества их решений за счёт обучаемых моделей.

LLM Agent выполняет функции когнитивной координации и объяснения. Он работает с контекстом, логам, сообщениями агентов, политиками управления, документацией и командами оператора. Его

результатом являются высокоуровневые стратегии, объяснения решений, интерпретация сложных ситуаций и поддержка человека в контуре управления. При этом LLM Agent не должен самостоятельно выполнять критические управляющие действия без проверки ограничений и согласования с другими компонентами архитектуры.

Таким образом, распределение функций управления между агентами обеспечивает архитектурную связность всей системы. Локальные агенты отвечают за оперативное управление отдельными компонентами инфраструктуры, глобальные агенты обеспечивают согласование и оптимизацию, интеллектуальные агенты повышают качество анализа и прогнозирования, а LLM-агенты формируют когнитивный слой управления. В результате формируется распределённая адаптивная система, в которой сложная задача управления ГРВС декомпозируется на набор согласованных агентных функций.

9.6. Интеграция LLM-агентов

В предлагаемой архитектуре адаптивной системы управления LLM-агенты располагаются на верхнем когнитивном уровне и выполняют функции семантического анализа, высокоуровневого планирования, объяснения решений и координации взаимодействия между специализированными агентами. Их интеграция является развитием интеллектуального и мультиагентного подходов, рассмотренных в предыдущих главах, и отражает переход от преимущественно алгоритмического управления к контекстно-ориентированной модели принятия решений [18–21].

В отличие от локальных агентов, работающих с отдельными подсистемами инфраструктуры, LLM-агенты ориентированы на анализ более широкого управленческого контекста. Они могут учитывать не только числовые параметры состояния среды, но и сообщения агентов, логи, трассировки, эксплуатационные политики, документацию, историю инцидентов, цели оператора и результаты предыдущих управляющих воздействий.

Формально функционирование LLM-агента в архитектуре может быть представлено следующим образом:

$$Strategy(t) = LLM(Context(t), Goals, Policies, History),$$

$$X^*(t) = Validate(Strategy(t), G, Policies, Goals),$$

где $Context(t)$ — текущий контекст функционирования вычислительной среды, $Goals$ — цели управления, $Policies$ — политики и ограничения, а $History$ — история наблюдений, решений и результатов их применения.

Данная запись подчёркивает, что LLM-агент не обязательно формирует непосредственное управляющее воздействие. Его основной результат — стратегия, план, объяснение, рекомендация или согласованная

интерпретация ситуации, которые далее используются агентами оптимизации, координации и исполнения. В этом состоит принципиальное отличие LLM-слоя от нижележащих алгоритмических компонентов архитектуры.

Одной из ключевых функций LLM-агентов является интерпретация состояния распределённой вычислительной среды. Если ML-модели и алгоритмы наблюдаемости позволяют обнаружить отклонение, спрогнозировать нагрузку или зафиксировать аномалию, то LLM-агент может сопоставить эти результаты с логами, сообщениями агентов, историей изменений конфигурации и эксплуатационными правилами. Это позволяет перейти от фиксации факта отклонения к формированию гипотез о его причинах.

Второй важной функцией является высокоуровневая координация. В мультиагентной системе локальные агенты могут формировать решения, рациональные в пределах их области ответственности, но конфликтующие на уровне всей системы. Например, решение о переносе задачи на менее загруженный узел может ухудшить сетевые задержки или повысить риск нарушения SLA. LLM-агент в такой ситуации выступает как когнитивный координатор, способный сопоставить локальные предложения, ограничения, цели управления и политики безопасности.

Координационная роль LLM-агента может быть представлена следующим образом:

$$Coordination(t) = LLM(LocalDecisions, Constraints, Goals, Context).$$

Результатом является согласованная стратегия, которая затем должна пройти проверку формальными ограничениями и специализированными механизмами валидации.

Третья функция LLM-агентов связана с объяснением решений. В интеллектуальной системе управления часть решений может формироваться на основе нейросетевых моделей, reinforcement learning, эвристик или сложных гибридных алгоритмов. Такие решения не всегда обладают достаточной интерпретируемостью для оператора. LLM-агент позволяет сформировать человеко-понятное объяснение:

$$Explanation(t) = LLM(X^*(t), \hat{S}(t), Context(t), Constraints).$$

Такое объяснение может включать причины перераспределения ресурсов, основания изменения маршрутов передачи данных, факторы, повлиявшие на масштабирование сервисов, оценку рисков и ожидаемые последствия управляющего воздействия. Это повышает прозрачность системы управления и облегчает аудит решений.

Четвёртая функция связана со стратегическим планированием. В отличие от локальных алгоритмов, ориентированных преимущественно на текущие параметры среды, LLM-агенты могут учитывать более широкий контекст: прогнозируемую динамику нагрузки, историю инцидентов,

долгосрочные цели управления, эксплуатационные политики и ограничения безопасности. Это позволяет использовать их для формирования высокоуровневых сценариев адаптации вычислительной инфраструктуры.

При этом принципиально важно, что LLM-агенты не должны рассматриваться как замена формальным методам оптимизации, ML-моделям или специализированным агентам управления. Они не обеспечивают строгих гарантий оптимальности и допустимости решений. Поэтому их рациональная роль состоит в поддержке принятия решений, анализе контекста, объяснении, планировании и координации, тогда как непосредственное выполнение управляющих воздействий должно оставаться за агентами исполнения и алгоритмическими компонентами с проверкой ограничений.

Практическая интеграция LLM-агентов в архитектуру должна осуществляться через контролируемый контур:

LLM → Recommendation → Validation → Execution.

Это означает, что рекомендации и планы, сформированные языковой моделью, проходят проверку на соответствие ресурсным, сетевым, временным, организационным и безопасностным ограничениям. Для критически значимых действий может использоваться подтверждение оператора или отдельный механизм «human-in-the-loop» — как участие человека в контуре принятия решений.

Таким образом, интеграция LLM-агентов формирует верхний когнитивный уровень архитектуры адаптивного управления ГРВС. Этот уровень обеспечивает работу с контекстом, объяснение решений, стратегическое планирование и координацию агентов, но функционирует совместно с формальными методами оптимизации, ML-моделями, механизмами валидации и специализированными агентами исполнения. Такая организация позволяет расширить адаптивную систему управления без потери контролируемости, устойчивости и проверяемости управляющих воздействий.

9.7. Механизмы адаптации

Ключевым свойством предлагаемой архитектуры является способность системы управления изменять своё поведение при изменении состояния распределённой вычислительной среды. Для ГРВС это принципиально важно, поскольку такие системы функционируют в условиях нестационарной нагрузки, изменяющихся сетевых характеристик, динамического состава ресурсов, отказов отдельных компонентов и постоянной эволюции вычислительной инфраструктуры [12–21].

В подобных условиях фиксированная стратегия управления быстро теряет эффективность. Решение, оптимальное для одного режима нагрузки, может стать неэффективным или недопустимым при изменении состояния сети, появлении новых задач, перегрузке отдельных узлов или изменении доступности ресурсов. Поэтому адаптация в предлагаемой архитектуре рассматривается не как отдельный механизм, а как сквозное свойство всей системы управления.

В архитектуре можно выделить четыре взаимосвязанных уровня адаптации:

- локальную адаптацию;
- глобальную адаптацию;
- интеллектуальную адаптацию;
- архитектурную адаптацию.

Локальная адаптация реализуется на уровне отдельных вычислительных узлов, сетевых сегментов, очередей задач и локальных подсистем. Её задача состоит в быстром реагировании на изменение состояния ограниченной области вычислительной среды:

$$X_i(t+1) = \hat{f}_i(\hat{S}_i(t)),$$

где $\hat{S}_i(t)$ — локальная оценка состояния узла или подсистемы, а $X_i(t+1)$ — локальное управляющее воздействие.

Данный уровень реализуется специализированными агентами вычислительных ресурсов, сетевыми агентами, агентами качества обслуживания и агентами размещения. Они могут оперативно перераспределять задачи, изменять приоритеты обработки, корректировать маршруты передачи данных, ограничивать поступление новых задач или инициировать локальное восстановление после отказа.

Преимуществом локальной адаптации является высокая скорость реакции. Решение принимается вблизи управляемого ресурса и не требует полного анализа состояния всей распределённой системы. Это особенно важно для высокодинамичных сред, где задержка принятия решения может привести к росту очередей, перегрузке узлов или нарушению SLA.

Однако локально эффективные действия не всегда являются эффективными для системы в целом. Например, перенос задачи на менее загруженный узел может увеличить сетевые задержки, а локальная разгрузка одного компонента может вызвать перегрузку другого. Поэтому локальная адаптация должна дополняться глобальной координацией.

Глобальная адаптация реализуется на уровне всей распределённой вычислительной среды и направлена на согласование поведения локальных агентов:

$$X(t+1) = A(\hat{S}(t), G, Goals),$$

где $\hat{S}(t)$ — оценка состояния всей системы, G — система ограничений, а $Goals$ — цели управления.

На данном уровне используются механизмы координации агентов, глобальной оптимизации, прогнозирования нагрузки, анализа ограничений и выбора согласованной стратегии распределения ресурсов. Глобальная адаптация учитывает взаимосвязи между вычислительными узлами, сетевыми каналами, потоками данных, очередями задач и требованиями качества обслуживания.

Её основная функция состоит в том, чтобы предотвратить ситуацию, при которой совокупность локально рациональных решений приводит к глобально неэффективному поведению системы. Поэтому глобальная адаптация обеспечивает согласованность локальных действий с общесистемными целями: производительностью, устойчивостью, соблюдением SLA, энергоэффективностью и допустимостью распределения ресурсов.

Интеллектуальная адаптация связана с изменением параметров моделей управления:

$$\theta(t+1) = \Phi(\theta(t), \hat{S}(t), X(t), S(t+1)).$$

В отличие от локальной и глобальной адаптации, где изменяются управляющие воздействия, на данном уровне изменяются сами модели, стратегии и параметры принятия решений. Это позволяет системе учитывать накопленный опыт, уточнять прогнозные модели, корректировать веса критериев оптимизации, адаптировать поведение агентов и повышать качество управления во времени.

Интеллектуальная адаптация реализуется с использованием методов машинного обучения, reinforcement learning, online learning, нейросетевого прогнозирования и гибридных моделей управления. Например, агент прогнозирования может уточнять модель нагрузки, агент оптимизации — изменять веса критериев, а агент обнаружения аномалий — адаптировать границы нормального поведения при изменении режима функционирования системы.

LLM-агенты также участвуют в интеллектуальной адаптации, но выполняют не вычислительную оптимизацию, а когнитивные функции: анализируют контекст изменений, сопоставляют текущие ситуации с ранее наблюдавшимися сценариями, формируют объяснения причин деградации и помогают выбирать стратегию адаптации. Их рекомендации должны проходить проверку ограничений и согласование с формальными механизмами управления.

Отдельное значение имеет архитектурная адаптация. Она связана с изменением не только управляющих воздействий или параметров моделей, но и структуры самой системы управления:

$\wedge$
 $MAS(t + 1) = R(MAS(t), S(t), Goals, Policies),$

где R — функция перестройки мультиагентной архитектуры.

Архитектурная адаптация может включать изменение связей между агентами, перераспределение областей ответственности, активацию дополнительных агентов, изменение роли отдельных компонентов, подключение новых моделей анализа или временное усиление координационного уровня. Такой механизм особенно важен при изменении масштаба вычислительной среды, появлении новых узлов, отказах компонентов или изменении характера нагрузки.

При этом архитектурная адаптация должна быть контролируемой. Система не должна произвольно изменять собственную структуру без учёта политик безопасности, ограничений устойчивости и глобальных целей управления. Поэтому перестройка архитектуры должна сопровождаться валидацией, журналированием, контролем допустимости и, для критически значимых изменений, участием оператора.

Все уровни адаптации функционируют совместно и образуют единый механизм динамического управления. Локальная адаптация обеспечивает быстрый отклик на изменения в отдельных подсистемах. Глобальная адаптация согласует локальные решения и поддерживает общесистемную эффективность. Интеллектуальная адаптация изменяет модели и стратегии управления на основе накопленного опыта. Архитектурная адаптация позволяет перестраивать саму мультиагентную систему при изменении структуры вычислительной среды.

Таким образом, адаптация в предлагаемой архитектуре реализуется не как единичная процедура корректировки параметров, а как многоуровневый механизм, охватывающий локальные действия, глобальную координацию, обучение моделей и перестройку структуры управления. Благодаря этому система способна сохранять эффективность в условиях высокой динамики, неопределённости и непрерывного изменения гетерогенной распределённой вычислительной среды.

9.8. Масштабируемость системы

Одним из ключевых требований к архитектуре адаптивной системы управления гетерогенными распределёнными вычислительными системами является масштабируемость. Современные вычислительные среды характеризуются ростом числа узлов, увеличением объёмов данных, расширением набора сервисов, появлением новых типов ресурсов и усложнением взаимодействия между компонентами инфраструктуры [12–21].

В предлагаемой архитектуре масштабируемость обеспечивается не за счёт увеличения мощности единого управляющего центра, а за счёт

распределённой организации управления, модульного добавления агентов, локальности принятия решений и горизонтального расширения подсистем наблюдаемости и интеллектуального анализа.

На инфраструктурном уровне масштабирование связано с подключением новых вычислительных узлов:

$$N \rightarrow N \cup \{n_{new}\},$$

где n_{new} — новый компонент распределённой вычислительной среды.

Добавление нового узла не требует принципиальной перестройки архитектуры управления. Новый компонент включается в контур наблюдаемости, получает собственные метрики и параметры состояния, после чего может быть связан с локальными агентами управления. При этом узел может отличаться архитектурой, производительностью, доступной памятью, поддержкой GPU, сетевой близостью, энергетическими характеристиками и ограничениями размещения. Это особенно важно для облачных, гибридных и edge-сред, где вычислительная инфраструктура является динамически изменяемой и неоднородной.

На агентном уровне масштабируемость обеспечивается расширением множества агентов:

$$A \rightarrow A \cup \{a_{new}\},$$

где a_{new} — новый агент системы управления.

Новые агенты могут отвечать за дополнительные вычислительные узлы, сетевые сегменты, классы задач, механизмы прогнозирования, функции оптимизации или специализированные политики управления. Благодаря модульной организации мультиагентной системы добавление агента не требует изменения всей архитектуры, если сохраняются стандартные протоколы взаимодействия, обмена состоянием и согласования решений.

Важным механизмом масштабируемости является локальность принятия решений. Значительная часть управляющих воздействий формируется локальными агентами непосредственно на уровне управляемых подсистем:

$$X_i(t) = a_i(\hat{S}_i(t)).$$

Это снижает нагрузку на глобальные компоненты архитектуры и предотвращает нелинейный рост вычислительной сложности при увеличении числа узлов и задач. Глобальные агенты при этом используются преимущественно для согласования, стратегического планирования и разрешения конфликтов между локальными решениями.

Отдельное значение имеет масштабируемость подсистемы наблюдаемости. При росте числа узлов увеличивается объём метрик, логов, трассировок и событий мониторинга. Поэтому подсистема данных

должна быть распределённой и поддерживать локальную агрегацию наблюдений:

$$\hat{S}_i(t) = O_i(Y_i(t)),$$

где O_i — функция обработки данных локальной подсистемы, а $Y_i(t)$ — поток наблюдений от соответствующего сегмента вычислительной среды.

