Федеральное государственное бюджетное учреждение науки
Институт динамики систем и теории управления имени В.М. Матросова
Сибирского отделения Российской академии наук

На правах рукописи

Костромин Роман Олегович

Модели, алгоритмы и инструментальные средства поддержки мультиагентного управления потоками вычислительных заданий

05.13.11 – Математическое и программное обеспечение
вычислительных машин, комплексов и компьютерных сетей

Диссертация на соискание ученой степени
кандидата технических наук

Научный руководитель:
к.т.н., доцент
А.Г. Феоктистов

Иркутск – 2020

Оглавление

Введение

Актуальность темы. В настоящее время как в России, так и за рубежом, ведутся активные исследования, связанные с созданием и применением высокопроизводительных вычислительных систем различного назначения на основе парадигм Grid и облачных вычислений. Результаты этих исследований представлены в работах С.М. Абрамова, А.И. Аветисяна, А.П. Афанасьева, И.В. Бычкова, Вл.В. Воеводина, Б.М. Глинского, В.П. Иванникова, И.А. Каляева, И.И. Левина, А.И. Легалова, Г.И. Радченко, Л.Б. Соколинского, В.В. Топоркова, Г.А. Опарина, А.Н. Черных, Д. Андерсена, Р. Байя, К. Кессельмана, Э. Танненбаума, Я. Фостера, а также многих других российских и зарубежных ученых. В связи с высокой интенсивностью потоков вычислительных заданий в таких системах необходимо эффективное и гибкое управление ими на метауровне.

Эффективность управления состоит в обеспечении высокого качества обслуживания очередей заданий, повышении надежности выполнения заданий, минимизации времени решения задач, поддержки равномерной балансировки загрузки ресурсов и достижении других заданных показателей. Гибкость управления заключается в рациональном предоставлении ресурсов, необходимых для выполнения заданий, в динамически изменяющихся системах.

В процессе управления потоками заданий требуется детальный учет и согласование критериев пользователей, определяемых спецификой решаемых задач, и предпочтений владельцев ресурсов, вытекающих из их характеристик. Как правило, распределенная вычислительная система (РВС), интегрирующая модели Grid и облачных вычислений, обладает рядом свойств, существенно усложняющих унификацию процесса управления заданиями. К свойствам такого рода, например, относятся архитектурно-функциональная разнородность, неполнота описания и динамичность ресурсов, широта спектра задач, решаемых с помощью этих ресурсов, наличие различных категорий пользователей, преследующих свои цели и задачи эксплуатации вычислительной среды.

Известные модели, методы, алгоритмы и программные средства

централизованного управления потоками заданий не решают перечисленные выше проблемы полностью. Как правило, это обусловлено следующими причинами [1]:

- ограниченным контролем централизованной системы над ее распределенными ресурсами, не позволяющим в полной мере оценить состояние этих ресурсов и воздействовать на него;
- отсутствием необходимой информации о разнородных ресурсах (степени их надежности, показателях производительности, вычислительной истории выполнения конкретных заданий, дисциплинах обслуживания очередей, административных политиках и других важных характеристиках) или существенным увеличением накладных расходов при получении таких сведений, которое приводит к снижению эффективности функционирования системы управления;
- недостаточной надежностью самой системы управления, так как при отказе центрального узла вся вычислительная среда становится неработоспособной;
- резким снижением производительности системы при значительном увеличении числа заданий потока.

В связи с этим возникает необходимость разработки новой эффективной и гибкой системы управления потоками заданий. Для решения данной проблемы целесообразно использование принципов организации распределенного группового управления. Качество управления, осуществляемого отдельными компонентами, обеспечивается наличием у них более полных локальных знаний об управляемых ими ресурсах по сравнению с централизованной системой.

Групповое управление потоками заданий в процессе решения сложной прикладной задачи в РВС осуществляется посредством передачи сигналов (сообщений) в коммуникационной среде между компонентами распределенной управляющей системы. Учитывая динамическую природу РВС, целесообразно применять адаптивное предоставление ресурсов в процессе управления потоками заданий. В этом случае перспективным подходом является использование мультиагентных технологий. В рамках такого подхода отдельные ресурсы

представляются специализированными приложениями (агентами), образующими в совокупности мультиагентную систему (МАС) управления.

Анализ результатов исследований в области самоорганизации вычислительных систем (см., например, работы Д.В. Винса, Т.А. Гавриловой, В.И. Гальперова, В.И. Городецкого, И.А. Каляева, А.С. Ковтуненко, Л.В. Массель, Д.А. Поспелова, В.Б. Тарасова, В.Ф. Хорошевского, N. Jennings, S.J. Russell, P. Norvig, G. Di Marzo Serugendo, M. Wooldridge, F. Zambonelli) показывает, что эффективное управление потоками заданий с помощью МАС достигается за счет применения алгоритмов работы агентов, адаптирующихся в процессе их выполнения к текущим условиям и состоянию функционирования среды в соответствии с заданной агентам целью, определенными критериями качества решения задачи и использования ресурсов, а также знаниями об особенностях предметных областей решаемых задач.

Построение проблемно-ориентированной самоорганизующейся МАС порождает ряд проблем, связанных с автоматизацией разработки агентов и агентных платформ, реализации алгоритмов функционирования агентов, а также накопления и применения предметных знаний агентами. Существующие в настоящее время инструменты для создания МАС не позволяют решить вышеперечисленные проблемы в полной мере [2].

Цель работы заключается в разработке новых инструментальных средств, обеспечивающих снижение трудозатрат при построении МАС по сравнению с существующими инструментариями, а также моделей и алгоритмов работы агентов создаваемых систем, позволяющих улучшить показатели (качество обслуживания очереди заданий, время и надежность их выполнения, балансировку загрузки ресурсов) управления потоками вычислительных заданий в разнородной РВС по сравнению с известными метапланировщиками GridWay [3] и Condor Directed Acyclic Graph Manager (DAGMan) [4].

Основные задачи для достижения поставленной цели:

– проведение сравнительного анализа известных методов и средств

организации МАС, а также известных методов и средств мультиагентного управления потоками вычислительных заданий;

- разработка модели поведения агентов;
- разработка алгоритмов функционирования агентов;
- разработка инструментального комплекса построения МАС для управления потоками вычислительных заданий;
- создание МАС управления потоками заданий в экспериментальной РВС;
- оценка надежности и эффективности управления потоками заданий с помощью разработанной МАС, а также трудоемкости ее построения.

Объектом исследования является процесс управления потоками вычислительных заданий в РВС.

Предметом исследования выступают мультиагентные модели, алгоритмы и система управления потоками заданий в разнородной РВС, а также инструментальные средства построения МАС.

Методы исследования. При решении поставленных задач использовались методы концептуального, имитационного, конкретизирующего и автоматного программирования, организации распределенных вычислений и мультиагентных технологий, а также машинного обучения.

Научная новизна диссертации состоит в интеграции уникальной совокупности методов и средств концептуального, имитационного, конкретизирующего и автоматного программирования, классификации заданий и параметрической настройки алгоритмов работы агентов в качестве основы их машинного обучения, организации распределенных вычислений и управления ими в процессе создания и применения оригинальных мультиагентных моделей, алгоритмов и системы управления потоками заданий в разнородной РВС, а также инструментальных средств их разработки.

На защиту выносятся следующие основные результаты:

1) ролевая модель поведения агентов, базирующаяся, в отличие от известных, на использовании конечных управляющих автоматов с динамическим

планированием их состояний-действий в разнородной РВС на концептуальной модели среды и применении механизма порождения дочерних автоматов для реализации специфических ролей в возникающих виртуальных сообществах;

2) система машинного обучения агентов, которая основывается на применении новой гибридной модели представления знаний, обеспечивающей интегрированное использование концептуального и имитационного моделирования, классификации заданий и параметрической настройки алгоритмов работы агентов в качестве методов обучения в сочетании с процессами самостоятельного извлечения и передачи знаний агентами;
3) мультиагентный алгоритм перераспределения ресурсов разнородной РВС в случае отказа ее программно-аппаратных средств, реализующий адаптивное мультисценарное решение данной проблемы на основе конкретизирующего программирования и тем самым существенно повышающий отказоустойчивость процесса выполнения заданий под управлением МАС по сравнению с метапланировщиками GridWay и Condor DAGMan;
4) инструментальный комплекс построения МАС, обеспечивающий по сравнению с известными инструментариями сокращение трудозатрат на реализацию разработанных моделей, алгоритмов и системы в целом путем автоматизации основных этапов разработки, настройки, конфигурации и применения агентов.

Достоверность и обоснованность полученных в работе результатов подтверждается корректным применением классических методов исследования, а также анализом адекватности разработанных моделей и алгоритмов на основе полунатурного моделирования.

Соответствие диссертации паспорту специальности. Тема и основные результаты диссертации соответствуют следующим областям исследований, включенным в паспорт специальности 05.13.11 – Математическое и программное обеспечение вычислительных машин, комплексов и компьютерных сетей:

- модели, методы, алгоритмы, языки и программные инструменты для организации взаимодействия программ и программных систем;
- модели и методы создания программ и программных систем для параллельной и распределенной обработки данных, языки и инструментальные средства параллельного программирования.

Практическая значимость. Применение вышеперечисленных результатов диссертационного исследования обеспечивает существенное улучшение качества обслуживания очередей заданий, минимизацию времени решения задач, повышение надежности их выполнения и сбалансированную загрузку ресурсов по сравнению с известными метапланировщиками GridWay и CondorDAGMan.

Исследование, разработка и применение рассматриваемых в диссертации программных средств выполнялись в рамках следующих научно-технических работ:

- проектов РФФИ № 14-08 3162-мол_а «Методические подходы и комплекс программ для оптимизации режимов работы крупных ТЭЦ», № 16-07-00931-а «Методология и инструментальные средства разработки и применения проблемно-ориентированных мультиагентных систем управления масштабируемыми вычислениями в разнородной распределенной вычислительной среде» и № 19-07-00097-а «Фундаментальные проблемы непрерывной интеграции функционального наполнения распределенных пакетов прикладных программ на основе инженерии знаний»;
- проекта «Фундаментальные проблемы решения сложных практических задач с помощью суперкомпьютеров» программы фундаментальных исследований президиума РАН № 27;
- проекта «Методы, алгоритмы и инструментальные средства децентрализованного группового решения задач в вычислительных и управляющих системах» программы фундаментальных исследований президиума РАН № 30;
- базовых тем исследований ИДСТУ СО РАН № IV.38.1.2 «Разработка

проблемно-ориентированных технологий, систем и сервисов поддержки научных исследований на основе интеллектных методов и алгоритмов организации параллельных и распределенных вычислений» и № IV.38.1.1 «Технологии разработки проблемно-ориентированных самоорганизующихся мультиагентных систем группового управления: методы, инструментальные средства, приложения».

Практическое использование разработанных моделей, алгоритмов, методов и инструментальных средств в процессе решения научных и прикладных задач подтверждено справкой об использовании разработанного программного обеспечения (ПО), представленной в Приложении А.