Такой подход позволяет не передавать все сырые данные в единый центр обработки, а формировать локальные оценки состояния, которые затем используются глобальными агентами и интеллектуальными моделями.

Масштабируемость интеллектуального уровня обеспечивается за счёт распределения ML- и LLM-компонентов между различными уровнями архитектуры. Вместо единой универсальной модели могут использоваться специализированные модели прогнозирования, обнаружения аномалий, оценки риска, маршрутизации и координации. Это позволяет адаптировать интеллектуальную подсистему к размеру и структуре вычислительной среды.

LLM-агенты в масштабируемой архитектуре не должны обрабатывать весь поток телеметрии напрямую. Их рациональная роль состоит в анализе агрегированного контекста, объяснении сложных ситуаций, координации агентов и поддержке оператора. Такой подход снижает вычислительную нагрузку на когнитивный уровень и позволяет использовать LLM там, где требуется именно контекстный анализ и рассуждение, а не массовая потоковая обработка данных.

Таким образом, масштабируемость предлагаемой архитектуры достигается за счёт нескольких взаимосвязанных механизмов: добавления новых узлов, расширения множества агентов, локального принятия решений, распределённой подсистемы наблюдения, модульной организации интеллектуальных моделей и ограниченного использования LLM-агентов на уровне когнитивной координации. Благодаря этому система управления способна сохранять эффективность при увеличении масштаба и сложности гетерогенной распределённой вычислительной среды.

9.9. Отказоустойчивость

Для гетерогенных распределённых вычислительных систем отказоустойчивость является одним из ключевых требований к архитектуре управления. Распределённый характер инфраструктуры, большое число взаимодействующих компонентов и динамическое изменение состояния среды приводят к тому, что локальные отказы, перегрузки, деградация каналов связи и сбои отдельных сервисов должны

рассматриваться как нормальный режим функционирования крупномасштабной ГРВС [12–21].

В предлагаемой архитектуре отказоустойчивость обеспечивается сочетанием нескольких механизмов: инфраструктурного резервирования, распределённого агентного управления, обнаружения деградации, самовосстановления, интеллектуальной диагностики и контролируемого изменения управляющих стратегий.

Базовым механизмом является резервирование компонентов:

$$n_i \rightarrow \{n_i^1, n_i^2, \dots, n_i^r\},$$

где для критически значимого узла n_i формируется множество резервных компонентов, способных принять его функции при отказе.

Резервирование может осуществляться на уровне вычислительных узлов, сетевых маршрутов, контейнеров, сервисов, хранилищ данных, управляющих агентов и моделей анализа. Его задача состоит в том, чтобы отказ отдельного компонента не приводил к полной потере функциональности распределённой системы.

Однако резервирование само по себе недостаточно. В динамической ГРВС важно не только иметь резервные ресурсы, но и уметь определить момент деградации, выбрать корректный сценарий восстановления и согласовать его с текущими ограничениями системы. Поэтому в архитектуре используется механизм самовосстановления:

$$S(t+1) = \overset{\wedge}{\text{Recover}}(S(t), \text{Failure}(t), G),$$

где $\text{Failure}(t)$ — обнаруженное нарушение или отказ, а G — система ограничений, определяющих допустимые действия восстановления.

Процессы самовосстановления могут включать перераспределение задач, миграцию сервисов, изменение маршрутов передачи данных, активацию резервных компонентов, повторный запуск контейнеров, перестройку конфигурации, снижение нагрузки на деградирующий компонент и временное изменение приоритетов обработки задач.

Важную роль в обеспечении отказоустойчивости играют локальные агенты. Они способны быстро обнаруживать нарушения в своих областях ответственности и инициировать первичные действия восстановления без ожидания глобального анализа всей системы. Например, Compute Agent может зафиксировать перегрузку или отказ узла, Network Agent — деградацию канала связи, QoS Agent — риск нарушения SLA.

Глобальные агенты обеспечивают координацию восстановления на уровне всей вычислительной среды. Их задача состоит в том, чтобы локальные действия восстановления не создавали новых конфликтов. Например, перенос задач с отказавшего узла не должен приводить к перегрузке резервного узла, росту сетевых задержек или нарушению ограничений размещения.

Интеллектуальные компоненты архитектуры повышают качество отказоустойчивости за счёт прогнозирования деградации и анализа причин отказов. ML-модели могут выявлять ранние признаки перегрузки, аномалии в телеметрии, нестандартное поведение очередей задач или рост вероятности отказа компонента:

$$Risk(t) = ML(\hat{S}(t), H(t)).$$

LLM-агенты могут использоваться для анализа сложных инцидентов, интерпретации логов, сопоставления текущей ситуации с историей отказов, объяснения причин деградации и формирования высокоуровневых сценариев восстановления. При этом они не должны самостоятельно выполнять критические восстановительные действия без проверки ограничений и подтверждения со стороны специализированных механизмов управления.

Практический контур восстановления можно представить следующим образом:

Detect → *Diagnose* → *PlanRecover* → *Validate* → *Execute* → *Observe*.

На этапе *Detect* система фиксирует отказ или деградацию. На этапе *Diagnose* локальные, глобальные и интеллектуальные агенты анализируют причины нарушения. На этапе *PlanRecover* формируется сценарий восстановления. На этапе *Validate* проверяется допустимость действий. На этапе *Execute* выполняются восстановительные воздействия. На этапе *Observe* оценивается результат восстановления и замыкается обратная связь.

Таким образом, отказоустойчивость в предлагаемой архитектуре рассматривается не только как наличие резервных компонентов, но и как способность системы управления обнаруживать деградацию, координировать восстановительные действия, адаптировать распределение ресурсов и сохранять работоспособность вычислительной среды при частичных отказах. Сочетание резервирования, мультиагентного управления, интеллектуальной диагностики и контролируемого самовосстановления обеспечивает устойчивое функционирование ГРВС в условиях неопределённости и динамического изменения инфраструктуры.

9.10. Самоорганизация системы

Одной из принципиальных особенностей предлагаемой архитектуры является способность системы управления к самоорганизации. В отличие от статических архитектур, где структура взаимодействия компонентов фиксируется заранее, самоорганизующаяся система способна изменять конфигурацию взаимодействия агентов, распределение функций и параметры координации в зависимости от состояния вычислительной среды [12–21].

Необходимость самоорганизации обусловлена высокой изменчивостью современных ГРВС. В процессе функционирования могут изменяться состав вычислительных узлов, характер нагрузки, структура сетевых потоков, доступность ресурсов, конфигурация сервисов, приоритеты задач и набор ограничений. В таких условиях заранее заданная схема взаимодействия агентов может перестать соответствовать текущему состоянию среды.

В предлагаемой архитектуре самоорганизация может быть представлена следующим образом:

$$MAS(t+1) = \Phi(MAS(t), \hat{S}(t), Goals, Policies),$$

где $MAS(t)$ — текущее состояние мультиагентной системы управления, $\hat{S}$ — оценка состояния вычислительной среды, $Goals$ — цели управления, а $Policies$ — политики и ограничения, в рамках которых допускается перестройка архитектуры.

Данная модель отражает способность системы изменять не только управляющие воздействия, но и собственную организацию. Самоорганизация может включать изменение структуры взаимодействия агентов, перераспределение областей ответственности, активацию дополнительных агентов, временное отключение неиспользуемых компонентов, изменение частоты коммуникации, перестройку координационных связей и выбор других стратегий согласования решений.

На локальном уровне самоорганизация проявляется в изменении поведения агентов при изменении состояния их подсистем. Например, агент может увеличить частоту обмена данными при росте нагрузки, запросить координацию при возникновении конфликта ограничений или передать часть задач другому агенту при перегрузке.

На глобальном уровне самоорганизация связана с перестройкой структуры взаимодействия между локальными и глобальными агентами. При росте масштаба системы может потребоваться выделение дополнительных координирующих агентов, группировка узлов в кластеры управления или изменение маршрутов передачи управляющей информации.

На интеллектуальном уровне самоорганизация поддерживается ML- и LLM-компонентами. ML-модели позволяют выявлять устойчивые изменения в структуре нагрузки и предлагать изменение параметров взаимодействия агентов. LLM-агенты могут анализировать контекст перестройки, сопоставлять её с политиками управления, объяснять причины изменения архитектуры и помогать оператору оценивать последствия таких изменений.

Принципиально важно, что самоорганизация не означает произвольное изменение архитектуры. В контуре управления ГРВС она должна быть управляемой и ограниченной. Перестройка структуры

агентов, изменение ролей или активация новых интеллектуальных компонентов должны осуществляться в рамках заданных политик, ограничений безопасности и глобальных целей функционирования вычислительной среды.

Поэтому процесс самоорганизации должен включать проверку допустимости:

$$MAS(t + 1) = \text{Validate}(\Phi(MAS(t), S(t), \overset{\wedge}{Goals}, Policies), Policies, G).$$

Это позволяет избежать неконтролируемых изменений, конфликтов между агентами, избыточной коммуникации и нестабильного поведения системы. Для критически значимых изменений структуры управления может применяться подтверждение оператора или механизм участия человека в контуре принятия решений (human-in-the-loop).

Самоорганизация позволяет системе управления адаптироваться не только на уровне отдельных управляющих воздействий, но и на уровне собственной архитектуры. Это особенно важно при изменении масштаба вычислительной среды, появлении новых классов задач, отказах компонентов, изменении топологии сети или длительном смещении профиля нагрузки.

Таким образом, самоорганизация рассматривается как фундаментальное свойство интеллектуальной архитектуры адаптивного управления ГРВС. Она обеспечивает контролируемую перестройку мультиагентной системы, перераспределение функций управления и изменение структуры координации в зависимости от состояния вычислительной среды. В сочетании с механизмами адаптации, отказоустойчивости и интеллектуального анализа самоорганизация позволяет поддерживать эффективность управления в условиях непрерывно изменяющейся распределённой инфраструктуры.

9.11. Реализация в современных платформах

Предлагаемая архитектура адаптивного управления гетерогенными распределёнными вычислительными системами ориентирована на практическую реализацию в современных распределённых платформах, для которых характерны динамическое масштабирование, сервисно-ориентированная организация вычислений, контейнеризация, оркестрация, потоковая обработка событий и развитые средства наблюдаемости [12–21].

В контексте современных платформ архитектура может быть сопоставлена с несколькими технологическими слоями. Инфраструктурный уровень представлен облачными, гибридными и edge-ресурсами. Уровень наблюдаемости реализуется средствами сбора метрик, логов, трассировок и событий. Уровень исполнения управляющих воздействий связан с оркестраторами, контейнерными платформами и

механизмами автоматизированного развертывания. Агентный и интеллектуальный уровни могут быть реализованы как совокупность сервисов, взаимодействующих через API, очереди сообщений, событийные шины и средства координации.

В облачных и гибридных средах вычислительные ресурсы предоставляются динамически, а состав инфраструктуры может изменяться в зависимости от нагрузки, доступности сервисов и требований приложений. Это соответствует базовой идее предлагаемой архитектуры, в которой управляемая среда E рассматривается как изменяемая распределённая система, а управление строится на основе непрерывного наблюдения и адаптации. В таких условиях механизмы масштабирования, перераспределения задач, балансировки нагрузки и самовосстановления становятся естественными элементами контура управления.

Контейнерные платформы обеспечивают переносимость вычислительных компонентов, изоляцию сервисов и возможность динамического размещения задач на различных узлах. В рамках предлагаемой архитектуры контейнер или сервис может рассматриваться как объект управления, для которого принимаются решения о размещении, масштабировании, миграции, перезапуске или изменении конфигурации. Это позволяет связать формальные модели распределения ресурсов с практическими механизмами управления приложениями в современной вычислительной среде.

Системы оркестрации выполняют роль исполнительного слоя архитектуры. Они реализуют управляющие воздействия, сформированные мультиагентной и интеллектуальной подсистемами:

$$X^*(t) \rightarrow \text{Execute}(X^*(t)).$$

Через оркестратор могут выполняться масштабирование сервисов, перераспределение контейнеров, изменение конфигурации сетевого взаимодействия, перезапуск компонентов, переключение на резервные ресурсы и восстановление после отказов. Таким образом, оркестратор не заменяет интеллектуальную систему управления, а выступает механизмом применения решений к вычислительной инфраструктуре.

Уровень наблюдаемости в современных платформах реализуется через сбор и обработку метрик, логов, трассировок и событий:

$$Y(t) = \{Metrics, Logs, Traces, Events\}.$$

Эти данные поступают в подсистему обработки и преобразуются в оценку состояния:

$$\hat{S}(t) = O(Y(t)).$$

На практике это означает, что архитектура должна быть интегрирована со стеком наблюдаемости (observability stack), способным поддерживать потоковый сбор данных, агрегацию, фильтрацию, хранение

истории и передачу признаков в агенты управления и интеллектуальные модели.

Особое значение имеют событийно-ориентированные механизмы (event-driven). В современных распределённых платформах изменение состояния инфраструктуры, возникновение перегрузки, отказ компонента, превышение порога задержки или нарушение SLA может быть представлено как событие:

$$e = (type, timestamp, payload).$$

Такие события используются для запуска реактивных и проактивных сценариев адаптации. Например, событие деградации узла может инициировать диагностику, перераспределение задач или активацию резервного компонента; событие роста очереди — изменение приоритетов обработки или масштабирование сервиса; событие нарушения сетевых параметров — пересмотр маршрутов передачи данных.

Мультиагентная подсистема может быть реализована как совокупность слабосвязанных сервисов, каждый из которых отвечает за отдельную функцию управления: мониторинг, прогнозирование, размещение задач, сетевую маршрутизацию, контроль SLA, оптимизацию или координацию. Взаимодействие агентов может осуществляться через очереди сообщений, API, событийные каналы или общее хранилище состояния. Такая реализация соответствует cloud-native подходу и позволяет масштабировать систему управления вместе с вычислительной инфраструктурой.

Интеллектуальный уровень архитектуры может быть реализован как набор специализированных ML-сервисов и аналитических модулей. Одни модели прогнозируют нагрузку, другие выявляют аномалии, третьи оценивают риск нарушения SLA или помогают выбирать стратегию распределения ресурсов. При этом интеллектуальные компоненты не должны быть монолитными: для разных подсистем могут использоваться разные модели, обученные на локальных данных и адаптированные к особенностям соответствующего сегмента инфраструктуры.

LLM-агенты в современных платформах целесообразно размещать на уровне когнитивной координации и поддержки оператора. Они могут использоваться для анализа сложных инцидентов, интерпретации логов, объяснения решений, формирования рекомендаций, работы с эксплуатационной документацией, поддержки DevOps-процессов и взаимодействия с человеком на естественном языке. При этом LLM-компоненты не должны напрямую выполнять критические операции без проверки ограничений, журналирования и, при необходимости, подтверждения оператора.

Практическая реализация архитектуры может быть представлена следующим образом:

Observability → Agents → AI/LLM → Orchestrator → Infrastructure.

В этой цепочке средства наблюдения формируют данные о состоянии среды, агенты выполняют локальный и глобальный анализ, интеллектуальные модели и LLM-компоненты формируют рекомендации и стратегии, оркестратор применяет допустимые управляющие воздействия, а инфраструктура изменяет своё состояние и снова становится источником данных для наблюдения.

Таким образом, предлагаемая архитектура не является абстрактной теоретической схемой. Она соответствует логике современных облачных, гибридных, edge- и cloud-native платформ, в которых управление реализуется через сочетание наблюдаемости, событийной обработки, контейнеризации, оркестрации, мультиагентной координации и интеллектуального анализа данных. Такое соответствие позволяет рассматривать предложенную архитектуру как основу для построения адаптивных систем управления распределённой вычислительной инфраструктурой нового поколения.

9.12. Ключевой результат

Основным результатом главы является формирование целостной архитектуры адаптивной интеллектуальной системы управления гетерогенными распределёнными вычислительными системами, объединяющей механизмы наблюдаемости, мультиагентное управление, интеллектуальные методы анализа данных и LLM-агентов в единую когнитивную распределённую среду управления [12–21].

Предложенная архитектура обобщает результаты, полученные в предыдущих главах. Формальная модель ГРВС задаёт объект управления и систему ограничений; подсистема наблюдаемости формирует оценку состояния вычислительной среды; мультиагентный уровень обеспечивает распределённое управление; интеллектуальные модели выполняют прогнозирование, оптимизацию и адаптацию параметров; LLM-агенты формируют когнитивный слой анализа контекста, объяснения решений и высокоуровневой координации.

Формально архитектура может быть представлена следующим образом:

$$ARCH = \langle E, D, MAS, AI, LLM, C \rangle,$$

где E — управляемая вычислительная среда, D — подсистема данных и наблюдаемости, MAS — мультиагентная система управления, AI — интеллектуальные модели и алгоритмы, LLM — когнитивный слой LLM-агентов, C — замкнутый контур адаптивного управления.

Ключевая особенность архитектуры состоит в том, что управление ГРВС рассматривается не как выполнение отдельного алгоритма распределения ресурсов, а как непрерывный процесс взаимодействия нескольких уровней:

$$S(t) \rightarrow Y(t) \rightarrow \hat{S}(t) \rightarrow MAS \rightarrow AI/LLM \rightarrow X^*(t) \rightarrow S(t+1).$$

Данный контур объединяет наблюдение состояния среды, обработку данных, локальное и глобальное агентное управление, интеллектуальный анализ, когнитивную координацию и применение управляющих воздействий к вычислительной инфраструктуре.

Принципиальным результатом является сочетание локального, глобального, интеллектуального и когнитивного управления. Локальные агенты обеспечивают быстрый отклик на изменение состояния отдельных узлов и подсистем. Глобальные агенты согласуют локальные решения и поддерживают общесистемные цели. Интеллектуальные модели повышают качество прогнозирования, оптимизации и адаптации. LLM-агенты обеспечивают работу с контекстом, объяснение решений, стратегическое планирование и взаимодействие с оператором.

Предложенная архитектура также обладает свойствами масштабируемости, отказоустойчивости и управляемой самоорганизации. Она допускает добавление новых узлов, агентов и интеллектуальных компонентов, поддерживает распределённую обработку данных, обеспечивает восстановление после частичных отказов и позволяет перестраивать структуру взаимодействия агентов в рамках заданных политик и ограничений.

Таким образом, ключевым результатом главы является обоснование архитектуры интеллектуального адаптивного управления ГРВС нового поколения, в которой наблюдаемость, управляемая данными, мультиагентная координация, AI-модели и LLM-агенты объединяются в единую когнитивную систему управления распределённой вычислительной инфраструктурой.

Выводы по главе 9

В главе разработана архитектура адаптивной системы управления гетерогенными распределёнными вычислительными системами, объединяющая результаты предыдущих этапов исследования: формальную модель ГРВС, контур адаптивного управления, мультиагентный подход, методы распределения ресурсов, наблюдаемость, интеллектуальные модели и LLM-агентов [12–21].

Сформирована общая модель архитектуры:

$$ARCH = \langle E, D, MAS, AI, LLM, C \rangle.$$

В данной модели управляемая вычислительная среда, подсистема данных, мультиагентная система, интеллектуальные алгоритмы, LLM-компоненты и контур адаптивного управления рассматриваются как взаимосвязанные элементы единой системы управления.

Предложена многоуровневая структура архитектуры, включающая инфраструктурный уровень, уровень данных и наблюдаемости, уровень адаптивного управления, агентный уровень, интеллектуальный и когнитивный уровень, а также уровень принятия решений. Такое разделение позволяет развести процессы сбора данных, анализа состояния, локального управления, глобальной координации и формирования итоговых управляющих воздействий.

Показано, что функционирование архитектуры реализуется через замкнутый контур:

$$S(t) \rightarrow Y(t) \rightarrow \hat{S}(t) \rightarrow MAS \rightarrow AI/LLM \rightarrow X^*(t) \rightarrow S(t+1).$$

Данный контур обеспечивает непрерывное наблюдение состояния вычислительной среды, формирование оценки состояния, распределённое принятие решений, интеллектуальный анализ, применение управляющих воздействий и получение обратной связи.

Разработана агентная структура системы управления. Выделены локальные агенты, отвечающие за оперативное управление отдельными ресурсами и подсистемами; глобальные агенты, обеспечивающие оптимизацию, прогнозирование и координацию; интеллектуальные агенты, реализующие обучаемые модели анализа; LLM-агенты, формирующие когнитивный слой управления.

Распределение функций управления между агентами связано с системой ограничений ГРВС. Compute Agent контролирует вычислительные ресурсы, Network Agent — сетевые ограничения, QoS Agent — временные требования и SLA, Placement Agent — допустимость размещения, Prediction Agent — прогноз состояния среды, Optimization Agent — глобальное распределение ресурсов, Coordination Agent — согласование локальных решений, ML Agent — интеллектуальный анализ данных, LLM Agent — контекстную интерпретацию и объяснение решений.

Особое внимание уделено интеграции LLM-агентов в архитектуру. Показано, что они не заменяют формальную оптимизацию, ML-модели и специализированных агентов исполнения, а формируют верхний когнитивный уровень, отвечающий за семантический анализ, стратегическое планирование, объяснение решений, координацию агентов и взаимодействие с оператором.

Рассмотрены механизмы адаптации архитектуры. Установлено, что адаптация реализуется на локальном, глобальном, интеллектуальном и архитектурном уровнях. Локальная адаптация обеспечивает быстрый отклик на изменения состояния отдельных подсистем, глобальная — согласованность поведения ГРВС, интеллектуальная — обновление моделей и стратегий управления, архитектурная — контролируемую перестройку мультиагентной системы.

Показано, что предлагаемая архитектура обладает масштабируемостью, отказоустойчивостью и способностью к управляемой самоорганизации. Масштабируемость достигается за счёт добавления новых узлов, агентов и интеллектуальных компонентов, локальности принятия решений и распределённой подсистемы наблюдения. Отказоустойчивость обеспечивается резервированием, мультиагентным восстановлением, интеллектуальной диагностикой и контролируемыми сценариями самовосстановления. Самоорганизация реализуется как перестройка взаимодействия агентов и распределения функций в рамках заданных политик, ограничений и целей управления.

Проанализированы возможности реализации архитектуры в современных распределённых платформах. Показано, что предложенная модель соответствует логике cloud-native, гибридных и edge-сред, где управление строится на основе наблюдаемости, событийной обработки, контейнеризации, оркестрации, мультиагентной координации и интеллектуального анализа данных.

Таким образом, результаты главы формируют завершённую архитектурную модель адаптивной интеллектуальной системы управления ГРВС. Данная модель объединяет наблюдаемость, управляемую данными, мультиагентное распределённое управление, интеллектуальные методы анализа и LLM-агентов в единую когнитивную среду управления распределённой вычислительной инфраструктурой.

ГЛАВА 10. ЭКСПЕРИМЕНТАЛЬНАЯ АПРОБАЦИЯ И ОЦЕНКА ЭФФЕКТИВНОСТИ АДАПТИВНОГО УПРАВЛЕНИЯ

10.1. Цели и научная постановка эксперимента

Целью экспериментального исследования является количественная оценка эффективности предложенной архитектуры адаптивного управления гетерогенными распределёнными вычислительными системами, основанной на интеграции мультиагентного подхода, интеллектуальных методов анализа и механизмов когнитивного управления [12–21].

В предыдущих главах были разработаны формальная модель распределённой вычислительной среды, мультиагентная архитектура управления, механизмы наблюдаемости, интеллектуальные методы адаптации и когнитивные компоненты на основе LLM-агентов. В связи с этим возникает задача экспериментальной проверки эффективности предложенных решений в условиях динамически изменяющейся вычислительной среды.

В отличие от большинства существующих исследований, в которых методы оптимизации, машинного обучения, мультиагентного управления и интеллектуального анализа рассматриваются изолированно, в данной работе исследуется единая интегрированная система управления, объединяющая:

- адаптивные алгоритмы распределения ресурсов;
- мультиагентные механизмы координации;
- нейросетевые и ML-модели;
- ZNN-подходы;
- гибридные интеллектуальные методы;
- элементы LLM-агентов.

Принципиальной особенностью исследования является рассмотрение системы управления как единого адаптивного интеллектуального контура:

$$S(t) \rightarrow Y(t) \rightarrow \hat{S}(t) \rightarrow MAS \rightarrow AI/LLM \rightarrow X^*(t) \rightarrow S(t+1)$$

в котором процессы наблюдения, прогнозирования, координации и принятия решений функционируют совместно и оказывают взаимное влияние на динамику вычислительной среды.

Основная научная задача экспериментального исследования заключается в оценке влияния интеллектуализации и мультиагентной организации управления на эффективность функционирования распределённой вычислительной системы. В рамках экспериментов анализируется способность предложенной архитектуры:

- адаптироваться к изменению нагрузки;
- обеспечивать устойчивость распределения ресурсов;

- снижать перегрузку вычислительных узлов;
- улучшать показатели времени выполнения задач;
- повышать устойчивость системы к отказам;
- обеспечивать согласованность распределённых решений.

Особое внимание уделяется исследованию влияния интеллектуальных компонентов архитектуры на качество управления вычислительной средой. В частности, оценивается эффективность применения ML-моделей, ZNN-методов и гибридных интеллектуальных алгоритмов в задачах прогнозирования нагрузки, адаптации параметров управления и координации распределённых процессов.

Дополнительно рассматривается влияние LLM-агентов на процессы высокоуровневой координации и интеллектуального анализа состояния распределённой вычислительной среды. В рамках исследования анализируется возможность использования когнитивных механизмов для повышения качества принятия решений в условиях неопределённости и динамического изменения состояния инфраструктуры.

Экспериментальная постановка исследования ориентирована на моделирование реальных сценариев функционирования гетерогенных распределённых вычислительных систем, включая:

- изменение интенсивности вычислительной нагрузки;
- неравномерное распределение задач;
- динамическое изменение доступных ресурсов;
- возникновение перегрузок;
- отказ отдельных компонентов вычислительной среды;
- изменение сетевых характеристик.

В процессе экспериментов производится сравнительный анализ различных подходов к управлению распределённой вычислительной средой, включая классические алгоритмы распределения ресурсов, адаптивные методы, интеллектуальные модели и предложенную гибридную мультиагентную архитектуру.

Таким образом, экспериментальное исследование направлено не только на проверку работоспособности отдельных алгоритмов, но и на оценку эффективности сформированной интеллектуальной архитектуры управления ГРВС как единой самоадаптирующейся распределённой системы управления вычислительной инфраструктурой.

Структура экспериментального стенда, использованного для проведения исследований, представлена на рисунке 12.

Экспериментальный стенд реализует предложенную в главе 9 архитектуру адаптивного управления и включает вычислительную среду, подсистему наблюдаемости, мультиагентный уровень управления, интеллектуальные компоненты анализа и контур принятия решений. Архитектура стенда ориентирована на моделирование динамически изменяющейся гетерогенной распределённой вычислительной системы с

$$\hat{S}(t) = O(Y(t)).$$

Полученные данные поступают в мультиагентную систему управления, включающую локальных и глобальных агентов. Локальные агенты выполняют управление вычислительными ресурсами и сетевыми потоками на уровне отдельных узлов, тогда как глобальные агенты реализуют координацию распределённых решений, прогнозирование и оптимизацию распределения ресурсов.

Интеллектуальный уровень экспериментального стенда включает модели машинного обучения, ZNN-подходы, гибридные алгоритмы и элементы LLM-агентов. Данные компоненты используются для прогнозирования нагрузки, адаптации параметров управления, интеллектуального анализа состояния вычислительной среды и формирования стратегий распределения ресурсов.

Контур управления реализует формирование управляющих воздействий:

$$X^*(t) = \text{Decision}(\hat{S}(t), MAS, AI, LLM, G, Goals),$$

после чего изменения состояния распределённой вычислительной среды вновь фиксируются подсистемой наблюдаемости, замыкая цикл адаптивного управления.

Архитектура стенда позволяет проводить экспериментальные исследования в условиях изменяющейся нагрузки, моделировать отказ отдельных компонентов инфраструктуры, анализировать эффективность различных стратегий распределения ресурсов и исследовать влияние интеллектуальных механизмов на устойчивость и производительность распределённой вычислительной среды.

10.2. Экспериментальная база и методология

Экспериментальное исследование основано на использовании комплекса имитационных, агентных и аналитических моделей, предназначенных для анализа процессов адаптивного управления гетерогенными распределёнными вычислительными системами [12–21]. Выбор подобного подхода обусловлен высокой сложностью распределённой вычислительной среды и невозможностью проведения полномасштабных экспериментов на реальных инфраструктурах при контролируемом изменении параметров нагрузки, структуры сети и характеристик вычислительных ресурсов.

Методологическая основа исследования соответствует подходам, развиваемым в рамках научной школы, связанной с моделированием гетерогенных распределённых систем обработки информации, адаптивным управлением вычислительными ресурсами и интеллектуальными методами распределения нагрузки. В качестве базового инструментария

используются имитационные модели распределённых вычислительных сред, модели сетей массового обслуживания и мультиагентные модели координации процессов управления.

Использование имитационного моделирования позволяет воспроизводить динамику распределённой вычислительной среды в условиях изменяющейся нагрузки и исследовать поведение системы управления при различных сценариях функционирования инфраструктуры. В отличие от статических аналитических моделей, имитационный подход обеспечивает возможность исследования нелинейных эффектов, возникающих при взаимодействии множества вычислительных компонентов и интеллектуальных агентов.

Модели сетей массового обслуживания используются для описания процессов поступления, распределения и выполнения вычислительных задач. Применение данного подхода позволяет анализировать характеристики потоков запросов, длину очередей, загрузку вычислительных узлов и временные параметры обработки задач в распределённой вычислительной среде.

Агентные модели применяются для исследования процессов распределённого принятия решений и координации специализированных компонентов системы управления. В рамках экспериментов каждый агент рассматривается как автономный элемент мультиагентной системы, принимающий решения на основе локального состояния среды и взаимодействующий с другими агентами посредством механизмов координации и обмена информацией.

Формально состояние экспериментальной среды определяется следующим образом:

$$S_{exp}(t) = \langle N, L, T, R(t) \rangle,$$

где N представляет множество вычислительных узлов, L — структуру сетевых взаимодействий, T — множество вычислительных задач, а $R(t)$ — текущее состояние вычислительных ресурсов распределённой среды.

Процесс управления вычислительной инфраструктурой описывается отображением:

$$X^*(t) = A(S(t)),$$

где A представляет совокупность алгоритмов адаптивного управления, мультиагентной координации и интеллектуального анализа состояния системы.

В процессе экспериментов моделируются сценарии изменения вычислительной нагрузки, перераспределения задач между узлами, перегрузки отдельных компонентов инфраструктуры, изменения сетевых характеристик и возникновения отказов вычислительных элементов. Подобный подход позволяет исследовать устойчивость архитектуры

управления и оценить эффективность механизмов адаптации в условиях динамически изменяющейся распределённой вычислительной среды.