Апробация результатов работы. Основные результаты работы докладывались автором на следующих научных мероприятиях: 13th International Symposium «Intelligent Systems» (INTELS-2018, Санкт-Петербург, Россия, 2018 г.), International Symposium on Cloud Computing and Services for High Performance Computing Systems (HPCS-2018, Орлеан, Франция, 2018 г.), 41th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO-2018, Опатия, Хорватия, 2018 г.), 4th International Conference on Information Technology and Nanotechnology (ITNT-2018, Самара, Россия, 2018 г.), XLVI Международная конференция «Информационные технологии в науке, образовании и управлении» (IT+S&E`17, Гурзуф, Россия, 2017 г.), IV Всероссийская мультиконференция по проблемам управления (МКПУ-2017, Дивноморское, Россия, 2017 г.), XXII и XXIII Байкальские Всероссийские конференции с международным участием «Информационные и математические технологии в науке и управлении» (ИМТ, Иркутск – Байкал, Россия, 2017–2018 гг.), Multidisciplinary Youth Academic Research Conference on Science Present and Future: Research Landscape in the 21st century (Иркутск, Россия, 2017 г.), XIII Всероссийская конференция молодых ученых «Моделирование, оптимизация и информационные технологии» (Иркутск, Россия, 2017 г.), конференции «Ляпуновские чтения» (Иркутск, Россия, 2016–2019 гг.), IV и V Всероссийские научно-технические конференции «Суперкомпьютерные технологии» (СКТ,

Дивноморское, Россия, 2016, 2018 гг.), XVI и XVIII Всероссийские конференции молодых ученых по математическому моделированию и информационным технологиям (Красноярск, Россия, 2015 г.; Иркутск, Россия, 2017 г.), XV Молодежная научно-практическая конференция «Российская цивилизация: история, проблемы, перспективы» (Иркутск, Россия, 2015 г.), а также семинарах ИДСТУ СО РАН.

Публикации. Результаты научных исследований автора отражены в 34 научных работах [5–38]. В их числе 6 публикаций [6, 7, 9, 15, 28, 29] в российских журналах, рекомендованных ВАК для опубликования научных результатов диссертации, 8 публикаций [14, 16, 25–27, 31, 32, 34] проиндексированных в международных базах цитирования Web of Science и Scopus. Получены 2 свидетельства о государственной регистрации программ для ЭВМ [39, 40] (Приложение Б).

Личный вклад автора. Все выносимые на защиту научные положения получены соискателем лично. В основных научных работа по теме диссертации, опубликованных в соавторстве, лично соискателем разработаны: в [7, 10, 11] – сравнительный анализ методов и средств организации МАС; в [14, 16, 20, 25, 26, 28, 34, 39] – ролевая модель поведения агентов и алгоритмы их функционирования; в [9, 15, 27, 29, 31, 32, 38, 40] – инструментальные средства создания агентов и результаты вычислительных экспериментов в рамках полунатурного моделирования мультиагентной системы.

Структура работы. Диссертация состоит из введения, четырех глав, заключения, списка используемых сокращений, глоссария, библиографии из 125 наименований и семи приложений. Общий объем работы – 173 страницы, из которых 117 страниц основного текста, включающего 55 рисунков и 13 таблиц.

Глава 1. Агентно-ориентированный подход к управлению потоками заданий

Широко используемым на практике подходом к интеллектуализации промежуточного ПО РВС является применение МАС.

В первой главе диссертации рассматриваются общие вопросы, связанные с организацией и применением МАС для управления потоками вычислительных заданий в РВС, исследуются известные МАС, проводится сравнительный анализ методов и средств построения таких систем, обосновывается необходимость разработки нового инструментального комплекса создания МАС для управления потоками заданий, формулируются системные и функциональные требования к разрабатываемому комплексу. В заключение подытоживаются основные результаты исследований, представленных в данной главе.

1.1. Мультиагентная система

Под агентом понимается аппаратная или программная сущность, способная действовать в интересах достижения целей, поставленных перед ним владельцем и/или пользователем [41–43]. Агент способен принимать самостоятельные решения, обучаться, находить других агентов для взаимодействия и объединения с ними в сообщества. Эти свойства агентов позволяют реализовать сложный механизм, благодаря которому агенты могут автономно существовать в своей агентной среде и выполнять свои обязанности в соответствии с назначенными ролями.

Агент может обладать следующими основными свойствами [44–46]:

- автономностью – способностью функционировать без вмешательства человека, осуществляя самоконтроль над своими действиями и внутренним состоянием;
- общественным поведением – способностью функционировать в сообществе агентов, обмениваясь сообщениями с помощью некоторого языка коммуникаций;

- реактивностью – способностью воспринимать состояние среды и своевременно реагировать на ее изменения;
- наличием знаний – информацией о себе, о среде и о других агентах, а также о правилах применения решений на основе этой информации;
- убеждениями – знаниями агента о недостоверной внешней среде и о других агентах, которые могут изменяться во времени и становиться неверными;
- желаниями – состояниями, достижение которых является для агента желательным. Они могут быть противоречивыми, но агент может выбирать только их непротиворечивое подмножество;
- намерениями – тем, что агент обязан сделать в соответствии со своим выбором или в силу своих обязательств по отношению к другим агентам;
- целями – конкретным множеством конечных и промежуточных состояний, достижение которых агент считает реализацией своих намерений;
- обязательствами – теми задачами, которые агент берет на себя по просьбе или по поручению других агентов в рамках кооперативных целей;
- проактивностью – способностью действовать в упреждающей манере, в частности, генерировать новые цели и действовать рационально для их достижения, а не только реагировать на события;
- мобильностью – способностью агента мигрировать по сети Интернет в поисках необходимой информации или сервисов;
- благожелательностью – готовностью агентов помогать друг другу, что предполагает отсутствие у агентов конфликтующих целей;
- правдивостью – свойством агента не манипулировать информацией, про которую ему заведомо известно, что она ложна;
- рациональностью – свойством агента действовать так, чтобы достигнуть своих целей, а не избегать их достижения в рамках своих знаний и убеждений.

Классификации агентов в зависимости от их свойств [47, 48] приведена в таблице 1. Следует отметить, что в настоящее время агенты, используемые на

практике, как правило, не превышают уровня смышленых агентов [47, 49].

Таблица 1 – Характеристики основных типов агентов

Характеристики	Простые (simple)	Смышленые (smart)	Интеллектуальные (intelligent)	Действительно интеллектуальные (truly intelligent)
Автономное выполнение	+	+	+	+
Взаимодействие с другими агентами и/или пользователями	+	+	+	+
Слежение за окружением (реактивность)	+	+	+	+
Способность использования абстракций		+	+	+
Способность использования предметных знаний		+	+	+
Способность адаптивного поведения для достижения целей			+	+
Обучение из окружения			+	+
Толерантность к ошибкам и/или неверным входным сигналам				+
Исполнение в реальном времени				+
Взаимодействие с пользователями на естественных языках				+

МАС представляет собой сеть слабо связанных решателей частных проблем (агентов), которые существуют в общей среде и взаимодействуют между собой для достижения тех или иных целей системы [50].

Как правило, МАС обладает следующими свойствами [51–54]:

- каждый агент обладает неполной информацией о среде, имеет собственную модель внешней среды и ограниченные возможности по решению «своей» проблемы;
- глобальное управление агентами ограничено или отсутствует;
- данные, которые используются агентами, децентрализованы и часть их может являться «собственностью» отдельных агентов;
- агенты функционируют в асинхронном режиме;
- агенты координируют свои действия путем обмена сообщениями на языке высокого уровня.

Рассмотрим ключевые отличительные свойства агентов МАС по сравнению с традиционными объектно-ориентированными системами [52].

Объект является пассивной программной сущностью. Его методы вызываются извне и объект выполняет запрос безоговорочно (детерминированная реакция). Если объект говорит «нет» (не реагирует на запрос), то это исключительная ситуация, требующая специальной обработки. В то же время обращение к агенту – это «просьба», которую он может и не исполнять на основе принятых в МАС соглашений о взаимных обязательствах в зависимости от своего внутреннего состояния и т.п., т.е. это решается агентом–потенциальным исполнителем (недетерминированная реакция). Поэтому агент является активной сущностью. Кроме того, объекты не могут инициировать взаимодействие и мигрировать по сети, в отличие от агентов.

Для взаимодействия с объектом имеется только один метод на каждый тип входного сообщения, формат сообщения должен строго соблюдаться и не поддерживается асинхронный режим. Содержание входного сообщения агента имеет более свободную форму в виде строки на языке высокого уровня. Коммуникации агента обычно асинхронные и поддерживают параллельную обработку. Агент может поддерживать сразу несколько диалогов, решая самостоятельно вопрос об очередности участия в них (в зависимости от своего

внутреннего состояния). Он сам решает, как чередовать диалоги с решением задач (когда переходить от «внешнего» поведения к «внутреннему»).

Важным свойством является способность к обучению. Объект создается классом и его способности далее не изменяются, при этом агент может быть снабжен способностью к обучению.

С точки зрения характера поведения можно рассмотреть следующие отличительные признаки: объект не может реагировать сам на события внешнего мира. Следует заметить, что последние версии UML [55] и Java [56] обеспечивают возможность «слушать» внешнюю среду. Поведение объекта строго предопределено. Агент же может не только неоднозначно реагировать на специфические запросы, представленные во входящем сообщении, но также и самостоятельно реагировать на события во внешнем мире используя механизм «подписки», и/или информацию от сенсоров, и принимать решение о выборе того или иного поведения.

Агент может запустить процесс по собственной инициативе (используя свои проактивные механизмы), в отличие от объекта. Объект не может объявлять свои интерфейсы. Объектно-ориентированные языки позволяют только спросить объект о его интерфейсах. Агент может объявлять свои сервисы и их интерфейсы, используя механизм сервисов белых и желтых страниц. В этой связи МАС хорошо подходят для реализации посреднических услуг в процессе обработки потоков заданий, когда агенты наделяются правами и полномочиями владельцев ресурсов и их пользователей, которые решают свои задачи с помощью данных ресурсов. Способность МАС к самоорганизации делает процесс управления более эффективным и гибким [52].

Самоорганизация МАС есть динамический адаптивный процесс, приводящий к возникновению и поддержке структур агентов и их локальных взаимодействий без внешнего вмешательства [57]. Основными отличительными чертами самоорганизующихся процессов (агентов) мультиагентного управления являются следующие признаки [58, 59]:

- автономность – быть в состоянии взаимодействовать с миром, управлять собственным поведением, направленным на достижение своих локальных целей без вмешательства извне;
- быть способными воспринимать внешнюю среду и локально воздействовать на нее;
- иметь программную среду для распределенного взаимодействия с другими агентами;
- глобальный порядок возникает в системе только благодаря локальным взаимодействиям ее компонент;
- наличие эмерджентных свойств, отсутствующих на уровне отдельных агентов и проявляющихся только на уровне системы в целом в процессе взаимодействия агентов;
- нелинейная динамика взаимодействия агентов, неустойчивость и чувствительность к вариациям начального состояния и малым вариациям параметров системы (это свойство не может быть выведено из свойств отдельных компонент и их локальных взаимодействий);
- множественность устойчивых состояний системы;
- избыточность обеспечивает нечувствительность к отказам или разрушениям отдельных элементов;
- адаптивность – это способность самоорганизующейся системы изменять свое поведение и переходить в новое устойчивое состояние со сменой организационной структуры;
- сложность системы обусловлена большим числом взаимодействующих компонент;
- простота правил, которые используются компонентами системы в процессах взаимодействия, способствующих сложному поведению системы в целом;
- иерархическая структура системы описывается, по крайней мере, на двух уровнях, а именно, на уровне локальных компонент системы и на мета-уровне, где проявляются эмерджентные свойства системы.

Технология МАС удовлетворяет базовым требованиям для программной реализации самоорганизующихся систем [42], при этом является единственной технологией, которая имеет все средства для реализации таких систем. По этой причине теория и практика самоорганизации сейчас является областью исследований и разработок, главным образом, в области МАС [52].

1.2. Мультиагентные системы управления потоками вычислительных заданий в распределенной вычислительной среде

Известен широкий спектр МАС, применяемых на практике для обработки потоков вычислительных заданий в РВС. В их числе такие системы, как Condor-G [60], GridSolve [61], Application Level Scheduling (AppLes) [62], Mobile Agent-based Grid Environment (MAGE) [63], Multi-Agent Architecture for Grid Environment (MAAG) [64], Singh Framework [65].

Проект Condor-G разрабатывается группой ученых Висконсинского университета в Мадисоне и является набором открытого ПО для организации крупномасштабных вычислений с интенсивным потоком заданий. Применяется для управления загрузкой кластеров выделенных узлов, а также для запуска вычислений на персональных компьютерах в момент их простоя. Клиент Condor-G работает под управлением операционных систем (ОС) на базе Linux, Unix, Mac OS X, FreeBSD и Microsoft Windows. Condor-G способен объединять в единую вычислительную среду как выделенные узлы (традиционные узлы кластеров, монтируемые в стойки), так и невыделенные узлы.

Система GridSolve предназначена для объединения распределенных вычислительных ресурсов посредством локальной сети, в том числе и персональных компьютеров. GridSolve играет роль клиента, сервера, агента и способен посредством удаленного вызова процедур обращаться к аппаратным и программным компонентам, является промежуточным ПО (middleware). С 2008 года не поддерживается.

Проект AppLeS предоставляет методологию, прикладное ПО и программную среду для адаптивного диспетчирования и распространения приложений в

многопользовательской разнородной Grid-среде.

MAGE – это система мониторинга Grid-среды для предоставления динамической реконфигурируемой многоуровневой архитектуры. MAGE является легковесным средством организации мобильных агентов для получения независимых типов обмена сообщениями и других актуальных задач. Каждый элемент приложения может быть заменен новым заданием без вмешательства в другие задания, так как агенты представлены сервисами в MAGE.

Архитектура MAAG создавалась с учетом свойств разнородности вычислительных ресурсов и приложений в Grid-средах. MAAG включает пять типов агентов, которые осуществляют совместное управление распределенными ресурсами в Grid-среде и поддерживают гибридные потоки заданий для обеспечения установленных критериев качества обслуживания (Quality of Service, QoS) приложений. В MAAG каждый сервис контролируется определенным агентом. Активная разработка системы прекращена с 2009 г.

Основным назначением платформы Singh Framework является построение МАС для семантических веб-сервисов. Она объединяет агентов, работающих на разных уровнях МАС, и расширяет возможности по их взаимодействию. Уровни МАС характеризуются различным набором знаний агентов, их возможностями и целями функционирования. Как правило, вышеперечисленным системам характерно наличие встроенных агентов, мониторинг среды и распределение ресурсов. Только в систему MAGE встроена возможность обучения агентов.

Поддержка управления в рамках как локальной системы (например, вычислительного кластера), так и глобальной системы (например, Grid-системы), реализована лишь в системе MAAG. Централизованный и децентрализованный алгоритмы взаимодействия агентов, управление на уровне приложений с целью оптимизации обработки заданий для конкретного приложения представлены в отдельных системах (Condor-G, GridSolve).

К сожалению, ряд функциональных возможностей (обеспечение повышенной надежности обмена сообщениями между агентами за счет

использования системы логического времени, возможность исполнения нескольких ролей одним агентом, обучение агентов, применение агентами экономических механизмов управления, формулирование непроцедурной и процедурной постановки задачи и автоматическое построение схемы решения задачи), необходимых для обеспечения эффективного и гибкого управления потоками заданий, зачастую отсутствует в подобных системах (см., например, AppLes, Condor-G).

1.3. Сравнительный анализ методов и средств организации мультиагентных систем

Использование общепринятого набора правил (стандарта) взаимодействия агентов является важным аспектом при разработке МАС [66], обеспечивающим открытость, надежность и гибкость коммуникационных процессов.

Основными стандартами взаимодействия агентов являются Knowledge Query and Manipulation Language (KQML) [67], Agent Communication Language (ACL), разработанный в соответствии со спецификацией Foundation for Intelligent Physical Agents (FIPA) и упоминаемый как FIPA-ACL [68], разработанная французской телекоммуникационной компанией система The Advanced Regional Traffic Interactive Management and Information System (ARTIMIS) и созданный для этой системы язык ARTIMIS COmmunication Language (ARCOL) [66], Domain independent COOrdination Language (COOL) [69].

Самым широко применяемым языком взаимодействия агентов является KQML. В нем используются выражения-действия, такие как ответить, сказать, отменить, сказать обратное и т.д. Каждое сообщение KQML содержит выражение-действие и дополнительные сведения, записанные в различные поля. Множества выражений-действий в KQML и их слотов вполне достаточно, чтобы позволить агентам взаимодействовать между собой. Известны проблемы с семантикой выражений-действий и с различной интерпретацией этих выражений агентами.

FIPA-ACL возник во многом под влиянием ARCOL. FIPA-ACL, ARCOL и KQML совместно установили пред-стандарт языка взаимодействия агентов.

Синтаксис и семантика FIPA-ACL очень похожи на синтаксис и семантику KQML.

Для системы ARTIMIS разработан специальный язык ARCOL. Он обладает меньшим набором коммуникационных примитивов, чем KQML.

Язык COOL основан на взаимодействии на основе речевых актов, нацелен на четкое представление и применение знаний в мультиагентных системах и фокусируется на управлении переговорами на основе правил (правила переговоров, правила ошибок, правила продолжения).

Общей проблемой при разработке МАС является выбор конкретного набора программных средств. На данный момент известен широкий спектр агентных платформ и инструментальных средств разработки агентов и МАС [66]. Большинство из них разрабатывается на языке программирования Java. Однако только отдельные системы востребованы для решения практических задач.

Разработка многих систем прекращена, другие системы имеют узкую специализацию и соответственно ограниченную область применения, а некоторые системы не соответствуют стандартам (например, FIPA, KQML). Немаловажным фактором востребованности инструментальных средств является их свободное распространение, наличие документации к ним, а также поддержка сообществом разработчиков (например, форум разработчиков) [70].

Для каждой из систем необходимо рассмотреть следующие свойства:

- совместимые ОС;
- лицензия и стоимость применения;
- соответствие стандартам работы и обмена сообщениями;
- мобильность разрабатываемых агентов;
- сложность освоения и использования;
- масштабируемость;
- производительность;
- поддерживаемые языки программирования.

Детальная информация по известным на данный момент системам представлена в Приложении В. Рассмотрим ряд наиболее популярных

инструментариев.

Agent Factory [71]. Основное назначение: построение агентных систем общего назначения. Разработчик: University College Dublin, Дублин, Ирландия. Популярность: средняя. Масштабируемость: хорошая.

Agent Factory – это среда открытый набор средств, платформ и языковых средств поддержки разработки и распространения МАС. Данное средство разделено на две сферы распространения агентов – ноутбуки, компьютеры, серверы и отдельно – мобильные устройства.

AgentBuilder [72]. Основное назначение: построение агентных систем общего назначения. Разработчик: Acronymics Inc, Альма, США. Популярность: средняя. Масштабируемость: хорошая. Проект завершен и не развивается.

AgentBuilder является интегрированным инструментальным средством быстрой разработки интеллектуальных агентов и агентно-ориентированных приложений. AgentBuilder позволяет уменьшить время и стоимость разработки агентов за счет наличия большого числа шаблонов агентов и встроенных библиотек.

AgentScape [73]. Основное назначение: масштабируемые распределенные агентные системы. Разрабатывается Delft University of Technology, Делфт, Нидерланды. Популярность: низкая. Масштабируемость: хорошая. Не развивается с 2010 г.

AgentScape представляет набор программных средств разработки моделей агентно-ориентированных систем общего назначения и средств визуализации работы моделей. Разработка моделей с применением AgentScape не требует написания больших объемов программного кода. Проект по большей части исследовательский, стандарты, API подвержены частым изменениям. Разрабатывается на Java под открытой лицензией.

Cognitive Agent Architecture (Cougaar) [74]. Основное назначение: комбинированные, масштабируемые распределенные приложения. Разработчик: Raytheon Bolt, Beranek and Newman Technologies, Вьенна, США. Популярность:

низкая. Масштабируемость: высокая.

Архитектура агентов Cougaar разработана под открытой лицензией и включает базовые сервисы и необходимые элементы инфраструктуры функционирования агентов. Работа агентов управления вычислениями основана на декомпозиции больших задач. Агенты управляют поведением приложений, а среда отвечает за адаптацию системы. Агенты и среда могут разрабатываться независимо, но выполняются всегда вместе. Для разработчиков доступны средства написания собственных плагинов и дополнительных модулей для расширения функциональных возможностей агентов Cougaar.

CybelePro [75]. Основное назначение: масштабируемые распределенные агентные системы. Разработчик: Intelligent Automation Inc, Роквилл, США. Популярность: низкая. Масштабируемость: высокая.

CybelePro предоставляет пользователям надежную высокопроизводительную инфраструктуру для быстрой разработки и разворачивания крупномасштабных, высокопроизводительных МАС, является коммерческим продуктом. Инфраструктура агентов CybelePro широко применяется в правительстве, промышленности и научной сфере для военной логистики, моделировании, управлении наземным и воздушным транспортом, коммуникационными сетями и в разработке открытых систем.

EMERALD [76]. Основное назначение: распределенные приложения, состоящие из автономных сущностей. Разработчик: Logic Programming and Intelligent Systems Group, Aristotle University of Thessaloniki, Салоники, Греция. Популярность: низкая. Масштабируемость: высокая.

EMERALD – это программная платформа для семантических веб-сервисов, ориентированная на знания. Предназначен для унификации процесса общения агентов и обмена знаниями в разнородных информационных системах, представленных в виде веб-сервисов. Одним из приложений EMERALD является изучение процесса обучения агентов во время проведения торгов от имени пользователей.

В EMERALD агенты могут обмениваться между собой наборами правил без необходимости согласования формата представления данных правил с другими агентами. Принимающий агент может использовать внешние сервисы для анализа семантического набора правил. В EMERALD встроена поддержка таких языков, как XML и Prolog.