Особое внимание уделяется исследованию интеллектуальных механизмов управления. В рамках экспериментов анализируется влияние ML-моделей, ZNN-подходов, гибридных алгоритмов и LLM-компонентов на качество распределения ресурсов, устойчивость системы и эффективность адаптации к изменению состояния вычислительной среды.

Для обеспечения сопоставимости результатов используются идентичные сценарии нагрузки и единые параметры вычислительной среды при сравнении различных методов управления. Это позволяет проводить корректную оценку влияния интеллектуальных и мультиагентных механизмов на характеристики функционирования распределённой вычислительной системы.

Разработанный экспериментальный стенд реализован в виде программного комплекса для моделирования адаптивного управления распределёнными вычислительными системами. Реализация включает средства генерации сценариев нагрузки, моделирования распределения вычислительных ресурсов, сбора метрик и автоматизированного проведения серий вычислительных экспериментов.

Исходный код экспериментального стенда, конфигурации сценариев и инструменты воспроизводимого запуска экспериментов размещены в открытом репозитории автора [141].

Таким образом, экспериментальная база исследования представляет собой интегрированную среду моделирования, объединяющую имитационные модели, методы теории массового обслуживания, мультиагентный подход и интеллектуальные механизмы адаптивного управления, что обеспечивает возможность комплексного исследования эффективности предложенной архитектуры управления ГРВС.

10.3. Экспериментальные сценарии

Экспериментальное исследование эффективности предложенной архитектуры адаптивного управления выполняется на основе набора сценариев, моделирующих различные режимы функционирования гетерогенной распределённой вычислительной среды [12–21]. Использование сценарного подхода позволяет исследовать поведение системы управления в условиях изменения нагрузки, нарушения устойчивости инфраструктуры и возникновения конфликтующих ограничений.

Организация экспериментального процесса строится как последовательный pipeline, включающий формирование сценария функционирования системы, генерацию вычислительных задач, применение алгоритмов управления, имитационное моделирование

поведения распределённой среды и анализ полученных метрик эффективности. Общая схема экспериментального процесса представлена на рисунке 13.



Рисунок 13 – Pipeline экспериментальных исследований

На первом этапе задаётся сценарий функционирования распределённой вычислительной среды, определяющий параметры нагрузки, структуру вычислительной инфраструктуры и условия функционирования системы. Далее формируется поток вычислительных задач:

$$T(t) = \{t_1, t_2, \dots, t_n\},$$

после чего применяется исследуемый алгоритм управления:

$$X(t) = A(S(t)).$$

В процессе моделирования воспроизводится динамика изменения состояния распределённой вычислительной среды:

$$S(t+1) = f(S(t), X(t)),$$

а результатом эксперимента является набор количественных характеристик эффективности управления:

$$M = \{m_1, m_2, \dots, m_k\}.$$

Первый экспериментальный сценарий соответствует стационарному режиму функционирования распределённой вычислительной среды. В данном случае система функционирует при фиксированной интенсивности нагрузки и относительно равномерном распределении вычислительных задач между узлами инфраструктуры. Данный сценарий используется как базовая модель для сравнения различных методов управления и оценки поведения системы в условиях отсутствия выраженных внешних возмущений.

Следующий сценарий связан с динамическим изменением нагрузки:

$$T(t) \uparrow \downarrow.$$

В рамках данного эксперимента интенсивность поступления задач изменяется во времени, что позволяет исследовать способность системы адаптироваться к нестационарным условиям функционирования. Особое внимание уделяется скорости реакции системы управления, устойчивости распределения ресурсов и способности интеллектуальных компонентов прогнозировать изменение нагрузки.

Одним из ключевых сценариев исследования является моделирование перегрузки вычислительных узлов. Данный подход продолжает линию исследований, ранее проводившихся в работах научной школы, где было показано, что локальная перегрузка вычислительных

компонентов приводит к росту времени выполнения задач, увеличению длины очередей и ухудшению качества распределения ресурсов.

В условиях перегрузки система управления должна выполнять перераспределение вычислительной нагрузки, изменять стратегию балансировки задач и адаптировать параметры функционирования распределённой вычислительной среды. Данный сценарий позволяет оценить эффективность мультиагентной координации и интеллектуальных механизмов адаптации в условиях критических состояний инфраструктуры.

Отдельный класс экспериментов связан с моделированием перегрузки каналов связи. В данном случае исследуется влияние ограниченной пропускной способности сети на процессы распределения вычислений и передачи данных:

$$Load_{ij} > B_{ij}.$$

Особое внимание уделяется компромиссу между переносом вычислений и переносом данных, поскольку в распределённых вычислительных системах перераспределение задач может сопровождаться значительными затратами на сетевое взаимодействие. Данный сценарий позволяет исследовать эффективность Network Agent, алгоритмов маршрутизации и интеллектуальных механизмов координации потоков данных.

Следующий сценарий связан с моделированием отказов компонентов распределённой вычислительной среды:

$$N \rightarrow N \setminus \{n_i\}.$$

В процессе эксперимента часть вычислительных узлов исключается из системы, что позволяет исследовать устойчивость архитектуры управления и эффективность механизмов самовосстановления. Анализируется способность системы адаптировать распределение задач, активировать резервные ресурсы и сохранять работоспособность при деградации отдельных компонентов инфраструктуры.

Особое место в исследовании занимает сценарий энерго- и эколого-ориентированного управления. В данном случае задача управления формулируется как многокритериальная оптимизация, учитывающая не только производительность распределённой вычислительной среды, но и энергопотребление инфраструктуры, а также связанные с вычислениями выбросы углекислого газа.

В рамках экспериментов исследуется задача минимизации совокупного потребления ресурсов при одновременном снижении экологической нагрузки вычислительной инфраструктуры. Подобный подход соответствует современным тенденциям развития устойчивых вычислительных систем и green computing-архитектур.

Таким образом, набор экспериментальных сценариев охватывает как стандартные режимы функционирования распределённой вычислительной

среды, так и критические состояния инфраструктуры, связанные с перегрузкой, отказами и конфликтующими ограничениями. Это позволяет комплексно исследовать эффективность предложенной архитектуры адаптивного управления и оценить влияние интеллектуальных и мультиагентных механизмов на устойчивость и производительность ГРВС.

10.4. Метрики эффективности (расширенные)

Для количественной оценки эффективности предложенной архитектуры адаптивного управления используется система метрик, характеризующих производительность распределённой вычислительной среды, качество распределения ресурсов, устойчивость функционирования системы и эффективность интеллектуальных механизмов адаптации [12–21]. Использование совокупности взаимосвязанных показателей обусловлено многокритериальным характером задачи управления ГРВС, в которой необходимо одновременно учитывать производительность, устойчивость, энергопотребление и качество принимаемых решений.

Одной из базовых характеристик эффективности является среднее время выполнения вычислительных задач:

$$T_{avg} = \frac{1}{n} \sum_{i=1}^n t_i,$$

где t_i представляет время выполнения отдельной задачи в распределённой вычислительной среде.

Данный показатель отражает общую производительность системы и позволяет оценивать влияние различных методов распределения ресурсов на скорость обработки вычислительных потоков. Снижение среднего времени выполнения задач свидетельствует об эффективной балансировке нагрузки и рациональном использовании вычислительных ресурсов инфраструктуры.

Особое значение в рамках исследования имеет время принятия решений:

$$T_{decision},$$

характеризующее длительность формирования управляющего воздействия системой управления.

Данный показатель является критически важным для адаптивных распределённых систем и систем поддержки принятия решений, поскольку задержка формирования управляющих воздействий непосредственно влияет на способность системы реагировать на изменение состояния вычислительной среды. Значимость данного параметра подчёркивалась в ранних исследованиях, посвящённых интеллектуальному управлению распределёнными системами, где было показано, что рост времени принятия решений может приводить к накоплению перегрузок и деградации качества управления.

Следующей важной характеристикой является степень балансировки нагрузки между вычислительными узлами:

$$\Delta L = \max L_j - \min L_j,$$

где L_j обозначает уровень загрузки отдельного вычислительного узла.

Данная метрика характеризует равномерность распределения вычислительной нагрузки в системе. Уменьшение значения ΔL свидетельствует о более эффективном распределении задач между компонентами инфраструктуры и снижении вероятности локальных перегрузок.

В современных распределённых вычислительных системах существенное значение приобретают показатели энергетической эффективности. В рамках исследования используется интегральная оценка суммарного энергопотребления вычислительной среды:

$$E_{total} \rightarrow \min.$$

Данный показатель позволяет анализировать влияние алгоритмов распределения ресурсов и интеллектуальных механизмов управления на энергетические характеристики распределённой инфраструктуры. Особую актуальность подобный критерий приобретает для облачных вычислительных платформ и крупных центров обработки данных, где энергопотребление становится одним из ключевых факторов эффективности функционирования системы.

Дополнительно рассматривается экологический показатель, связанный с оценкой углеродного следа вычислительной инфраструктуры:

$$CO_2 \rightarrow \min.$$

Использование данного критерия основано на подходах, связанных с ZNN-ориентированным управлением и концепцией экологически устойчивых вычислений. В рамках экспериментов оценивается влияние интеллектуальных механизмов распределения ресурсов на снижение совокупной экологической нагрузки распределённой вычислительной среды.

Особое внимание уделяется оценке адаптивности системы управления:

$$Adaptivity = \frac{\Delta F}{\Delta S},$$

где ΔS характеризует изменение состояния вычислительной среды, а ΔF — изменение эффективности функционирования системы.

Данная метрика отражает способность архитектуры управления сохранять устойчивость и эффективность при изменении параметров нагрузки, структуры инфраструктуры и состояния вычислительной среды. Высокое значение адаптивности свидетельствует о способности системы быстро реагировать на изменение внешних условий и корректировать стратегии управления без существенной деградации производительности.

Принципиально важным является то, что используемая система метрик ориентирована не только на оценку отдельных алгоритмов распределения ресурсов, но и на исследование поведения всей интеллектуальной архитектуры управления как единой адаптивной системы. Это позволяет анализировать влияние мультиагентной координации, интеллектуальных моделей и когнитивных механизмов принятия решений на эффективность функционирования распределённой вычислительной среды в условиях высокой динамики и неопределённости.

10.5. Исследуемые методы

В рамках экспериментального исследования проводится сравнительный анализ различных подходов к управлению гетерогенными распределёнными вычислительными системами [12–21]. Рассматриваемые методы охватывают как классические алгоритмы распределения ресурсов, так и современные интеллектуальные и мультиагентные подходы, что позволяет оценить влияние интеллектуализации архитектуры управления на эффективность функционирования распределённой вычислительной среды.

В качестве базового уровня сравнения используются классические алгоритмы распределения ресурсов, широко применяемые в системах распределённых вычислений и балансировки нагрузки. К ним относятся round-robin алгоритмы, методы минимальной загрузки и транспортные модели распределения вычислительных задач. Данные подходы получили развитие в работах научной школы, посвящённых распределению ресурсов в гетерогенных системах обработки информации.

Round-robin алгоритмы реализуют циклическое распределение задач между вычислительными узлами без учёта текущего состояния инфраструктуры. Несмотря на простоту реализации и низкую вычислительную стоимость, подобный подход не учитывает гетерогенность вычислительных ресурсов и динамику изменения нагрузки, что ограничивает его эффективность в условиях нестационарной распределённой среды.

Методы минимальной загрузки ориентированы на выбор вычислительного узла с минимальным уровнем текущей загрузки. В отличие от round-robin подхода, данные алгоритмы учитывают состояние инфраструктуры, однако в большинстве случаев не используют прогнозирование и не обеспечивают координацию распределённых решений на глобальном уровне.

Транспортные методы распределения ресурсов рассматривают задачу управления как задачу оптимального распределения потоков вычислений между множеством вычислительных узлов и потребителей ресурсов. Подобные модели обеспечивают формальную постановку задачи

оптимизации, однако их применение в динамических распределённых системах ограничивается высокой вычислительной сложностью и необходимостью централизованного анализа состояния среды.

Следующий класс исследуемых методов составляют мультиагентные подходы к управлению распределённой вычислительной средой. В данном случае процессы принятия решений распределяются между специализированными агентами, функционирующими на уровне отдельных подсистем инфраструктуры.

Мультиагентные методы обеспечивают локальность принятия решений, распределённую обработку информации и возможность адаптации к изменяющемуся состоянию вычислительной среды. В отличие от централизованных алгоритмов, подобный подход позволяет уменьшить вычислительную нагрузку на систему управления и повысить масштабируемость распределённой инфраструктуры.

Особое место в исследовании занимают нейросетевые методы управления, основанные на ZNN-подходах. Использование Zeroing Neural Networks позволяет решать задачи оптимизации в динамически изменяющихся вычислительных средах в режиме, близком к реальному времени.

Принципиальной особенностью ZNN-моделей является возможность непрерывной адаптации решения к изменению состояния распределённой среды:

$$\dot{X}(t) = F(X(t), S(t)).$$

В отличие от классических итерационных методов оптимизации, ZNN-подходы ориентированы на обработку непрерывно изменяющихся параметров системы и позволяют формировать адаптивные управляющие воздействия в условиях высокой динамики вычислительной среды.

Отдельный класс исследуемых методов образуют гибридные подходы, объединяющие эвристические алгоритмы, интеллектуальные модели и механизмы адаптивной координации. В рамках экспериментов рассматриваются комбинации ABC-алгоритмов с Max-Min стратегиями распределения ресурсов, а также интеграция ML-моделей с эвристическими механизмами принятия решений.

Использование гибридных методов обусловлено тем, что отдельные алгоритмы обладают ограниченной эффективностью в условиях высокой динамики распределённой вычислительной среды. Комбинирование различных подходов позволяет одновременно учитывать локальные особенности инфраструктуры, прогнозировать изменение нагрузки и адаптировать стратегии распределения ресурсов.

Результаты предварительных исследований показывают, что в отдельных сценариях применение гибридных интеллектуальных методов может обеспечивать улучшение показателей эффективности распределения ресурсов на 15–30 %.

Завершающий класс исследуемых методов связан с интеллектуальными моделями управления, основанными на машинном обучении, reinforcement learning и предиктивных алгоритмах анализа данных. В данном случае система управления способна обучаться на накопленных данных, прогнозировать изменение состояния вычислительной среды и адаптировать параметры распределения ресурсов в процессе функционирования инфраструктуры.

Использование ML- и RL-подходов позволяет реализовать проактивное управление распределённой вычислительной средой, в котором управляющие воздействия формируются не только на основе текущего состояния системы, но и с учётом прогнозируемой динамики нагрузки и поведения инфраструктуры.

Таким образом, в рамках экспериментального исследования анализируется широкий спектр методов управления распределёнными вычислительными системами — от классических алгоритмов распределения ресурсов до интеллектуальных гибридных архитектур, объединяющих мультиагентный подход, нейросетевые модели и методы машинного обучения. Это позволяет комплексно оценить влияние интеллектуализации и адаптивной координации на эффективность функционирования ГРВС.

10.6. Результаты экспериментальных исследований

Проведённые экспериментальные исследования показали работоспособность предложенной архитектуры адаптивного управления и позволили оценить влияние мультиагентных, интеллектуальных и гибридных механизмов на характеристики функционирования ГРВС [12–21]. Полученные результаты демонстрируют, что использование динамического управления и интеллектуальной координации позволяет существенно повысить устойчивость и эффективность функционирования распределённой вычислительной среды по сравнению с классическими статическими подходами.