General Agent Modeling Approach (GAMA) [77]. Основное назначение: масштабное распределенное пространственно-точное агентное моделирование. Разработчик: Institute of Research for Development, Mathematical and Computer Modelling of Dynamical Systems, Ханой, Вьетнам. Популярность: низкая. Масштабируемость: хорошая.

GAMA предназначен для моделирования и разработки имитационной среды для построения агентных моделей с помощью специального высокоуровневого языка GAMA Language (GAML). Является системой общего назначения, может найти применение в широком спектре предметных областей.

Java Agent Development Framework (JADE) [78]. Основное назначение: распределенные приложения, состоящие из автономных сущностей. Разработчик: Telecom Italia, Рим, Италия. Популярность: самая высокая. Масштабируемость: высокая.

JADE – это программная среда разработки МАС и приложений, полностью написанная на языке Java и выполняется в виртуальной машине Java (Java Virtual Machine, JVM). Данная среда упрощает разработку МАС в виде промежуточного ПО, полностью совместимого с FIPA. В JADE встроен набор графических инструментов, необходимых при отладке и разворачивании агентов. Разработанные с помощью JADE системы могут мигрировать между платформами МАС, размещенных на различных узлах. JADE включает в себя набор стандартных агентов для управления конструируемыми агентами МАС и библиотеку стандартных классов для Java-программ. Некоторые инструментарии используют средства JADE. Например, работа агентов EMERALD реализована с помощью этих средств.

Java Agent Development framework for programming intelligent software agents in XML (JADEx) [79]. Основное назначение: распределенные приложения, состоящие из автономных сущностей в рамках модели убеждений, желаний и намерений (belief-desire-intention). Разработчик: Hamburg University, Гамбург, Германия. Популярность: высокая. Масштабируемость: высокая.

Модель убеждений, желаний и намерений является хорошо известной агентной архитектурой, которая облегчает описание поведения, целей, планов и вознаграждений агентов. Суть идеи в том, чтобы обеспечить явное различие между результатом (цель) и тем, как эта цель достигнута (план). Такое разделение помогает по ряду причин: с одной стороны, позволяет разрабатывать поведение агентов в удобном и понятном виде, а с другой стороны – поведение становится более явным после того, как выполнены цели и планы.

Multi Agent Development Kit (MaDKit) [80]. Основное назначение: мультиагентные системы с агентным моделированием. Разработчик: Institut universitaire de technologie, Монпелье, Франция. Популярность: средняя. Масштабируемость: хорошая.

MaDKit – это мультиагентная среда разработки, написанная на Java, которая позволяет легко проектировать распределенные приложения и проводить моделирование с применением мультиагентного подхода. Одной из ключевых особенностей является применение модели Агент/Группа/Роль, которая не привязана ни к одной предопределенной модели, т.е. агент играет свою роль внутри группы, образуя виртуальные сообщества (ВС). Кроме того, MaDKit не вмешивается во внутреннюю структуру агентов, что позволяет разработчику свободно создавать собственную архитектуру агентов.

В таблице 2 приведено сравнение ключевых характеристик рассматриваемых агентных платформ.

Таблица 2 – Ключевые характеристики распространенных агентных платформ

Инструментарий	Совместимые ОС	Бесплатная	Соответствие стандартам	Обмен сообщениями
Agent Factory	Любая с JVM	+	Частично FIPA	HTTP
AgentBuilder	Windows, Linux, Sun Solaris	–	KQML	KQML, TCP/IP
AgentScape	Любая с JVM	+	Неизвестно	Внутренний язык платформы
Cougaar	Windows, Linux	+	Неизвестно	Cougaar Message Transport Service
CybelePro	Любая ОС с JVM	–	Неизвестно	Внутренний язык платформы
EMERALD	Любая с JVM	+	FIPA, Semantic web standards	Асинхронный ACL
GAMA	Windows, Linux, Mac OS	+	FIPA, Geographic Information Systems	ACL
JADE	Любая ОС с JVM	+	FIPA	Асинхронный ACL, HTTP, WAP
JADEx	Любая ОС с JVM	+	FIPA, Web Services Description Language	HTTP
MaDKit	Любая ОС с JVM	+	Unified Modeling Language	P2P

Исходя из сравнительных результатов, приведенных в таблице 2, можно заключить, что наиболее пригодными для организации МАС являются следующие инструментальные средства: EMERALD, GAMA, JADE, JADEx. Они находятся в активной разработке, соответствуют стандартам, являются кроссплатформенными, имеют открытый исходный код, являются применимыми к широкому спектру задач. Несмотря на то, что GAMA удовлетворяет большинству требований разработчиков и имеет удобный интерфейс пользователя, она обладает следующим недостатком: для разработки агентной системы необходимо осваивать язык GAML, что затрудняет применение этой платформы совместно с популярными языками программирования. Эта система больше подходит для агентного моделирования.

Системы Agent Factory, EMERALD, JADE и JADEx в целом близки по

сравниваемым характеристикам, но при этом популярность у EMERALD достаточно низкая, Agent Factory не в полной мере соответствует стандартам FIPA и KQML. Это связано с тем, что одним из желательных критериев является возможность работы агентов на мобильных платформах, таким требованиям удовлетворяют только JADE [78, 81] и JADEx [79, 82]. Обе эти платформы чаще остальных используются для разработки мультиагентных систем [2].

Общим недостатком рассмотренных систем является слабая привязка к предметной области решаемой задачи при разработке МАС.

1.4. Функциональные и системные требования к инструментальным средствам разработки мультиагентной системы

Результаты исследования известных средств и методов мультиагентного управления потоками вычислительных заданий позволяют сделать вывод о том, что ряд функциональных возможностей, необходимых для обеспечения эффективной и гибкой обработки заданий в РВС, зачастую отсутствует в подобных системах. Сравнительный анализ методов и средств организации МАС показал, что они также не обеспечивают реализацию таких функциональных возможностей в разрабатываемых системах.

Таким образом, возникает необходимость в разработке новых инструментальных средств, обеспечивающих автоматизацию основных этапов разработки, настройки и конфигурации виртуальных сообществ агентов управления потоками заданий, и наделение агентов необходимыми функциями. В этой связи разрабатываемый инструментальный комплекс должен отвечать нижеследующим требованиям.

Системные требования:

- работа под управлением различных ОС (кроссплатформенность);
- соответствие стандартам FIPA;
- толерантность к ошибкам и/или неверным входным сигналам;
- использование системы логического времени.

Функциональные требования:

- проблемная ориентированность разрабатываемой МАС;
- использование элементов обучения агентов;
- взаимодействие с другими агентами и/или пользователями;
- слежение за окружением (реактивность);
- способность использования предметных знаний;
- способность адаптивного поведения для достижения целей;
- обучение из окружения;
- возможность исполнения нескольких ролей одним агентом;
- применение агентами экономических механизмов управления.

Разработка данного инструментального комплекса в соответствии с вышеперечисленными требованиями позволит создавать агентов с требуемыми системными и функциональными возможностями.

1.5. Выводы

В первой главе получены следующие основные результаты:

- рассмотрены общие вопросы, связанные с теорией и практикой организации и применения МАС для управления потоками вычислительных заданий в РВС;
- рассмотрен ряд известных МАС (EMERALD, GAMA, JADE, JADEx и другие системы), определены их достоинства и недостатки;
- проведен детальный сравнительный анализ методов и средств организации таких систем;
- обоснована необходимость разработки нового инструментального комплекса разработки МАС для управления потоками вычислительных заданий;
- сформулированы системные и функциональные требования к разрабатываемому инструментальному комплексу.

Содержание данной главы отражено в публикациях [5–8, 10, 11].

Глава 2. Модели и алгоритмы мультиагентного управления потоками заданий

Во второй главе рассмотрена структура МАС [83], применяемой для управления вычислениями в РВС, организованной с использованием ресурсов Центра коллективного пользования (ЦКП) «Иркутский суперкомпьютерный центр СО РАН» (ИСКЦ) [84]. Концептуальная модель, предложенная в [85] и используемая для представления описания предметных областей научных приложений, расширена новыми структурами представления знаний агентов. Предложена новая ролевая модель их поведения. Разработаны алгоритмы функционирования агентов, включая алгоритм перераспределения вычислительных ресурсов для выполнения остаточной схемы решения задачи в РВС. Сформирована система обучения агентов. В заключение сформулированы выводы по результатам исследований, представленных в данной главе.

2.1. Структура мультиагентной системы

Диссертационная работа направлена на разработку моделей, алгоритмов и инструментальных средств, ориентированных на автоматизацию создания и применения агентов для МАС с заданной организационной структурой (рисунок 2.1). Ее иерархическая структура может включать два или более уровней функционирования агентов. На каждом уровне могут функционировать агенты, играющие различные роли и соответственно выполняющие различные функции. Роли агентов могут быть постоянными и временными, возникающими в дискретные моменты времени в связи с необходимостью организации коллективного взаимодействия. Уровни иерархии агентов отличаются объемом их знаний, агенты более высокого уровня иерархии обладают большим объемом знаний по сравнению с агентами более низкого уровня и, кроме того, могут обращаться к агентам любого ниже лежащего уровня с запросом на получение локальных знаний этих агентов. На каждом уровне агенты могут объединяться в виртуальные сообщества, кооперироваться и конкурировать в их рамках.

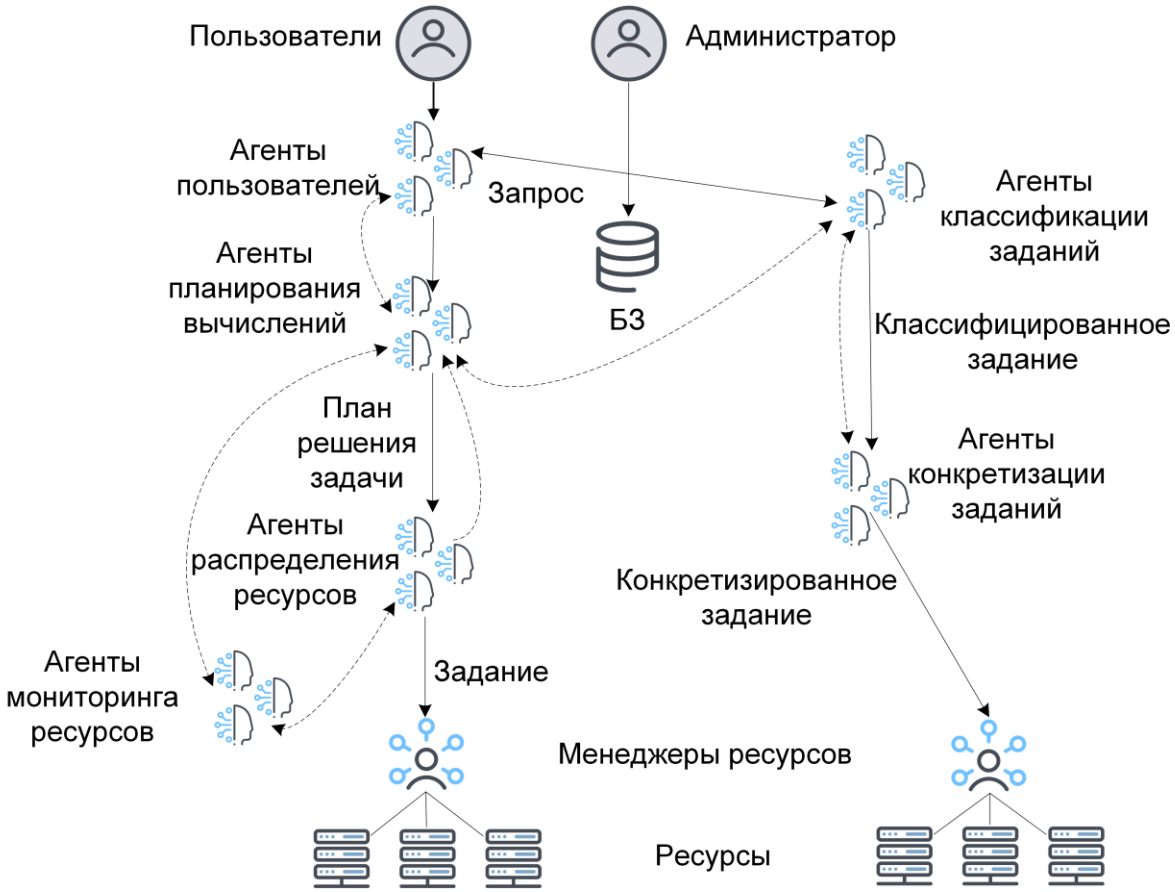


Рисунок 2.1 – Структура МАС

Координация действий агентов осуществляется с помощью общих правил группового поведения. Агенты функционируют в соответствии с заданными ролями и для каждой роли определены свои правила поведения в виртуальном сообществе агентов. Мультиагентная система включает агентов распределения ресурсов и управляющего агента. Задачей агентов на уровне РВС является получение такого распределения поступающих в систему потоков заданий, которое сохраняет показатели качества функционирования этой системы в заданных ее администратором пределах.

МАС включает агентов постановки задачи, планирования вычислений, мониторинга и распределения ресурсов, классификации, конкретизации и выполнения заданий, а также агентов параметрической настройки алгоритмов функционирования вышеперечисленных агентов на основе имитационного моделирования.

Схемы баз знаний агентов формируются на основе специализированной

концептуальной модели [85], которая в отличие от известных вычислительных моделей обеспечивает взаимосвязанное представление проблемно-ориентированного, программно-аппаратного, имитационного и управляющего слоев знаний о РВС. Тем самым обеспечивается проблемная ориентация МАС. Данные о текущем состоянии РВС передаются в ее базу знаний системой метамониторинга [86]. Эффективность и надежность функционирования разрабатываемых МАС исследуется с помощью полунатурного моделирования.

Рассматриваемая в диссертационной работе МАС ориентирована на управление потоками заданий распределенных пакетов прикладных программ (РППП), разрабатываемых с помощью специализированных инструментальных комплексов на основе САТУРН-технологии [87].

2.2. Концептуальная модель

Концептуальная модель является частным случаем семантической сети. Она обеспечивает представление знаний о программных модулях для решения задач в предметных областях и работы с объектами РВС, схемных знаний о модульной структуре модели и алгоритмов, продукционных знаний для поддержки принятия решений по выбору оптимальных алгоритмов в зависимости от состояния среды, а также знаний о программно-аппаратной инфраструктуре данной системы и административных политиках в ее узлах.

Пусть Z, F и M – это множества параметров, операций и программных модулей модели. Модули являются элементом вычислительных знаний. Параметры, операции и их взаимосвязи отражают схемные знания. Операции из F определяют вычислительные действия на множестве параметров Z.

Каждой операции $f_i \in F$ соответствует модуль $m_j \in M$, где $i \in \overline{1,n_f}$, $j \in \overline{1,n_m}$, n_f – число операций, n_m – число модулей. Один модуль может реализовывать несколько операций. Спецификация модуля включает язык программирования, тип и семантику входных, выходных и транзитных параметров, способы передачи параметров и обработки нестандартных ситуаций, модуль представления,

требуемый компилятор и другие сведения.

С каждой операцией f_i связано два множества параметров $Z_i^{in}, Z_i^{out} \subset Z$. Множество Z_i^{in} определяет параметры, значения которых необходимо задать, чтобы получить значения параметров, представленных множеством Z_i^{out}. Множества Z_i^{in} и Z_i^{out} представляют соответственно множества входных и выходных параметров модуля m_j, реализующего операцию f_i. Различаются базовые и составные операции. Составные операции могут включать базовые операции, а также конструкции для организации ветвления и итерации.

Постановки задач могут формулироваться в полной или сокращенной (процедурной или непроцедурной) форме. По сформулированной постановке задачи строится схема ее решения (абстрактная программа) на основе методов статического, динамического или статико-динамического планирования вычислений [88]. Оператор статико-динамического планирования может включаться в составную операцию. В общем случае в модели может существовать множество S эквивалентных схем решения задачи. Схема $s \in S$ определяет, какие операции и в какой последовательности должны быть выполнены для решения задачи. Полная постановка задачи s_f, совпадающая со схемой s, определяется структурой $s_f =< F_s, X_0, Y_0 >$, где $F_s \subset F$ – множество операций, которые нужно выполнить для решения задачи, $X_0 \subset Z$ – множество исходных параметров, значения которых заданы, $Y_0 \subset Z$ – множество целевых параметров, значения которых нужно вычислить.

В множествах параметров и операций модели введены соответственно подмножества системных параметров, отражающих характеристики объектов РВС, и операций, представляющих алгоритмы планирования вычислений, мониторинга и распределения ресурсов, моделирования вычислительных процессов и других действий в данной системе. В процессе эксплуатации РВС значения системных параметров определяются системой метамониторинга.

2.3. Модель поведения агентов

Функционирование агентов осуществляется с использованием парадигмы конечно-автоматного программирования. В диссертации разработана новая модель агента, представленная структурой

$$M^{agent} =< sm^p, \left\{ sm^c_{i,j} : i \in \overline{1, n_{vc}},\ j \in \overline{1, n_{rol}}, \right\}, MES >,$$

где sm^p – родительский автомат, $sm^c_{i,j}$ – дочерние автоматы, n_{vc} – число виртуальных сообществ, в которых состоит агент, n_{rol} – число ролей, которые может играть агент, MES – множество сообщений агента. Основной функцией родительского автомата sm^p является создание дочернего автомата $sm^c_{i,j}$ при каждом включении агента в новое виртуальное сообщество, где i и j – это соответственно номера виртуального сообщества и роли агента.

Модель родительского автомата представлена структурой

$$sm^p =< STS^p, sts^p_o, X^p_{inputs}, X^p_{outputs}, g^p, h^p, ACT^p >,$$

где STS^p – конечное множество управляющих состояний родительского автомата, $sts^p_0 \in STS^p$ – начальное состояние, $X^p_{inputs} \subset Z$ – конечное множество входных воздействий, порождаемых РВС, $X^p_{outputs} \subset Z$ – конечное множество выходных воздействий, $g^p : STS^p \times X^p_{inputs} \times ACT^p \to X^p_{outputs}$ – функция выходов, $h^p : STS^p \times X^p_{inputs} \to STS^p$ – функция переходов, $g^p, h^p \in F$, $ACT^p \subset F$ – конечное множество действий родительского автомата. В множество STS^p входит конечное состояние sts^p_{end}, в котором агент завершает свою работу. В множестве $X^p_{inputs} \cup X^p_{outputs}$ выделяется набор глобальных переменных GV родительского автомата, доступных дочерним автоматам. В общем случае $X^p_{inputs} \cap X^p_{outputs} \neq \varnothing$. Схема формирования X^p_{inputs} и $X^p_{outputs}$ представлена на рисунке 2.2.

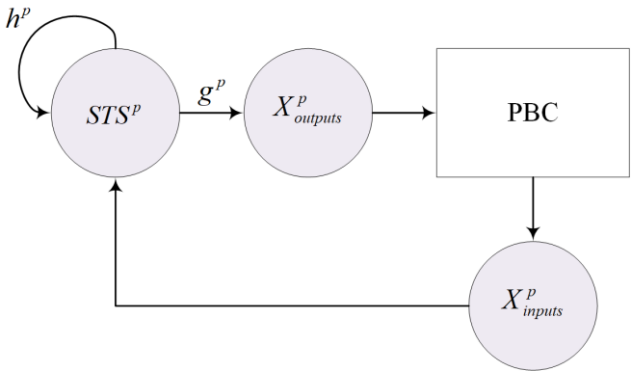


Рисунок 2.2 – Формирование входных и выходных воздействий родительского автомата

При создании агента все схемы выполнения действий, представляющие собой абстрактные программы, генерируются на языке программирования Java.

Модель дочернего автомата представлена структурой

$$sm^c_{i,j} =< STS^c_{i,j}, sts^c_{i,j,0}, X^c_{i,j,inputs}, X^c_{i,j,outputs}, g^c_{i,j}, h^c_{i,j}, ACT^c_{i,j} >,$$

где $sm^c_{i,j}$ – конечное множество управляющих состояний дочернего автомата, $sts^c_{i,j,0} \in sts^c_{i,j}$ – начальное состояние, $X^c_{i,j,inputs} \subset Z$ – конечное множество входных воздействий, порождаемых РВС, $X^c_{i,j,outputs} \subset Z$ – конечное множество выходных воздействий, $g^c_{i,j} : STS^c_{i,j} \times X^c_{i,j,inputs} \times ACT^c_{i,j} \to X^c_{i,j,outputs}$ – функция выходов, $h^c_{i,j} : STS^c_{i,j} \times X^c_{i,j,inputs} \to STS^c_{i,j}$ – функция переходов, $g^c_{i,j}, h^c_{i,j} \in F$, $ACT^c_{i,j} \subset F$ – конечное множество действий дочернего автомата. В множество $STS^c_{i,j}$ входит конечное состояние $sts^c_{i,j,end}$, в котором агент завершает свою работу. В общем случае $X^c_{i,j,inputs} \cap X^c_{i,j,outputs} \neq \varnothing$. Схема формирования $X^c_{i,j,inputs}$ и $X^c_{i,j,outputs}$ представлена на рисунке 2.3.

Дочерние автоматы разных агентов, входящих в одно ВС, взаимодействуют путем обмена сообщениями, передаваемыми через родительские автоматы. Автоматы, являющиеся потомками одного и того же родительского автомата, обмениваются информацией об использовании общих ресурсов агента через глобальные переменные родительского автомата и агентную базу знаний.