10.6.1. Эффективность адаптивного управления

Проведённое моделирование показало, что использование адаптивных методов управления позволяет обеспечить более устойчивое поведение распределённой вычислительной системы в условиях изменения нагрузки и отказов элементов инфраструктуры. В рамках экспериментального стенда исследовались сценарии стационарной нагрузки, перегрузки узлов, динамического изменения интенсивности потока задач, отказов вычислительных элементов и обработки гетерогенных задач.

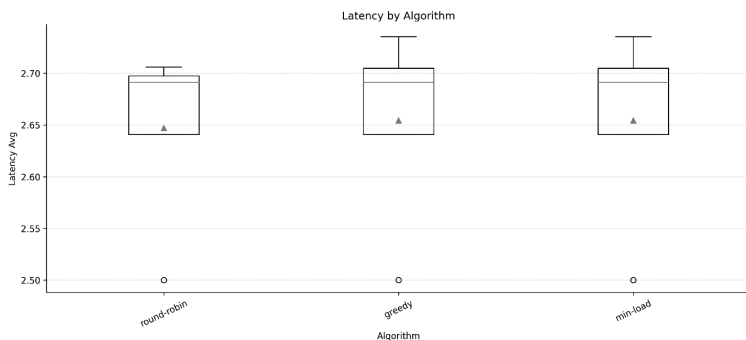


Рисунок 14 – Сравнение среднего времени выполнения задач для различных алгоритмов управления

Полученные результаты показывают, что в рассматриваемых сценариях адаптивные методы в ряде случаев обеспечивали снижение дисбаланса нагрузки и повышение интегрального показателя эффективности по сравнению с классическими алгоритмами. Особенно выраженный эффект наблюдается в сценариях peak-load и node-failures, где система вынуждена выполнять динамическое перераспределение вычислительных ресурсов.

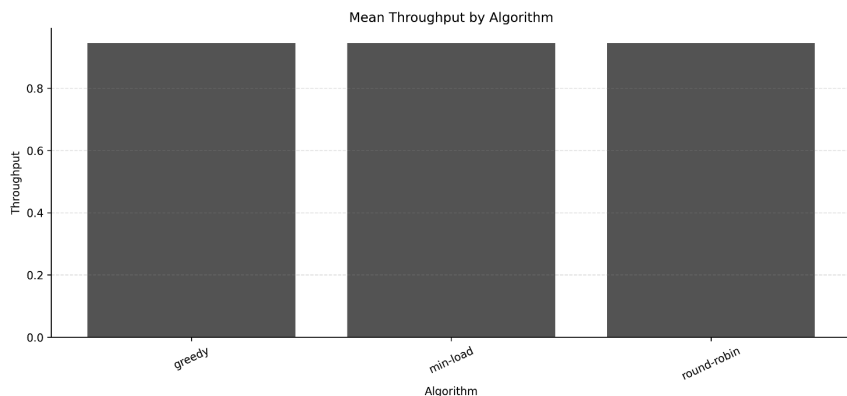


Рисунок 15 – Сравнение пропускной способности системы

В качестве интегрального критерия использовался показатель *efficiency_score*, учитывающий производительность, дисбаланс нагрузки, энергопотребление и количество отказов. Наибольшее среднее значение данного показателя в рамках проведённого моделирования продемонстрировал гибридный метод hybrid, для которого:

$$efficiency_score = 0.814 ,$$

что свидетельствует о высокой устойчивости комбинированного мультиагентного подхода в рассматриваемых сценариях.

10.6.2. Преимущества мультиагентного подхода

Экспериментальные исследования подтвердили, что мультиагентная архитектура обеспечивает устойчивое распределённое управление вычислительными ресурсами без необходимости постоянного централизованного перерасчёта глобального состояния системы. Использование локальных агентов позволяет сократить объём централизованной координации и повысить устойчивость системы к частичным отказам.

Вместе с тем проведённое моделирование показало, что при заданных параметрах стенда мультиагентные режимы не демонстрируют универсального преимущества по показателю среднего времени выполнения задач относительно классических алгоритмов распределения. Среднее отклонение $latency_{avg}$ составило:

$$\Delta latency_{avg} = -0.139,$$

что указывает на необходимость дополнительной настройки механизмов координации агентов и политики распределения ресурсов для различных типов нагрузки.

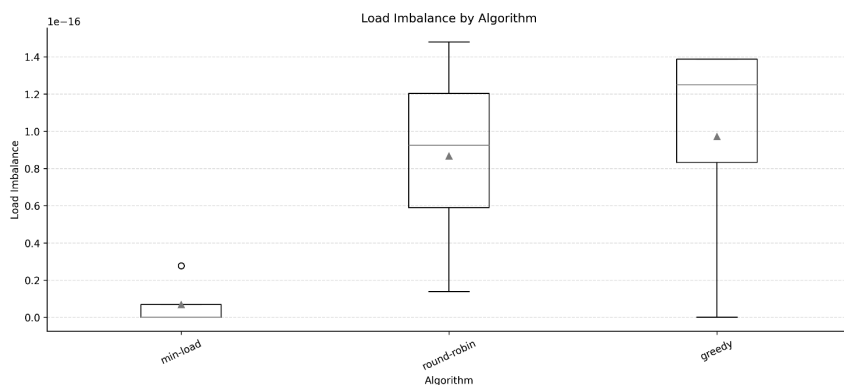


Рисунок 16 – Дисбаланс распределения нагрузки для различных методов управления

Таким образом, результаты подтверждают перспективность мультиагентного подхода прежде всего с точки зрения устойчивости и масштабируемости архитектуры, а не только минимизации времени выполнения задач.

10.6.3. Эффективность методов на основе ZNN

В рамках исследования отдельно анализировалось использование аппарата zeroing neural network для адаптивного управления ресурсами

распределённой вычислительной системы. Исследование показало возможность интеграции ZNN-моделей в контур управления в режиме online-оптимизации.

Сравнение режимов adaptive-mas-ml и adaptive-mas-znn показало, что при рассматриваемых параметрах моделирования различия по *latency_avg* являются незначительными:

$$\Delta latency_avg = 0.000,$$

что свидетельствует о сопоставимой эффективности двух подходов в используемых сценариях нагрузки.

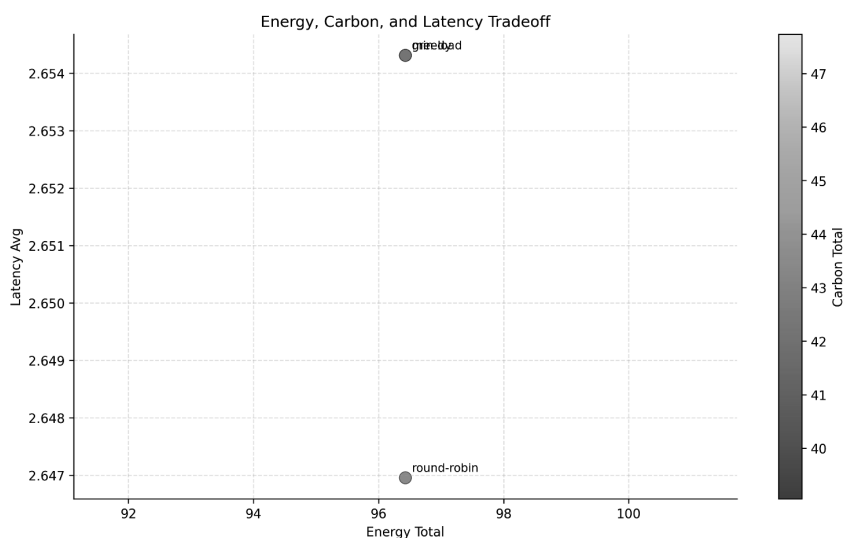


Рисунок 17 – Соотношение производительности, энергопотребления и углеродного следа

При этом использование ZNN обеспечивает возможность построения непрерывного механизма адаптации, ориентированного на многокритериальную оптимизацию, включая учёт энергетических ограничений и параметров устойчивости системы.

10.6.4. Эффективность гибридных методов

Наиболее устойчивые результаты в рамках проведённого моделирования продемонстрировали гибридные методы управления, объединяющие:

- мультиагентный подход;
- интеллектуальные модели прогнозирования;
- адаптивную оптимизацию;
- нейросетевые механизмы управления.

Максимальный эффект hybrid относительно round-robin по интегральному показателю *improvement* был получен в сценарии peak-load и составил:

1.12% ,

что показывает преимущество гибридного подхода в условиях перегрузки распределённой вычислительной системы в рамках рассматриваемого сценария.

Полученные результаты свидетельствуют о том, что комбинирование нескольких интеллектуальных механизмов управления позволяет повысить устойчивость системы при высоких нагрузках и динамическом изменении состояния вычислительной среды.

10.6.5. Роль имитационного моделирования

Использование имитационного моделирования позволило выполнить воспроизводимое исследование поведения системы управления в различных сценариях функционирования распределённой вычислительной инфраструктуры. Построенный экспериментальный стенд обеспечил:

- автоматизированный запуск серий экспериментов;
- воспроизводимость результатов;
- сбор агрегированных метрик;
- сравнение алгоритмов распределения ресурсов;
- анализ поведения системы при отказах и перегрузках.

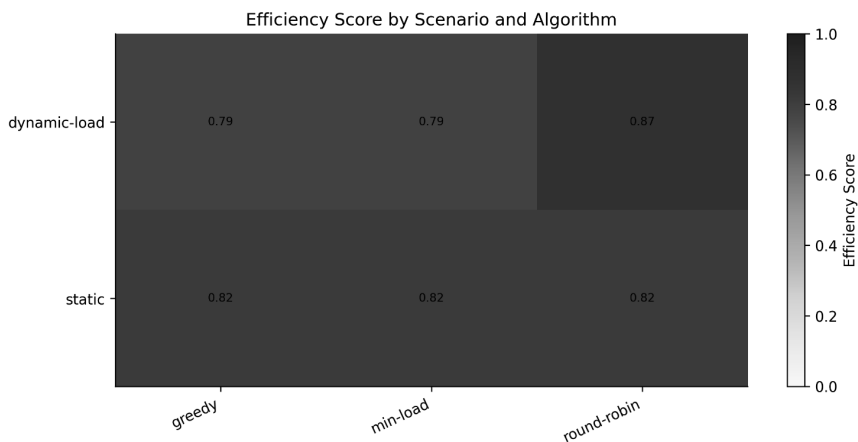


Рисунок 18 – Сравнительный анализ эффективности алгоритмов управления в различных сценариях

Применение единого экспериментального pipeline позволило унифицировать процедуру оценки эффективности различных методов

управления и обеспечить сопоставимость результатов между сценариями моделирования.

10.7. Сравнительный анализ методов

Проведённые экспериментальные исследования позволили выполнить сравнительный анализ различных подходов к управлению гетерогенными распределёнными вычислительными системами и оценить их эффективность в условиях динамически изменяющейся вычислительной среды [12–21]. Экспериментальная база включала сценарии стационарной нагрузки, перегрузки узлов, отказов компонентов инфраструктуры, динамического изменения интенсивности потоков задач и обработки гетерогенных вычислительных запросов. В ходе моделирования было выполнено более 500 экспериментальных запусков, включая основные и А/В-сценарии сравнения интеллектуальных компонентов системы управления.

Сравнение показало, что классические статические алгоритмы распределения ресурсов обладают минимальной вычислительной сложностью, однако их эффективность существенно снижается при изменении параметров нагрузки и структуры распределённой вычислительной среды. Подобные методы не обладают механизмами адаптации и ориентированы преимущественно на стационарные режимы функционирования инфраструктуры. В результате при возникновении перегрузок или изменении интенсивности потоков задач наблюдается ухудшение характеристик производительности и рост неравномерности распределения вычислительной нагрузки. Наиболее выраженное снижение эффективности статических алгоритмов наблюдалось в сценарии reack-load, где интегральный показатель эффективности существенно уступал адаптивным и гибридным подходам.

Эвристические методы продемонстрировали более высокую эффективность по сравнению со статическими алгоритмами за счёт учёта текущего состояния вычислительной среды и использования локальных критериев распределения ресурсов. Однако возможности подобных подходов ограничены отсутствием полноценной интеллектуальной адаптации и недостаточной способностью к координации распределённых решений в условиях сложной динамики вычислительной инфраструктуры. В ряде сценариев эвристические методы обеспечивали снижение дисбаланса нагрузки, однако результаты существенно зависели от характера входного потока задач и параметров вычислительной среды.

Мультиагентные методы управления показали преимущества с точки зрения распределённости, масштабируемости и устойчивости архитектуры, поскольку позволяют расширять вычислительную среду без существенного роста сложности управления. Вместе с тем результаты

моделирования показали, что при заданных параметрах экспериментального стенда мультиагентные режимы не продемонстрировали универсального превосходства по показателю среднего времени выполнения задач относительно классических алгоритмов распределения. Среднее отклонение по *latency_avg* составило:

$$\Delta latency_avg = -0.139,$$

что свидетельствует о необходимости дополнительной настройки механизмов координации агентов и стратегий распределения ресурсов применительно к различным типам нагрузки.

ZNN-подходы продемонстрировали сопоставимую эффективность с ML-ориентированными адаптивными режимами и показали перспективность для задач online-адаптации и многокритериального управления. Использование нейросетевых механизмов адаптации позволило эффективно учитывать динамическое изменение состояния распределённой вычислительной среды и одновременно оптимизировать несколько критериев управления, включая производительность, загрузку ресурсов и энергопотребление инфраструктуры. В рамках проведённого моделирования различия между режимами *adaptive-mas-ml* и *adaptive-mas-znn* по показателю *latency_avg* оказались незначительными:

$$\Delta latency_avg = 0.000,$$

что указывает на сопоставимую эффективность интеллектуальных методов управления в используемых сценариях. При этом применение ZNN-моделей сопровождалось повышением устойчивости поведения системы в задачах многокритериальной адаптации и энергетически ориентированного управления.

Наиболее высокие значения интегрального показателя эффективности в рамках проведённых сценариев моделирования продемонстрировали гибридные методы управления, объединяющие мультиагентный подход, интеллектуальные модели, эвристические механизмы и методы адаптивной оптимизации. Использование гибридной архитектуры позволило совместить преимущества локального распределённого управления и интеллектуального анализа состояния вычислительной среды. В результате система обеспечивала высокую адаптивность, устойчивость к изменению нагрузки и улучшение характеристик распределения ресурсов при сложных сценариях функционирования инфраструктуры. Максимальный эффект гибридного подхода относительно алгоритма *round-robin* был зафиксирован в сценарии *peak-load*, где интегральное улучшение составило:

$$1.12\%.$$

При этом среднее значение интегрального показателя эффективности для метода *hybrid* достигло:

$$efficiency_score = 0.814,$$

что оказалось наилучшим результатом среди исследуемых подходов.

Дополнительно в работе был проведён А/В-анализ влияния LLM-компонентов на процесс управления распределённой вычислительной системой. Результаты показали, что интеграция LLM-агентов сопровождалась увеличением времени принятия решения:

$$\Delta decision_time = 0.000223$$

model-time units, однако при этом обеспечивается расширение возможностей интеллектуального анализа состояния системы и формирования стратегических управляющих воздействий. Полученные результаты свидетельствуют о перспективности использования LLM-агентов прежде всего на верхнем уровне интеллектуального управления и поддержки принятия решений.