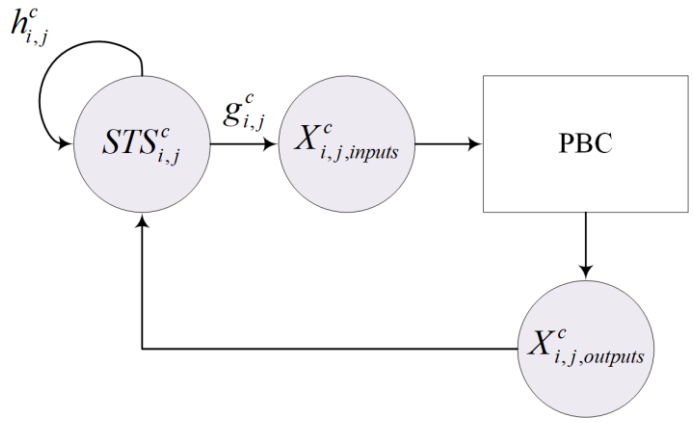


Рисунок 2.3 – Формирование входных и выходных воздействий дочернего автомата

Рассмотренные выше родительский и дочерний автоматы являются частными случаями конечного управляющего автомата, представленного в [89]. Их принципиальным отличием является то, что в качестве состояний автомата используются состояния-действия. В рамках одного состояния агент может выполнять одно или несколько действий. В каждом состоянии родительский (дочерний) автомат изменяет значения параметров функций переходов h^p ($h^c_{i,j}$) с помощью функции g^p ($g^c_{i,j}$), а затем осуществляет переход в новое состояние.

При вступлении агента в *i*-е виртуальное сообщество для соответствующего дочернего автомата создается система логического времени SLT_i, определяемая структурой $SLT_i = < T, T_m, g_t, g_m, g_r >$, где T – область значений логического времени, T_m – область значений временных маркеров датировки сообщений, $T_m \subseteq T$, g_t, g_m и g_r – функции датировки событий автомата, маркировки сообщений и сравнения значений логического времени $\forall i \in \overline{1, n_{vc}}$. Система логического времени использует векторные часы, в которых число компонент вектора времени равно числу агентов виртуального сообщества, и обеспечивает отношение частичного порядка на множестве событий виртуального сообщества с учетом их обусловленности. Применение функции датировки сообщений обеспечивает их обработку в установленной логической последовательности, а не в произвольном порядке поступления их в общий пул.

2.4. Расширение концептуальной модели

С целью поддержки построения и применения предложенной ролевой модели поведения агентов, концептуальная модель расширена новыми объектами: агентами, их ролями, виртуальными сообществами и отношениями между ними, а также состояниями, функциями и графами переходов автоматов. В качестве состояния агента мы используем состояние-действие – последовательность системных операций, выполняемых над полем системных параметров модели. Фрагмент такой модели, описывающей объекты, необходимые для построения графов переходов агентов, представлен на рисунке 2.4. Здесь G – множество графов переходов, A – множество агентов, VC – множество виртуальных сообществ агентов, R – множество ролей агентов и STS – множество состояния автоматов. На рисунке 2.4 отношения между объектами обозначены $o_1 - o_8$. Отношение o_4 представляет взаимосвязь состояний с операциями, реализующими функции переходов. Отношение o_5 представляет взаимосвязь состояний с остальными системными операциями. База знаний агента создается на основе фрагмента модели РВС, рассмотренного выше.

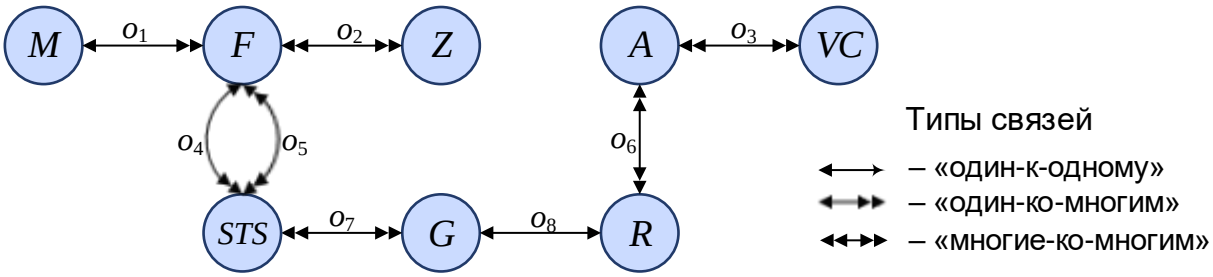


Рисунок 2.4 – Фрагмент описания расширенной концептуальной модели

2.5. Алгоритмы функционирования агентов

Выбор узлов вычислительного кластера, в которых будет выполняться задание, осуществляется с помощью алгоритма многоуровневого конкретизирующего планирования заданий с заданными критериями качества (показателями надежности, времени и стоимости) их выполнения. Процесс планирования осуществляется агентами, представляющими узлы вычислительного

кластера. Этапы работы алгоритма детально представлены в [83, 90]. Процесс планирования включает формирование всего множества доступных узлов; конкретизацию сформированного множества путем исключения из него перегруженных узлов (относительно текущей средней загрузки узлов с учетом имеющихся очередей заданий); построение поливариантного плана выполнения задания в узлах; извлечение из построенного поливариантного плана специализированного плана, удовлетворяющего заданным критериям качества выполнения задания с учетом текущего состояния РВС и назначение узлов вычислительного кластера для выполнения этого задания. Построение специализированного плана осуществляется на основе экономического механизма регулирования спроса и предложения вычислительных ресурсов.

В таблицах 3-5 представлены состояния-действия агентов, функции переходов $f_1 - f_{36} \in F$ и параметры $q_1 - q_{25} \in Z$ этих функций.

Таблица 3 – Состояния-действия

Состояние	Действия
AS_{start}	Ожидание готовности таймера перехода
AS_{end}	Завершение работы
AS_1	Проверка среды, инициализация
AS_2	Обработка очереди сообщений (отправка/получение)
AS_3	Формулировка постановки задачи
AS_4	Обработка сбойных постановок задач
AS_5	Построение плана решения задачи
AS_6	Обработка сбойных планов решения задач
AS_7	Передача задач агенту классификатору
AS_8	Передача классифицированного задания агенту формирования *i*-го ВС
AS_9	Обслуживание дочерних автоматов *i*-го ВС
AS_{10}	Проверка соответствия области допустимых значений характеристик задания характеристикам имеющихся классов вычислительных ресурсов
AS_{11}	Формирование множества допустимых для распределения ресурсов
AS_{12}	Передача классифицированного задания агенту формулировки постановки задачи и построения плана ее решения
AS_{13}	Подготовка уведомления о формировании нового ВС, топологии ВС и классифицированного задания агентам, представляющим ресурсы
AS_{14}	Обработка полученных ставок и определение агента координатора *i*-го ВС

Состояние	Действия
AS_{15}	Подготовка уведомления о завершении торгов и распределении модулей задания между агентами
AS_{16}	Инициализация работы *i*-го ВС, рассылка модулей на выполнение ресурсным агентам
AS_{17}	Опрос состояния агентов *i*-го ВС, получение информации от агента мониторинга
AS_{18}	Обработка отказа, выбор сценария
AS_{19}	Формирование и отправка остаточной схемы агенту постановки задач
AS_{20}	Обработка завершенных заданий и отправка сообщения родительскому автомату о завершении выполнения модулей в *i*-м ВС
AS_{21}	Обработка результатов вычисления, расформирование *i*-го ВС, завершение работы координатора *i*-го ВС
AS_{22}	Запуск дочернего автомата агента, выполняющего модули задания в *i*-м ВС
AS_{23}	Передача модулей менеджеру ресурсов на выполнение
AS_{24}	Восстановление связи с координатором
AS_{25}	Обработка завершенных заданий
AS_{26}	Организация перевыборов координатора
AS_{27}	Завершение работы дочернего автомата агента, выполняющего модули задания в *i*-м ВС
AS_{28}	Вступление в *i*-е ВС, формирование ставки за модули (аукцион Викри), формирование предложения по выбору координатора
AS_{29}	Отправка сообщений агенту координатору о завершении выполнения модулей
AS_{30}	Запуск дочернего автомата агента координатора *i*-го ВС

Таблица 4 – Функции переходов

Функция	Условие перехода
f_1	$\overline{q_1} \wedge q_2$
f_2	q_1
f_3	$\overline{q_1} \wedge \overline{q_2} \wedge q_3$
f_4	$\overline{q_3}$
f_5	$\overline{q_1} \wedge \overline{q_2} \wedge \overline{q_3}$
f_6	$\overline{q_1} \wedge \overline{q_2} \wedge \overline{q_3} \wedge \overline{q_{18}}$
f_7	$\overline{q_1} \wedge \overline{q_2} \wedge \overline{q_3} \wedge \overline{q_{11}}$
f_8	$\overline{q_1} \wedge \overline{q_2} \wedge \overline{q_3} \wedge \overline{q_{25}} \wedge \overline{q_8}$
f_9	$\overline{q_1} \wedge \overline{q_2} \wedge \overline{q_3} \wedge \overline{q_{12}} \wedge \overline{q_{23}}$
f_{10}	q_4

Функция	Условие перехода
f_{11}	q_6
f_{12}	q_7
f_{13}	q_9
f_{14}	q_{10}
f_{15}	1
f_{16}	$\overline{q_1} \wedge \overline{q_2} \wedge \overline{q_3} \wedge q_{11}$
f_{17}	$\overline{q_1} \wedge \overline{q_2} \wedge \overline{q_3} \wedge \overline{q_{12}} \wedge q_{23}$
f_{18}	$\overline{q_1} \wedge \overline{q_2} \wedge \overline{q_3} \wedge q_{12}$
f_{19}	$\overline{q_5} \wedge q_{21}$
f_{20}	q_{16}
f_{21}	q_{13}
f_{22}	q_{14}
f_{23}	$\overline{q_1} \wedge \overline{q_2} \wedge \overline{q_3} \wedge q_{18}$
f_{24}	q_{19}
f_{25}	q_{21}
f_{26}	q_{22}
f_{27}	q_{24}
f_{28}	$\overline{q_1} \wedge \overline{q_2} \wedge \overline{q_3} \wedge q_{25}$
f_{29}	$\overline{q_1} \wedge \overline{q_2} \wedge \overline{q_3} \wedge \overline{q_{25}} \wedge q_8$
f_{30}	q_{28}
f_{31}	q_5
f_{32}	$\overline{q_1} \wedge \overline{q_2} \wedge \overline{q_3} \wedge \overline{q_{17}} \wedge q_{15}$
f_{33}	$\overline{q_1} \wedge \overline{q_2} \wedge \overline{q_3} \wedge q_{17}$
f_{34}	$\overline{q_1} \wedge \overline{q_2} \wedge \overline{q_3} \wedge \overline{q_{17}} \wedge \overline{q_{15}}$
f_{35}	q_{20}
f_{36}	$\overline{q_5} \wedge \overline{q_{21}}$

Таблица 5 – Параметры функций переходов

Параметр	Описание
q_1	Флаг выхода
q_2	Таймер перехода не готов
q_3	Система не инициализирована
q_4	Наличие необработанных задач

Параметр	Описание
q_5	Наличие предложения вступить в *i*-е ВС
q_6	Успешно обработаны все новые задачи
q_7	Сформулирована постановка задачи
q_8	Наличие завершенных модулей агентов *i*-го ВС
q_9	Успешное построение планов для всех новых задач
q_{10}	Построен план решения задачи
q_{11}	Наличие новых классифицированных заданий
q_{12}	Готовность таймера проверки дочерних автоматов
q_{13}	Наличие новых классифицированных заданий
q_{14}	Сформировано новое уведомление для рассылки агентам *i*-го ВС
q_{15}	Наличие завершенных модулей данного агента
q_{16}	Завершено выполнение всех модулей данного агента
q_{17}	Потеряна связь с координатором
q_{18}	Наличие новых сообщений от агентов *i*-го ВС
q_{19}	Выборы агента координатора *i*-го ВС завершены успешно
q_{20}	Сформировано новое уведомление о завершении торгов и распределении модулей задания между агентами
q_{21}	Получено уведомление о назначении роли координатора *i*-го ВС
q_{22}	*i*-е ВС успешно инициализировано
q_{23}	Завершены все модули *i*-го ВС
q_{24}	Сформировано новое сообщение для агентов *i*-го ВС
q_{25}	Обнаружены признаки отказов

На различных этапах функционирования агентов между ними происходит обмен сообщениями в формате JavaScript Object Notation (JSON) [91]. Сообщения содержат следующую основную информацию:

- идентификатор контейнера JADE, в котором работает агент;
- идентификатор агента;
- сервисная информация;
- тело сообщения.

Свойства формата JSON позволяют в текстовом формате передавать любые данные в виде «ключ»: «значение», к том числе и массивы. Это достоинство используется для формирования и передачи ставок агентов, передачи списка

параметров, выполняемых модулей. Каждый агент содержит необходимые сведения для интерпретации сообщений.

Ограничения корректности функций, определяющих условия переходов, приведены в таблице 6. Здесь функция f_{li} определяет переход между состояниями агента в графе G, $l=\overline{1,n_v}$, $i=\overline{1,n_{lf}}$, n_v – число вершин графа G, n_{lf} – число функций, определяющих переходы из вершины v_l графа G.

Таблица 6 – ограничения корректности функций, определяющих условия переходов

n_{lf}	Ограничение
1	$f_{l1}\equiv 1,$ $l=\overline{1,n_v}$
2	$f_{l1}f_{l2}\vee\overline{f}_{l1}\overline{f}_{l2}=0$
$i=\overline{3,n_{lf}}$	$a\vee b=0,$ $a=\vee_{i=3}^{n_{lf}-1}\vee_{k=i+1}^{n_{lf}}(f_{li}\wedge f_{lk}),$ $b=\wedge_{i=3}^{n_{lf}}\overline{f}_{li},$ $q_r=0,\overline{q}_s=0,r\neq s,r,s\in\overline{1,n_q}.$

Функции переходов, представленные в таблице 4, удовлетворяют ограничения их корректности (таблица 6). В частности, таблицы истинности для функций, определяющих три или более переходов из одного состояния, приведены в Приложении Г. Случаи с функциями, определяющими не более двух переходов из одного состояния, являются очевидными для всех рассмотренных графов переходов, а также легко проверяются аналитически.

Графы переходов основных рассматриваемых агентов представлены на рисунках 2.5-2.10. Каждая вершина графа представлена состоянием-действием, а условием перехода в данное состояние является удовлетворение условиям соответствующих функций перехода, представленных в таблице 4. Для выполнения модулей очередной задачи пользователя формируется ВС, индекс которого в таблицах представлен через *i*.

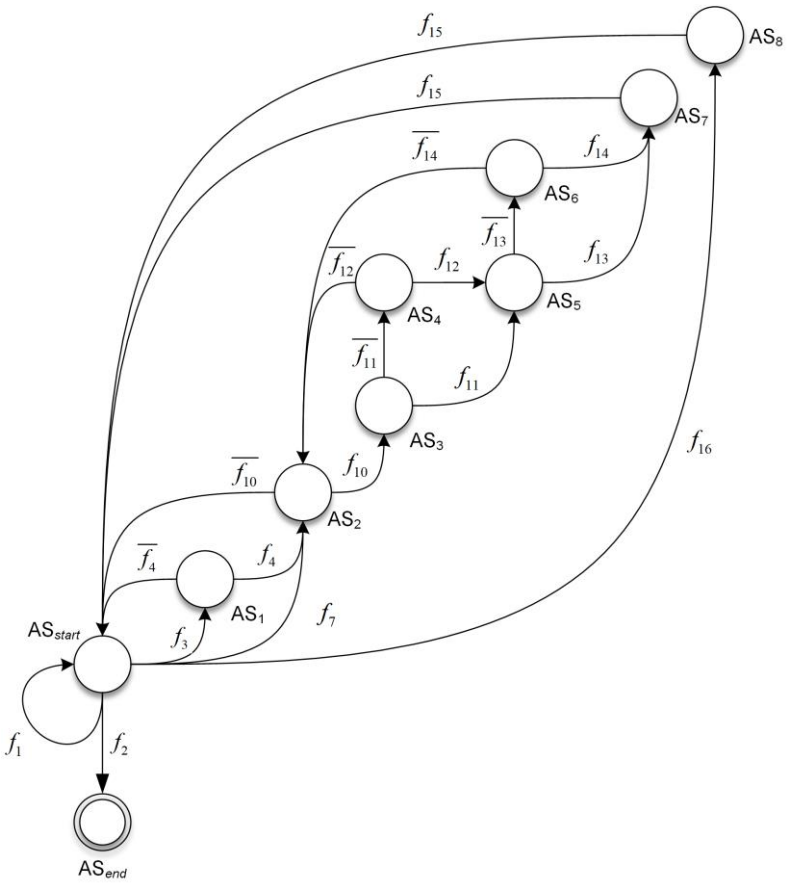


Рисунок 2.5 – Граф переходов родительского автомата для агента формулировки постановки задачи и построения плана ее решения

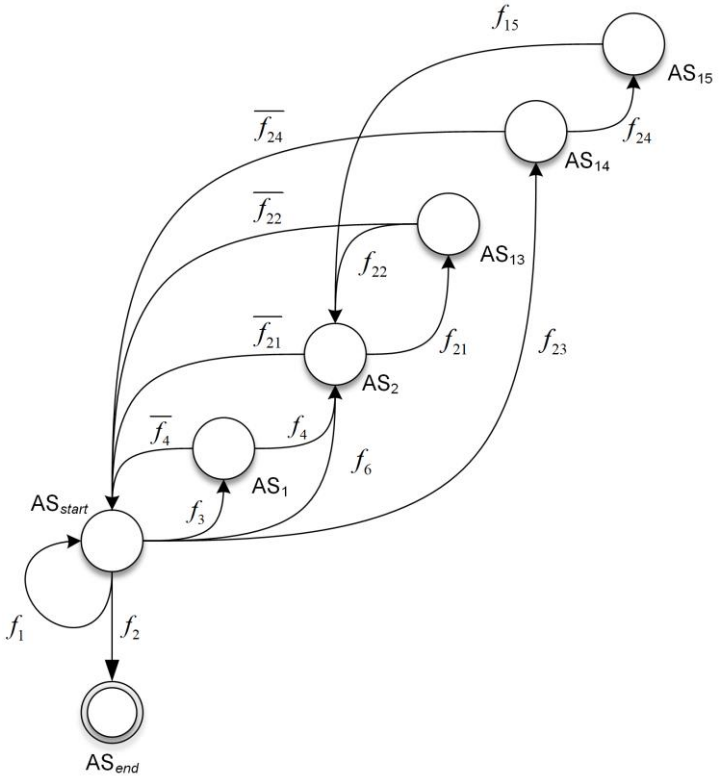


Рисунок 2.6 – Граф переходов родительского автомата для агента организации ВС

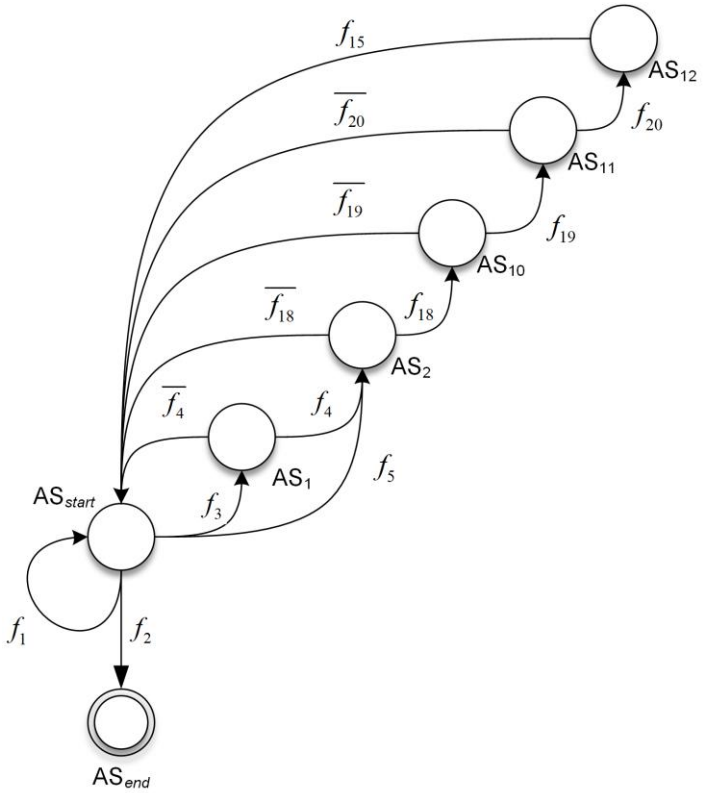


Рисунок 2.7 – Граф переходов родительского автомата агента классификации заданий

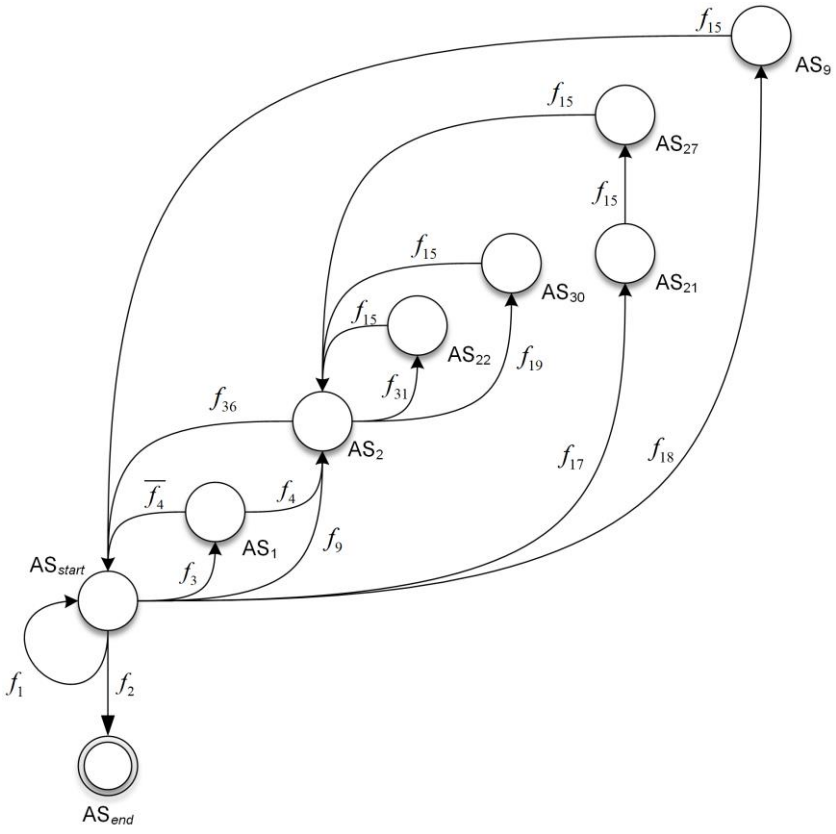


Рисунок 2.8 – Граф переходов родительского автомата агента, представляющего ресурсы среды