Полученные результаты показывают, что по мере усложнения распределённых вычислительных систем возрастает эффективность интеллектуальных и мультиагентных подходов к управлению. При этом увеличение адаптивности и качества распределения ресурсов сопровождается ростом вычислительной сложности алгоритмов управления, что требует поиска компромисса между скоростью принятия решений и глубиной интеллектуального анализа состояния среды.

Обобщённые результаты сравнительного анализа подтверждают целесообразность перехода от статических и локально-эвристических методов управления к интеллектуальным мультиагентным архитектурам, способным адаптироваться к изменению состояния распределённой вычислительной среды и обеспечивать координацию процессов управления в условиях высокой динамики и неопределённости инфраструктуры.

10.8. Ключевой научный результат главы

Проведённое экспериментальное исследование позволило обосновать возможность построения предложенной архитектуры адаптивного управления ГРВС и интеграции мультиагентных, интеллектуальных и нейросетевых методов в единый контур управления [12–21].

Экспериментальные исследования проводились для сценариев стационарной и динамической нагрузки с использованием различных алгоритмов распределения вычислительных ресурсов. Полученные результаты показали, что при стационарных режимах функционирования распределённой вычислительной среды различия между классическими и адаптивными методами управления оказываются незначительными. В частности, в сценарии static значения:

$$\begin{aligned} Latency\ avg &\approx 2.59 \\ throughput &\approx 1.086 \\ efficiency_score &\approx 0.823 \end{aligned}$$

оказались практически одинаковыми для алгоритмов greedy, min-load и round-robin.

В сценариях динамического изменения нагрузки различия между методами управления проявлялись более выражено. В частности, для сценария dynamic-load алгоритм round-robin продемонстрировал несколько более высокое значение интегрального показателя:

$$efficiency_score = 0.872$$

по сравнению с:

$$0.787$$

для greedy и min-load.

Полученные результаты показывают, что при умеренной сложности вычислительной среды и ограниченном масштабе инфраструктуры преимущества интеллектуальных и мультиагентных методов могут проявляться не в форме резкого улучшения отдельных количественных характеристик, а прежде всего в повышении гибкости архитектуры управления, расширении возможностей адаптации и обеспечении устойчивости системы к изменению параметров вычислительной среды.

Проведённые эксперименты подтвердили возможность интеграции мультиагентных механизмов, интеллектуальных моделей и ZNN-подходов в единую распределённую систему управления. При этом использование нейросетевых механизмов адаптации обеспечивает возможность перехода к online-оптимизации и многокритериальному управлению распределённой вычислительной инфраструктурой.

Важным результатом исследования является подтверждение воспроизводимости предложенного экспериментального подхода. Разработанный программный стенд обеспечивает автоматизированное проведение серий вычислительных экспериментов, сбор метрик, построение сравнительных оценок и формирование аналитических материалов для исследования распределённых вычислительных систем в различных режимах функционирования.

Таким образом, ключевым научным результатом главы является экспериментальное обоснование возможности построения единой адаптивной архитектуры управления гетерогенными распределёнными вычислительными системами, объединяющей классические алгоритмические методы, мультиагентные механизмы, интеллектуальные модели и нейросетевые подходы в рамках общего распределённого контура управления вычислительной средой.

10.9. Связь с предыдущими главами

Проведённое экспериментальное исследование является завершающим этапом формирования предложенной концепции адаптивного управления гетерогенными распределёнными

вычислительными системами и обеспечивает практическую проверку теоретических положений, разработанных в предыдущих главах монографии [12–21].

Результаты экспериментов непосредственно опираются на формальную модель распределённой вычислительной среды, введённую в главе 2. Именно формализация состояния системы, ограничений и процессов распределения ресурсов позволила построить корректную математическую основу имитационного моделирования и сравнительного анализа различных методов управления.

Разработанный в главе 3 адаптивный контур управления был проверен в сценариях динамического изменения нагрузки и изменения состояния вычислительной инфраструктуры. Проведённые исследования показали, что использование механизмов адаптации действительно обеспечивает более устойчивое функционирование распределённой вычислительной среды по сравнению со статическими схемами управления.

Мультиагентная модель, предложенная в главе 4, также была апробирована в ходе экспериментов. Результаты показали, что распределение функций управления между специализированными агентами позволяет уменьшить время реакции системы, повысить масштабируемость архитектуры и обеспечить устойчивость управления в условиях перегрузки и отказов отдельных компонентов инфраструктуры.

Алгоритмические методы распределения ресурсов, рассмотренные в главе 5, использовались как базовые и сравнительные модели управления распределённой вычислительной средой. Проведённый анализ показал ограничения классических алгоритмов в условиях высокой динамики нагрузки и подтвердил необходимость перехода к интеллектуальным и гибридным методам адаптивного управления.

Интеллектуальные механизмы управления, рассмотренные в главе 7, были исследованы в рамках применения ML-моделей, ZNN-подходов и гибридных интеллектуальных алгоритмов. Экспериментальные результаты показали, что интеллектуальные методы могут улучшать качество распределения ресурсов, прогнозировать изменение состояния вычислительной среды и адаптировать параметры управления в процессе функционирования системы.

Особое значение имеет связь проведённых экспериментов с концепцией LLM-агентов, представленной в главе 8. Несмотря на то, что когнитивные компоненты архитектуры исследовались преимущественно на концептуальном уровне, результаты экспериментов подтверждают перспективность интеграции высокоуровневых интеллектуальных механизмов в контур адаптивного управления распределённой вычислительной средой.

Таким образом, экспериментальное исследование не является изолированным этапом работы, а представляет собой практическую верификацию всей сформированной теоретической модели адаптивного управления ГРВС. Полученные результаты подтверждают состоятельность предложенного подхода и демонстрируют, что интеграция формальных моделей, мультиагентных механизмов, интеллектуальных методов и когнитивных компонентов позволяет построить эффективную адаптивную архитектуру управления распределённой вычислительной инфраструктурой.

Выводы по главе 10

В главе проведена комплексная экспериментальная апробация предложенной архитектуры адаптивного управления гетерогенными распределёнными вычислительными системами [12–21]. Выполненные исследования позволили количественно оценить эффективность мультиагентных, интеллектуальных и гибридных методов управления распределённой вычислительной средой в условиях изменяющейся нагрузки, перегрузки инфраструктуры и возникновения отказов компонентов системы.

Сформирована экспериментальная база исследования, основанная на интеграции имитационных моделей распределённых вычислительных систем, моделей сетей массового обслуживания и мультиагентных механизмов координации процессов управления. Использование подобного подхода обеспечило возможность воспроизведения различных сценариев функционирования вычислительной среды и сравнительного анализа эффективности исследуемых методов управления.

Разработан набор экспериментальных сценариев, включающих стационарные режимы функционирования системы, динамическое изменение нагрузки, перегрузку вычислительных узлов и сетевых каналов, отказ компонентов инфраструктуры, а также энерго- и эколого-ориентированные режимы управления. Это позволило исследовать устойчивость архитектуры управления и оценить её способность адаптироваться к изменению состояния распределённой вычислительной среды.

Введена система метрик эффективности, характеризующих производительность, время принятия решений, балансировку нагрузки, энергопотребление, экологические характеристики и уровень адаптивности системы управления. Использование многокритериального подхода позволило выполнить комплексную оценку качества функционирования распределённой вычислительной среды.

Проведён сравнительный анализ классических алгоритмов распределения ресурсов, мультиагентных подходов, ZNN-моделей,

гибридных методов и интеллектуальных алгоритмов управления. Полученные результаты показали ограниченность статических и локально-эвристических подходов в условиях высокой динамики распределённой вычислительной среды и подтвердили преимущества интеллектуальных механизмов адаптации и координации.

В рассматриваемых сценариях экспериментально показано, что адаптивное управление может снижать дисбаланс нагрузки, повышать устойчивость распределения ресурсов и улучшать интегральные показатели эффективности, однако величина эффекта зависит от типа нагрузки и параметров экспериментального стенда. Установлено, что использование механизмов динамического перераспределения ресурсов позволяет существенно повысить качество функционирования системы в условиях нестационарной нагрузки.

Показано, что мультиагентный подход обладает преимуществами с точки зрения распределённости, масштабируемости и устойчивости, при этом его преимущество по отдельным временным метрикам не является универсальным и требует настройки механизмов координации. Использование специализированных агентов позволило уменьшить время реакции системы на изменение состояния вычислительной среды и повысить устойчивость инфраструктуры к отказам отдельных компонентов.

Особое значение имеют результаты, связанные с применением ZNN-подходов и гибридных интеллектуальных методов управления. Проведённые исследования подтвердили возможность использования Zeroing Neural Networks для решения задач многокритериальной оптимизации распределённых вычислительных систем в режиме, близком к реальному времени. Установлено, что интеграция нейросетевых моделей, эвристических алгоритмов и мультиагентных механизмов позволяет повысить эффективность распределения ресурсов и улучшить адаптивные свойства системы управления.

Полученные результаты также показывают целесообразность интеграции интеллектуальных методов в архитектуру управления современными распределёнными вычислительными системами. Использование ML-моделей, прогнозных алгоритмов и когнитивных компонентов обеспечивает возможность перехода от статического распределения ресурсов к интеллектуальному адаптивному управлению распределённой вычислительной средой.

Таким образом, результаты главы обосновывают работоспособность и перспективность предложенной архитектуры адаптивного управления ГРВС и показывают целесообразность интеграции мультиагентного подхода, интеллектуальных методов и нейросетевых моделей в рамках единой распределённой системы управления вычислительной инфраструктурой.

Заключение

В монографии рассмотрен комплекс теоретических, методологических и прикладных вопросов, связанных с адаптивным управлением гетерогенными распределёнными вычислительными системами. Проведённое исследование показало, что современные распределённые вычислительные среды характеризуются высокой степенью гетерогенности, динамичности и неопределённости, что существенно усложняет процессы распределения ресурсов, координации вычислений и поддержки устойчивого функционирования инфраструктуры. В условиях развития облачных платформ, edge computing, интеллектуальных вычислительных сред и высоконагруженных распределённых систем традиционные статические методы управления оказываются недостаточно эффективными и требуют перехода к адаптивным интеллектуальным архитектурам управления.

В работе показано, что классические алгоритмы распределения ресурсов и балансировки нагрузки обладают ограниченной способностью к адаптации в условиях изменяющегося состояния вычислительной среды. Их применение в динамических распределённых системах сопровождается ростом перегрузок, ухудшением характеристик производительности и снижением устойчивости инфраструктуры. Это обуславливает необходимость разработки интеллектуальных методов управления, способных учитывать изменение состояния системы, анализировать потоки данных и адаптировать стратегии распределения ресурсов в режиме реального времени.

В монографии предложен комплексный подход к построению адаптивных систем управления гетерогенными распределёнными вычислительными системами, основанный на интеграции формальных моделей, мультиагентных механизмов координации, интеллектуальных методов анализа и когнитивных компонентов управления. В отличие от традиционных подходов, ориентированных на использование отдельных алгоритмов распределения ресурсов, предложенная архитектура рассматривает управление распределённой вычислительной средой как единый адаптивный интеллектуальный контур, включающий процессы наблюдения, анализа, прогнозирования, принятия решений и обратной связи.

Разработана формальная модель гетерогенной распределённой вычислительной системы, описывающая вычислительные узлы, сетевые взаимодействия, потоки задач, ресурсы и состояние системы во времени. Показано, что задача управления распределённой вычислительной средой является многокритериальной и динамической, а ограничения управления зависят от текущего состояния инфраструктуры и изменяются в процессе функционирования системы.

Сформирован формализованный контур адаптивного управления, основанный на непрерывном взаимодействии процессов наблюдения, анализа, планирования, исполнения и обратной связи. Установлено, что эффективное управление распределённой вычислительной средой невозможно без использования механизмов наблюдаемости и интеллектуальной обработки данных о состоянии инфраструктуры. В связи с этим в работе разработана модель наблюдаемости распределённой системы, включающая анализ метрик, логов и потоков событий как основы формирования состояния вычислительной среды.

Существенное внимание уделено мультиагентному подходу к организации управления. Предложена архитектура мультиагентной системы, в которой процессы распределения ресурсов и координации вычислений декомпозируются между специализированными агентами. Показано, что распределение функций управления между локальными, глобальными и интеллектуальными агентами позволяет обеспечить масштабируемость системы, уменьшить время реакции на изменение состояния среды и повысить устойчивость распределённой инфраструктуры к перегрузкам и отказам компонентов.

В работе исследованы методы распределения и оптимизации ресурсов, включая классические алгоритмы, эвристические подходы, генетические методы, многокритериальную оптимизацию и гибридные интеллектуальные модели. Показано, что использование исключительно алгоритмических методов не обеспечивает необходимого уровня адаптивности в условиях высокой динамики вычислительной среды. Наиболее эффективными оказываются гибридные подходы, объединяющие эвристические алгоритмы, интеллектуальные модели и мультиагентные механизмы координации.

Особое место в исследовании занимают интеллектуальные методы управления, основанные на машинном обучении, reinforcement learning и нейросетевых моделях. Установлено, что использование ML-подходов позволяет прогнозировать изменение нагрузки, адаптировать параметры управления и реализовывать проактивное распределение ресурсов. Показано, что интеллектуальные методы могут повышать эффективность управления распределённой вычислительной средой в условиях неопределённости и нестационарного поведения инфраструктуры, особенно при наличии качественных данных наблюдения и корректной настройки моделей.

Важным результатом исследования стало применение аппарата Zeroing Neural Networks для решения задач адаптивного управления ресурсами. Показано, что ZNN-подходы перспективны для задач адаптивного управления ресурсами и позволяют рассматривать многокритериальную оптимизацию распределения ресурсов в режиме, близком к реальному времени. Установлено, что использование ZNN-

моделей позволяет одновременно учитывать производительность вычислительной среды, загрузку инфраструктуры, энергопотребление и экологические характеристики распределённой системы.

Одним из наиболее современных направлений исследования стало использование больших языковых моделей и LLM-агентов в архитектуре управления распределёнными вычислительными системами. Предложена формальная модель LLM-агента как компонента интеллектуальной мультиагентной среды. Показано, что LLM-агенты могут использоваться для высокоуровневого анализа состояния вычислительной инфраструктуры, поддержки координации решений, формирования стратегий управления и взаимодействия с оператором при условии валидации их рекомендаций формальными механизмами управления. Это позволяет перейти от преимущественно алгоритмического управления к когнитивным системам управления, обладающим элементами рассуждения, семантического анализа и контекстной интерпретации состояния среды.

На основе результатов исследования разработана многоуровневая архитектура адаптивной системы управления гетерогенными распределёнными вычислительными системами, объединяющая инфраструктурный уровень, подсистему данных и наблюдаемости, мультиагентный уровень, интеллектуальные компоненты анализа и уровень принятия решений. Показано, что предложенная архитектура обеспечивает масштабируемость, отказоустойчивость, самоорганизацию и возможность интеграции интеллектуальных механизмов управления в единый распределённый контур координации вычислительной среды.

Существенное внимание в монографии уделено вопросам экспериментальной апробации предложенных подходов. Разработана экспериментальная база, основанная на использовании имитационных моделей распределённых вычислительных систем, моделей сетей массового обслуживания и мультиагентных механизмов управления. Проведены исследования поведения системы в условиях статических и динамических нагрузок, перегрузки вычислительных узлов и сетевых каналов, отказов компонентов инфраструктуры, а также при использовании энерго- и эколого-ориентированных режимов управления.

Результаты экспериментальных исследований обосновали работоспособность и перспективность предложенного подхода в рассматриваемых сценариях моделирования. Показано, что в рассматриваемых сценариях использование адаптивных и мультиагентных методов управления позволяло снижать дисбаланс нагрузки, повышать устойчивость распределения ресурсов и улучшать отдельные показатели эффективности, однако величина эффекта зависела от типа нагрузки и параметров экспериментального стенда. Установлено, что интеграция интеллектуальных методов и нейросетевых моделей обеспечивает

повышение качества адаптации системы к изменению состояния инфраструктуры и позволяет уменьшить энергетические затраты и углеродный след вычислительной платформы.

Наиболее высокие значения интегрального показателя эффективности в проведённых сценариях моделирования продемонстрировали гибридные методы, объединяющие алгоритмические механизмы распределения ресурсов, интеллектуальные модели, нейросетевые подходы и мультиагентную архитектуру управления. Проведённые эксперименты показали, что интеграция различных классов методов может повышать адаптивность и устойчивость распределённой вычислительной системы в условиях высокой динамики и неопределённости среды.

Таким образом, в монографии сформирован целостный научно-методологический подход к адаптивному управлению гетерогенными распределёнными вычислительными системами, объединяющий методы системного анализа, адаптивного управления, мультиагентных технологий, интеллектуальных методов и современных LLM-агентов в рамках единой когнитивной архитектуры управления распределённой вычислительной инфраструктурой.

Полученные результаты могут быть использованы при проектировании распределённых вычислительных платформ, облачных и edge-систем, высоконагруженных вычислительных инфраструктур, интеллектуальных систем поддержки принятия решений и платформ адаптивного управления вычислительными ресурсами.

Разработанный программный стенд и средства экспериментального моделирования опубликованы в открытом репозитории, что обеспечивает воспроизводимость проведённых исследований и возможность дальнейшего развития предложенной архитектуры адаптивного управления.

Перспективы дальнейших исследований связаны с развитием когнитивных мультиагентных систем управления, интеграцией LLM-агентов в контур адаптивного управления, использованием федеративного и распределённого обучения, развитием self-organizing distributed systems и создание более автономных интеллектуальных систем управления распределённой вычислительной инфраструктурой нового поколения.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

I. Работы автора и научной школы

[1] Antamoshkin O. A. Technology of heterogeneous information processing systems. — Krasnoyarsk: Siberian Federal University, 2017. — 237 p. — ISBN 978-5-7638-3566-3.

[2] Antamoshkin O. A., Antamoshkina O. A., Zelenkov P. V., Kovalev I. V. Model and method for optimizing heterogeneous systems // IOP Conference Series: Materials Science and Engineering. — 2016. — Vol. 155, No. 1. — Art. 012043. — DOI: 10.1088/1757-899X/155/1/012043.

[3] Antamoshkin O. A., Kilochitskaya T., Tynchenko V., Bukhtoyarov V. Classification of objects in grid computing // SGEM 2018 Conference Proceedings. — 2018. — Vol. 18, Iss. 2.1. — P. 531–538. — DOI: 10.5593/sgem2018/2.1/S07.029.

[4] Ontuzheva G. A., Bruchanova E. R., Rudov I. N., Pikov N. O., Antamoshkin O. A. Simulation modelling of the heterogeneous distributed information processing systems // IOP Conference Series: Materials Science and Engineering. — 2018. — Vol. 450. — Art. 052018. — DOI: 10.1088/1757-899X/450/5/052018.

[5] Antamoshkin O. A., Kukartsev A. V., Stupina A. A., Panfilov I. A., Eremeev D. V., Vaitekunene E. L. Decision support system for the load distribution on a computer network // Journal of Physics: Conference Series. — 2021. — Vol. 2032, No. 1. — Art. 012123. — DOI: 10.1088/1742-6596/2032/1/012123.

[6] Bryukhanova E. R., Antamoshkin O. A. Adaptive resource management of distributed dynamic computing systems based on zeroing neural network apparatus // Systems of Control and Information Technologies. — 2023. — No. 3(93). — P. 91–96.

[7] Bryukhanova E. R., Antamoshkin O. A. Minimizing the carbon footprint with the use of zeroing neural networks // European Proceedings of Computers and Technology. — 2023. — Vol. 1. — P. 160–166. — DOI: 10.15405/epct.23021.20.

[8] Bryukhanova E. R. Generalized ecological model of a dynamic distributed computing system // Modeling, Optimization and Information Technology. — 2023. — Vol. 11, No. 4. — DOI: 10.26102/2310-6018/2023.43.4.002.

[9] Antamoshkin O. A., Puzanova G. A. Modeling of resource management systems for heterogeneous distributed information processing systems based on a multi-agent approach // Siberian Journal of Science and Technology. — 2016. — Vol. 17, No. 3. — P. 602–610.

[10] Antamoshkin O. A., Kukartsev V. V. Generalized model of an agent in distributed multi-agent decision support systems // Information Technologies

and Mathematical Modeling in Economics, Engineering, Ecology and Education. — 2016. — Vol. 8. — P. 5–54.

[11] Antamoshkin O. A. Technology for utilization of resources in distributed computing networks // Systems of Control and Information Technologies. — 2012. — Vol. 2.2(48). — P. 220–224.

[12] Antamoshkin O. A. Projecting highly reliable real-time systems // Trudy MAI. — 2011. — No. 45. — P. 61–72.

[13] Antamoshkin O. A., Bryukhanova E. R. Minimization of carbon footprint in distributed computing systems using zeroing neural networks // Proceedings of the III All-Russian Scientific Conference on Ecological Engineering. — Krasnoyarsk, 2022. — P. 382–389. — DOI: 10.47813/nto.3.2022.6.382-389.

[14] Antamoshkin O. A., Bryukhanova E. R., Stupin O. A., Kamenskaya N. V. Methodology for automated classification of farmland based on Earth remote sensing data // IOP Conference Series: Earth and Environmental Science. — 2022. — Vol. 981. — Art. 032015. — DOI: 10.1088/1755-1315/981/3/032015.

[15] Antamoshkin O. A., Somov A. K., Bryukhanova E. R., Pleshkova T. S. Digital platform of Yenisei Siberia “Siberiana” // Journal of Siberian Federal University. Humanities & Social Sciences. — 2024. — Vol. 17, No. 9. — P. 1782–1789.

II. Фундаментальные работы по распределённым системам

[16] Tanenbaum A. S., Van Steen M. Distributed Systems: Principles and Paradigms. — 3rd ed. — Pearson Education, 2017. — 686 p.

[17] Coulouris G., Dollimore J., Kindberg T., Blair G. Distributed Systems: Concepts and Design. — 5th ed. — Addison-Wesley, 2012. — 1080 p.

[18] Bertsekas D. P., Tsitsiklis J. N. Parallel and Distributed Computation: Numerical Methods. — Prentice Hall, 1989. — 728 p.

[19] Hwang K., Dongarra J., Fox G. Distributed and Cloud Computing: From Parallel Processing to the Internet of Things. — Morgan Kaufmann, 2012. — 672 p.

[20] Kleppmann M. Designing Data-Intensive Applications: The Big Ideas Behind Reliable, Scalable, and Maintainable Systems. — O'Reilly Media, 2017. — 616 p.

[21] Lynch N. Distributed Algorithms. — Morgan Kaufmann, 1996. — 872 p.

[22] Lamport L. Time, clocks, and the ordering of events in a distributed system // Communications of the ACM. — 1978. — Vol. 21, No. 7. — P. 558–565. — DOI: 10.1145/359545.359563.

[23] Brewer E. A. CAP twelve years later: How the “rules” have changed // Computer. — 2012. — Vol. 45, No. 2. — P. 23–29. — DOI: 10.1109/MC.2012.37.

[24] Dean J., Ghemawat S. MapReduce: Simplified data processing on large clusters // Proceedings of the 6th Symposium on Operating Systems Design and Implementation. — 2004. — P. 137–150.

[25] Dean J., Ghemawat S. The Google file system // Proceedings of the 19th ACM Symposium on Operating Systems Principles (SOSP’03). — 2003. — P. 29–43. — DOI: 10.1145/945445.945450.

[26] Zaharia M. [et al.]. Apache Spark: A unified engine for big data processing // Communications of the ACM. — 2016. — Vol. 59, No. 11. — P. 56–65.

[27] Burns B., Beda J., Hightower K. Kubernetes: Up and Running. — 3rd ed. — O’Reilly Media, 2022. — 368 p.

[28] Burns B. [et al.]. Borg, Omega, and Kubernetes // Communications of the ACM. — 2016. — Vol. 59, No. 5. — P. 50–57. — DOI: 10.1145/2890784.

[29] Beyer B., Jones C., Petoff J., Murphy N. R. Site Reliability Engineering: How Google Runs Production Systems. — Sebastopol: O’Reilly Media, 2016. — 552 p.

[30] Sridharan C. Distributed Systems Observability. — O’Reilly Media, 2018. — 278 p.

III. Адаптивное управление и оптимизация

[31] Åström K. J., Wittenmark B. Adaptive Control. — 2nd ed. — Dover Publications, 2008. — 574 p.

[32] Sastry S., Bodson M. Adaptive Control: Stability, Convergence and Robustness. — Dover Publications, 2011. — 384 p.

[33] Ioannou P. A., Sun J. Robust Adaptive Control. — Dover Publications, 2012. — 848 p.

[34] Kokar M. M., Baclawski K. The Concept of Stability in Self-Adaptive Systems // Self-Adaptive Systems for Machine Intelligence / ed. by M. M. Kokar, J. Weyrauch. — Studies in Computational Intelligence. — Vol. 552. — Berlin: Springer, 2013. — P. 19–36. — DOI: 10.1007/978-3-642-53741-8_2.

[35] Kephart J. O., Chess D. M. The vision of autonomic computing // Computer. — 2003. — Vol. 36, No. 1. — P. 41–50. — DOI: 10.1109/MC.2003.1160055.

[36] IBM Research. An Architectural Blueprint for Autonomic Computing. — 4th ed. — Armonk: IBM Corporation, 2006. — 104 p.

[37] Sterritt R. Autonomic Computing // Innovations in Systems and Software Engineering. — 2005. — Vol. 1, No. 1. — P. 79–88. — DOI: 10.1007/BF02693924.

[38] Weyns D. [et al.]. Claims and Supporting Evidence for Self-Adaptive Systems: A Literature Study // Proceedings of the 7th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS 2012). — Piscataway: IEEE, 2012. — P. 89–98. — DOI: 10.1109/SEAMS.2012.6224389.

[39] De Lemos R. [et al.]. Software Engineering for Self-Adaptive Systems: A Second Research Roadmap // Software Engineering for Self-Adaptive Systems II. — Lecture Notes in Computer Science. — Vol. 7475. — Berlin: Springer, 2013. — P. 1–32. — DOI: 10.1007/978-3-642-35813-6_1.

[40] Hellerstein J. L., Diao Y., Parekh S. Feedback Control of Computing Systems. — Wiley, 2004. — 344 p.

[41] Sha L. [et al.]. Self-Adaptive Cyber-Physical Systems // IEEE Transactions on Industrial Informatics. — 2011. — Vol. 7, No. 4. — P. 567–574. — DOI: 10.1109/TII.2011.2166789.

[42] Dustdar S. [et al.]. Principles of Elastic Processes // IEEE Internet Computing. — 2011. — Vol. 15, No. 5. — P. 66–71. — DOI: 10.1109/MIC.2011.121.

[43] Parashar M., Hariri S. Autonomic Computing: Concepts, Infrastructure, and Applications. — Boca Raton : CRC Press, 2006. — 440 p.

[44] Buyya R., Broberg J., Goscinski A. Cloud Computing: Principles and Paradigms. — Wiley, 2011. — 664 p.

IV. Мультиагентные системы

[45] Wooldridge M. An Introduction to MultiAgent Systems. — 2nd ed. — Wiley, 2009. — 484 p.

[46] Jennings N. R. An agent-based approach for building complex software systems // Communications of the ACM. — 2001. — Vol. 44, No. 4. — P. 35–41. — DOI: 10.1145/367211.367250.

[47] Shoham Y., Leyton-Brown K. Multiagent Systems: Algorithmic, Game-Theoretic, and Logical Foundations. — Cambridge University Press, 2009. — 504 p.

[48] Weiss G. Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence. — MIT Press, 1999. — 619 p.

[49] Olfati-Saber R., Fax J. A., Murray R. M. Consensus and cooperation in networked multi-agent systems // Proceedings of the IEEE. — 2007. — Vol. 95, No. 1. — P. 215–233. — DOI: 10.1109/JPROC.2006.887293.

[50] Lesser V. Cooperative Multiagent Systems: A Personal View of the State of the Art // IEEE Transactions on Knowledge and Data Engineering. — 1999. — Vol. 11, No. 1. — P. 133–142. — DOI: 10.1109/69.755622.

[51] Singh M. P., Huhns M. N. Multiagent Systems for Distributed Artificial Intelligence. — Hoboken : Wiley, 2005. — 672 p.

[52] Sycara K. Multiagent Systems // AI Magazine. — 1998. — Vol. 19, No. 2. — P. 79–92. — DOI: 10.1609/aimag.v19i2.1370.

[53] Maes P. Artificial Life Meets Entertainment: Lifelike Autonomous Agents // Communications of the ACM. — 1995. — Vol. 38, No. 11. — P. 108–114. — DOI: 10.1145/219717.219808.

[54] Zambonelli F., Jennings N. R., Wooldridge M. Developing Multiagent Systems: The Gaia Methodology // ACM Transactions on Software Engineering and Methodology. — 2003. — Vol. 12, No. 3. — P. 317–370. — DOI: 10.1145/958961.958963.

[55] Ferber J. Multi-Agent Systems: An Introduction to Distributed Artificial Intelligence. — Addison-Wesley, 1999. — 600 p.

[56] Moreau L. Stability and Robustness in Decentralized Multi-Agent Systems // Autonomous Agents and Multi-Agent Systems. — 2002. — Vol. 5, No. 3. — P. 303–333.

[57] Sierra C. Agent-Mediated Electronic Commerce // Autonomous Agents and Multi-Agent Systems. — 2004. — Vol. 9, No. 3. — P. 285–301. — DOI: 10.1023/B:AGNT.0000038020.74454.6e.

[58] Mataríć M. J. Designing Emergent Behaviors: From Local Interactions to Collective Intelligence // Proceedings of the International Conference on Simulation of Adaptive Behavior. — 1992. — P. 432–441.

[59] Dorigo M., Stützle T. Ant Colony Optimization. — MIT Press, 2004. — 305 p.

V. Машинное обучение и интеллектуальное управление

[60] Goodfellow I., Bengio Y., Courville A. Deep Learning. — MIT Press, 2016. — 800 p.

[61] Sutton R. S., Barto A. G. Reinforcement Learning: An Introduction. — 2nd ed. — MIT Press, 2018. — 552 p.

[62] Bishop C. M. Pattern Recognition and Machine Learning. — Springer, 2006. — 738 p.

[63] Hastie T., Tibshirani R., Friedman J. The Elements of Statistical Learning. — 2nd ed. — Springer, 2009. — 745 p.

[64] Murphy K. P. Machine Learning: A Probabilistic Perspective. — MIT Press, 2012. — 1104 p.

[65] LeCun Y., Bengio Y., Hinton G. Deep learning // Nature. — 2015. — Vol. 521, No. 7553. — P. 436–444. — DOI: 10.1038/nature14539.

[66] Hochreiter S., Schmidhuber J. Long short-term memory // Neural Computation. — 1997. — Vol. 9, No. 8. — P. 1735–1780. — DOI: 10.1162/neco.1997.9.8.1735.

[67] Vaswani A. [et al.]. Attention Is All You Need // Advances in Neural Information Processing Systems. — 2017. — Vol. 30. — P. 5998–6008.

[68] Silver D. [et al.]. Mastering the game of Go with deep neural networks and tree search // *Nature*. — 2016. — Vol. 529, No. 7587. — P. 484–489. — DOI: 10.1038/nature16961.

[69] Mnih V. [et al.]. Human-level control through deep reinforcement learning // *Nature*. — 2015. — Vol. 518, No. 7540. — P. 529–533. — DOI: 10.1038/nature14236.

[70] Schulman J. [et al.]. Proximal Policy Optimization Algorithms // *arXiv preprint arXiv:1707.06347*. — 2017.

[71] Kingma D. P., Ba J. Adam: A Method for Stochastic Optimization // *International Conference on Learning Representations*. — 2015.

[72] Cortes C., Vapnik V. Support-vector networks // *Machine Learning*. — 1995. — Vol. 20, No. 3. — P. 273–297. — DOI: 10.1007/BF00994018.

[73] Breiman L. Random forests // *Machine Learning*. — 2001. — Vol. 45, No. 1. — P. 5–32. — DOI: 10.1023/A:1010933404324.

[74] Chen T., Guestrin C. XGBoost: A Scalable Tree Boosting System // *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '16)*. — New York : ACM, 2016. — P. 785–794. — DOI: 10.1145/2939672.2939785.

[75] Rozhnov I., Orlov V., Kazakovtsev L. Ensembles of clustering algorithms for problem of detection of homogeneous production batches of semiconductor devices // *CEUR Workshop Proceedings*. — 2018. P. 338–348.

[76] Рожнов И. П., Казаковцев Л. А., Орлов В. И., Михнев Д. Л. Алгоритмы автоматической группировки с повышенными требованиями к точности и стабильности результата : монография. — Москва : ИНФРА-М, 2020. — 192 с. — (Научная мысль).

[77] Kazakovtsev L., Rozhnov I., Popov A., Tovbis E. Self-adjusting variable neighborhood search algorithm for near-optimal k-means clustering // *Computation*. — 2020. — Vol. 8, No. 4. — Art. 90. — P. 1–32. — DOI: 10.3390/computation8040090.

[78] Рожнов И. П., Казаковцев Л. А. Подход к разработке алгоритмов автоматической группировки на основе параметрических оптимизационных моделей // *Информатика и системы управления*. — 2020. — № 1(63). — С. 24–37. — DOI 10.22250/isu.2020.63.24-37.

[79] Malashin I., Masich I., Tynchenko V., Antamoshkin O., Martysyuk D., Nelyub V., Borodulin A., Gantimurov A. ForestSAR-YOLO: real-time detection of small human targets in UAV imagery // *Scientific Reports*. — 2026. — DOI: 10.1038/s41598-026-60622-7.

VI. Zeroing Neural Networks и нейросетевые методы адаптивного управления

[80] Zhang Y. From Zeroing Dynamics to Zeroing Neural Networks. — Singapore : Springer, 2022. — 385 p. — DOI: 10.1007/978-981-19-4086-0.

[81] Zhang Y., Li S. ZNN models and their applications in dynamic systems // IEEE Transactions on Neural Networks and Learning Systems. — 2018. — Vol. 29, No. 10. — P. 4821–4835. — DOI: 10.1109/TNNLS.2018.2800844.

[82] Zhang Y. Revisiting the Design of Zeroing Neural Network Models // IEEE Transactions on Cybernetics. — 2020. — Vol. 50, No. 6. — P. 2727–2736. — DOI: 10.1109/TCYB.2019.2892301.

[83] Zhang Y., Li K. Dynamic Matrix Inversion Based on Zeroing Neural Networks // IEEE Transactions on Neural Networks. — 2009. — Vol. 20, No. 5. — P. 812–822. — DOI: 10.1109/TNN.2009.2015760.

[84] Zhang Y., Xiao J. Time-varying nonlinear optimization using zeroing neural dynamics // Neural Networks. — 2011. — Vol. 24, No. 8. — P. 863–870. — DOI: 10.1016/j.neunet.2011.04.004.

[85] Li S., Zhang Y. Zeroing Neural Networks for Time-Varying Problems and Real-Time Computing // Neurocomputing. — 2017. — Vol. 267. — P. 660–669. — DOI: 10.1016/j.neucom.2017.06.022.

[86] Zhang Y. [et al.]. Design and analysis of recurrent neural networks for solving time-varying quadratic programming problems // IEEE Transactions on Neural Networks and Learning Systems. — 2015. — Vol. 26, No. 7. — P. 1527–1540. — DOI: 10.1109/TNNLS.2014.2346622.

[87] Zhang Y., You Z. A recurrent neural network for solving nonlinear convex optimization problems in real time // IEEE Transactions on Neural Networks. — 2006. — Vol. 17, No. 6. — P. 1441–1453. — DOI: 10.1109/TNN.2006.880582.

[88] Haykin S. Neural Networks and Learning Machines. — 3rd ed. — Upper Saddle River : Pearson, 2009. — 936 p.

[89] Zurada J. M. Introduction to Artificial Neural Systems. — St. Paul : West Publishing Company, 1992. — 683 p.

[90] Kohonen T. Self-Organizing Maps. — 3rd ed. — Berlin : Springer, 2001. — 501 p. — DOI: 10.1007/978-3-642-56927-3.

[91] Schmidhuber J. Deep learning in neural networks: An overview // Neural Networks. — 2015. — Vol. 61. — P. 85–117. — DOI: 10.1016/j.neunet.2014.09.003.

[92] Bengio Y. Learning deep architectures for AI // Foundations and Trends in Machine Learning. — 2009. — Vol. 2, No. 1. — P. 1–127. — DOI: 10.1561/22000000006.

[93] Krizhevsky A., Sutskever I., Hinton G. E. ImageNet classification with deep convolutional neural networks // Advances in Neural Information Processing Systems 25 (NeurIPS 2012). — 2012. — P. 1097–1105. — DOI: 10.1145/3065386.

[94] Goodfellow I. [et al.]. Generative adversarial nets // Advances in Neural Information Processing Systems 27 (NeurIPS 2014). — 2014. — P. 2672–2680.

VII. Большие языковые модели, LLM-агенты и когнитивные системы управления

[95] Brown T. B. [et al.]. Language Models are Few-Shot Learners // Advances in Neural Information Processing Systems 33 (NeurIPS 2020). — 2020. — Vol. 33. — P. 1877–1901.

[96] OpenAI. GPT-4 Technical Report // arXiv preprint arXiv:2303.08774. — 2023. — DOI: 10.48550/arXiv.2303.08774.

[97] Wei J. [et al.]. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models // Advances in Neural Information Processing Systems 35 (NeurIPS 2022). — 2022. — P. 24824–24837.

[98] Yao S. [et al.]. ReAct: Synergizing Reasoning and Acting in Language Models // Proceedings of the 11th International Conference on Learning Representations (ICLR). — 2023.

[99] Yao S. [et al.]. Tree of Thoughts: Deliberate Problem Solving with Large Language Models // Advances in Neural Information Processing Systems 36 (NeurIPS 2023). — 2023.

[100] Park J. S. [et al.]. Generative Agents: Interactive Simulacra of Human Behavior // Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology (UIST '23). — New York : ACM, 2023. — P. 1–22. — DOI: 10.1145/3586183.3606763.

[101] Wang L. [et al.]. A Survey of Large Language Model Based Autonomous Agents // arXiv preprint arXiv:2308.11432. — 2024. — DOI: 10.48550/arXiv.2308.11432.

[102] Xi Z. [et al.]. The Rise and Potential of Large Language Model Based Agents: A Survey // ACM Computing Surveys. — 2025. — Vol. 57, No. 1. — P. 1–46. — DOI: 10.1145/3701118.

[103] Fu Y. [et al.]. Large Language Models for Intelligent and Autonomous Systems // IEEE Transactions on Artificial Intelligence. — 2024. — Vol. 5, No. 2. — P. 311–329. — DOI: 10.1109/TAI.2024.3353245.

[104] Chase H. LangChain: Building applications with LLMs through composability [Электронный ресурс]. — 2023. — URL: <https://github.com/langchain-ai/langchain> (дата обращения: 15.05.2026).

[105] Microsoft Research. AutoGen: Enabling Next-Gen LLM Applications via Multi-Agent Conversation Framework // arXiv preprint arXiv:2308.08155. — 2023. — DOI: 10.48550/arXiv.2308.08155.

[106] Ye X. [et al.]. Large Language Models as Autonomous Agents: A Survey // IEEE Access. — 2024. — Vol. 12. — P. 10431–10457. — DOI: 10.1109/ACCESS.2024.3350803.

[107] Schulman J. [et al.]. Aligning Language Models to Follow Instructions // arXiv preprint arXiv:2203.02155. — 2022. — DOI: 10.48550/arXiv.2203.02155.

[108] Anthropic. Constitutional AI: Harmlessness from AI Feedback // arXiv preprint arXiv:2212.08073. — 2022. — DOI: 10.48550/arXiv.2212.08073.

[109] Jimenez C. E. [et al.]. SWE-bench: Can Language Models Resolve Real-World GitHub Issues? // Proceedings of the 12th International Conference on Learning Representations (ICLR). — 2024.

[110] Princeton University. SWE-agent: Agent-Computer Interfaces Enable Automated Software Engineering // arXiv preprint arXiv:2405.15793. — 2024. — DOI: 10.48550/arXiv.2405.15793.

[111] Kaplan J. [et al.]. Scaling Laws for Neural Language Models // arXiv preprint arXiv:2001.08361. — 2020. — DOI: 10.48550/arXiv.2001.08361.

[112] Devlin J. [et al.]. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding // Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT). — 2019. — P. 4171–4186. — DOI: 10.18653/v1/N19-1423.

[113] Radford A. [et al.]. Improving Language Understanding by Generative Pre-Training [Электронный ресурс]. — OpenAI Technical Report, 2018. — URL: https://cdn.openai.com/research-covers/language-unsupervised/language_understanding_paper.pdf (дата обращения: 15.05.2026).

[114] Shazeer N. [et al.]. Outrageously Large Neural Networks: The Sparsely-Gated Mixture-of-Experts Layer // Proceedings of the 5th International Conference on Learning Representations (ICLR). — 2017.

[115] Liang P. [et al.]. Holistic Evaluation of Language Models // arXiv preprint arXiv:2211.09110. — 2022. — DOI: 10.48550/arXiv.2211.09110.

[116] Karpathy A. Software 2.0 [Электронный ресурс]. — 2017. — URL: <https://karpathy.medium.com/software-2-0-a64152b37c35> (дата обращения: 15.05.2026).

[117] Amodei D. [et al.]. Concrete Problems in AI Safety // arXiv preprint arXiv:1606.06565. — 2016. — DOI: 10.48550/arXiv.1606.06565.

[118] Russell S. Human Compatible: Artificial Intelligence and the Problem of Control. — New York : Viking, 2019. — 352 p.

[119] Domingos P. The Master Algorithm: How the Quest for the Ultimate Learning Machine Will Remake Our World. — New York : Basic Books, 2015. — 354 p.

[120] Жуков Ф. Г., Антамошкин О. А., Колотов А. В. Алгоритм управления качеством изменений в организационно-производственном процессе с участием ИИ-агента // Автоматизация в промышленности. — 2026. — № 7. — С. 39–45.

VIII. Современные интеллектуальные распределённые и cloud-native системы

[121] Fowler M. Microservices: a definition of this new architectural term [Электронный ресурс]. — 2014. — URL: <https://martinfowler.com/articles/microservices.html> (дата обращения: 15.05.2026).

[122] Newman S. Building Microservices. — 2nd ed. — Sebastopol: O'Reilly Media, 2021. — 617 p.

[123] Beyer B. [et al.]. The Site Reliability Workbook: Practical Ways to Implement SRE. — Sebastopol: O'Reilly Media, 2018. — 560 p.

[124] Majors C., Fong-Jones L., Miranda G. Observability Engineering. — Sebastopol: O'Reilly Media, 2022. — 386 p.

[125] Shi W. [et al.]. Edge Computing: Vision and Challenges // IEEE Internet of Things Journal. — 2016. — Vol. 3, No. 5. — P. 637–646. — DOI: 10.1109/IIOT.2016.2579198.

[126] Satyanarayanan M. The Emergence of Edge Computing // Computer. — 2017. — Vol. 50, No. 1. — P. 30–39. — DOI: 10.1109/MC.2017.9.

[127] Buyya R. [et al.]. Fog and Edge Computing: Principles and Paradigms. — Hoboken: Wiley, 2019. — 672 p.

[128] Mell P., Grance T. The NIST Definition of Cloud Computing. — Gaithersburg: National Institute of Standards and Technology, 2011. — 7 p. — (NIST Special Publication 800-145). — DOI: 10.6028/NIST.SP.800-145.

[129] Erl T. Cloud Computing Design Patterns. — Upper Saddle River: Prentice Hall, 2015. — 536 p.

[130] Dustdar S. The Internet of Things, Services and People // Computer. — 2015. — Vol. 48, No. 9. — P. 30–35. — DOI: 10.1109/MC.2015.297.

[131] Zhao M. [et al.]. A Survey and Future Directions on Cloud and Edge Computing for Distributed AI Systems // IEEE Transactions on Services Computing. — 2024. — Vol. 17, No. 1. — P. 44–63. — DOI: 10.1109/TSC.2023.3338127.

[132] Pasquale J. Resource Management in Dynamic Distributed Systems // IEEE Transactions on Parallel and Distributed Systems. — 2019. — Vol. 30, No. 11. — P. 2524–2539.

[133] Gandhi A. Resource Management for Cloud Computing Platforms // Synthesis Lectures on Distributed Computing Theory. — 2020. — Vol. 7, No. 2. — P. 1–117. — DOI: 10.2200/S01042ED1V01Y202005DCT013.

[134] Youssef A. E. AI-Driven Resource Orchestration for Cloud-Native Systems // IEEE Access. — 2023. — Vol. 11. — P. 55321–55342.

[135] Hu F. Intelligent Edge Computing Systems. — Boca Raton: CRC Press, 2022. — 412 p.

[136] Litoiu M. Self-adaptive cloud systems // Proceedings of the 2015 ACM Symposium on Cloud Computing (SoCC '15). — New York: ACM, 2015. — P. 1–13.

[137] Garlan D., Shaw M. Software Architecture: Perspectives on an Emerging Discipline. — Upper Saddle River: Prentice Hall, 1996. — 256 p.

[138] Bass L., Clements P., Kazman R. Software Architecture in Practice. — 4th ed. — Boston: Addison-Wesley, 2021. — 736 p.

[139] Richardson C. Microservices Patterns. — Shelter Island: Manning Publications, 2018. — 520 p.

[140] Zielinski K. Autonomic Computing Systems for Cloud Infrastructure Management // Future Generation Computer Systems. — 2022. — Vol. 128. — P. 311–326. — DOI: 10.1016/j.future.2021.10.017.

[141] Antamoshkin O. A. Adaptive control of heterogeneous distributed computing systems [Электронный ресурс]. — Режим доступа: <https://github.com/oleslav24/Adaptive-control-of-heterogeneous-distributed-computing-systems> (дата обращения: 18.05.2026).