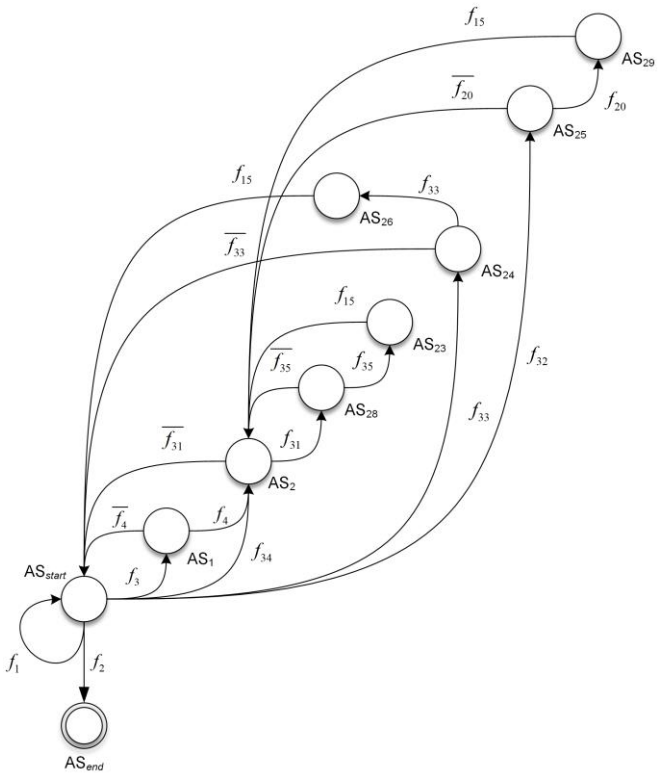


Рисунок 2.9 – Граф переходов дочернего автомата для агента, представляющего ресурсы среды и функционирующего в роли ресурсного агента

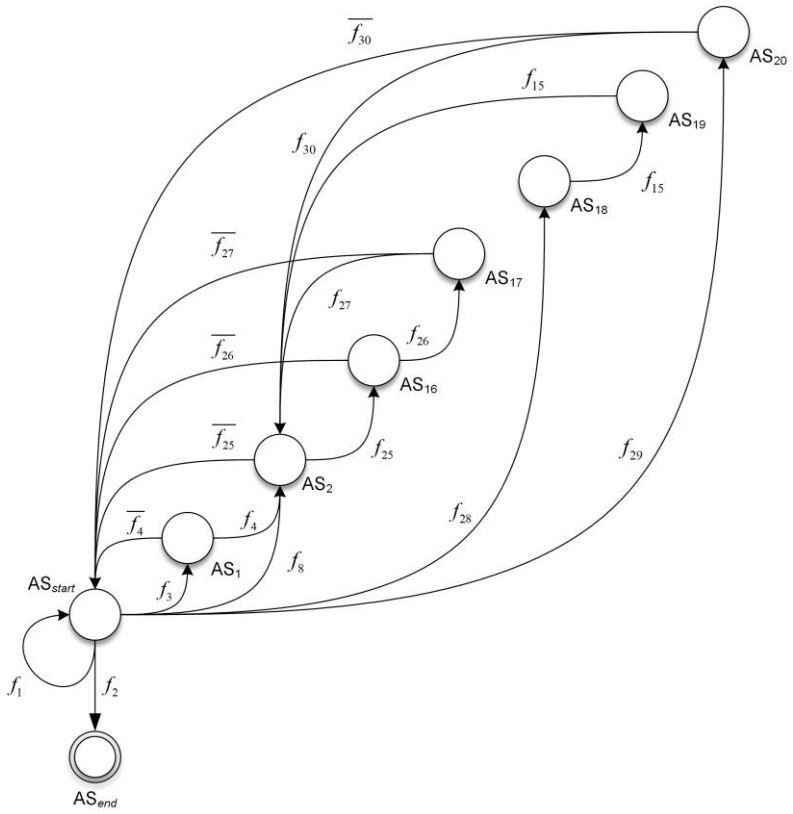


Рисунок 2.10 – Граф переходов дочернего автомата для агента, представляющего ресурсы среды и функционирующего в роли координатора ВС

2.6. Алгоритм перераспределения ресурсов среды

В настоящее время обеспечение отказоустойчивости вычислительного процесса в РВС, например, в Grid-системах или облачных платформах, по-прежнему остается актуальной проблемой.

В этой связи в диссертации разработан новый мультиагентный алгоритм для перераспределения вычислительных ресурсов РВС в случае отказа процесса решения задачи. Остаточная схема решения задачи формируется с использованием методов конкретизирующего программирования. В отличие от известных алгоритмов подобного назначения предложенный алгоритм реализует адаптивное мультисценарное решение данной задачи и тем самым повышает степень отказоустойчивости вычислительного процесса.

Проблема восстановления вычислительного процесса (задания) вследствие отказа программно-аппаратного обеспечения актуальна при решении больших фундаментальных и прикладных задач в разнородных РВС. Существуют различные подходы к решению данной проблемы [86, 92].

В частности, широко применяемым на практике подходом к перезапуску заданий является механизм контрольных точек [93]. Однако формирование пользовательских контрольных точек в узле вычислительной системы, в котором выполняется вычислительный процесс, не позволяет произвести рестарт в случае отказа самого узла. Создание же системных контрольных точек на уровне ОС, обеспечивающих перенос задания в другие узлы, влечет существенные накладные расходы и поддерживается далеко не всеми средствами управления вычислениями.

В данной главе рассматривается алгоритм восстановления вычислительного процесса после его отказа в РВС, где управление вычислениями реализуется МАС. В отличие от других подходов к управлению вычислениями и обеспечению их надежности [94, 95] предложенный подход базируется на реализации адаптивного мультисценарного алгоритма решения данной задачи.

2.6.1. Остаточная схема решения задачи

В общем случае s, определенная в разделе 2.2, является поливариантной

схемой решения задачи и описывает альтернативные алгоритмы выполнения вычислительного процесса. Заменим множество операций F_s в описании s на множество модулей M_s, которые реализуют эти операции.

На рисунке 2.11 приведены примеры поливариантной схемы решения задачи (а) и двух ее специализированных вариантов: с использованием для решения задачи модуля m_5 (б) и модуля m_6 (в), которые вычисляют значения одного и того же параметра, но применяют разные алгоритмы или обладают различными требованиями к ресурсам и показателями эффективности решения задачи.

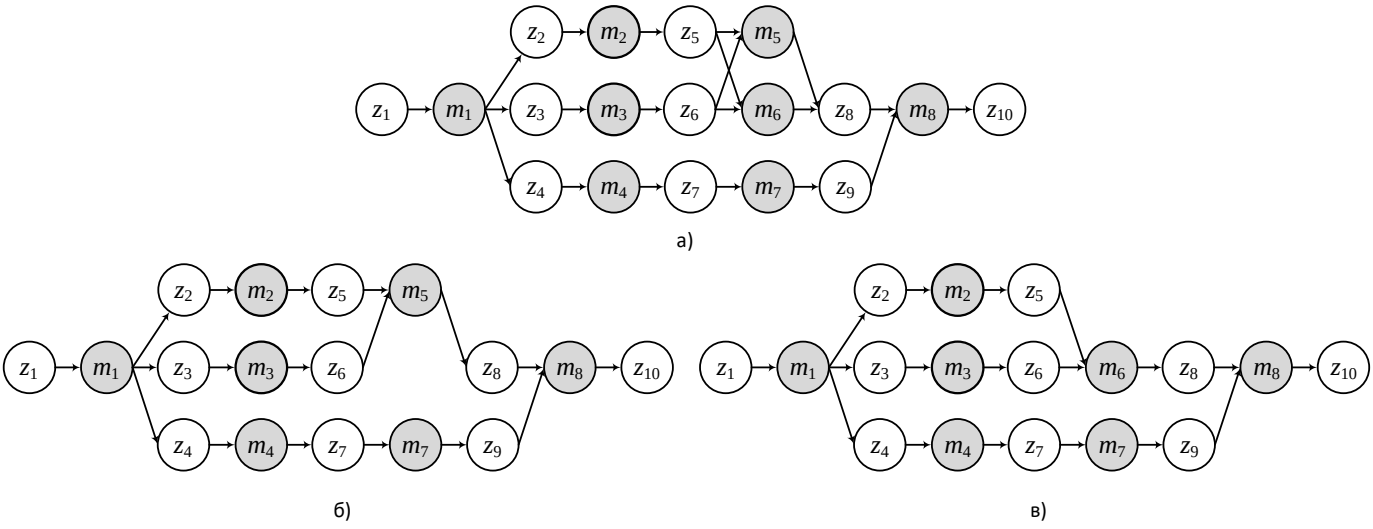


Рисунок 2.11 – Схемы решения задачи: поливариантная (а), схема с использованием модуля m_5 (б), схема с использованием модуля m_6 (в)

Выполнение схемы решения задачи s осуществляется путем ее интерпретации. Пусть в процессе интерпретации схемы s выполнено множество $M_e \subset M_s$ модулей, $|M_e| = n_e$. Тогда множество $M_u = M_s \setminus M_e$ будет включать модули схемы s, которые уже выполняются или ожидают своего запуска, $|M_e| = n_u$. Остаточной схемой решения задачи будем называть схему s_u, определяемую структурой

$$< M_u, X_u, Y_u >, \qquad (1)$$

$$X_u = X_o \cup \left(\bigcup_{l=1}^{n_e} Z_{i_l}^{out} \right), \quad (2)$$

$$Y_u = Y_o \cap \overline{\left(X_o \cup \left(\bigcup_{l=1}^{n_e} Z_{i_l}^{out} \right) \right)}, \quad (3)$$

где X_u и Y_u представляют входные и выходные параметры схемы, $i_l \in \overline{1, n_m} : m_{i_l} \in M_e$, множество X_u представляет промежуточные результаты вычислений. Остаточная схема является адаптацией понятия остаточной программы, введенной в [96].

На рисунке 2.12 изображен фрагмент исходной схемы (рис. 2.11, а) с параметрами и модулями, относящимися к остаточной схеме решения задачи. Параметры $z_1 - z_6$ представляют собой результат промежуточных вычислений. Темно-серым цветом заливки выделены модули m_4 и m_5, которые выполняются. Модули m_7 и m_8 ожидают своего запуска. Модуль m_6 реализует вычислительную избыточность схемы и может быть использован в том случае, если решение задачи с использованием модуля m_5 будет невозможно.

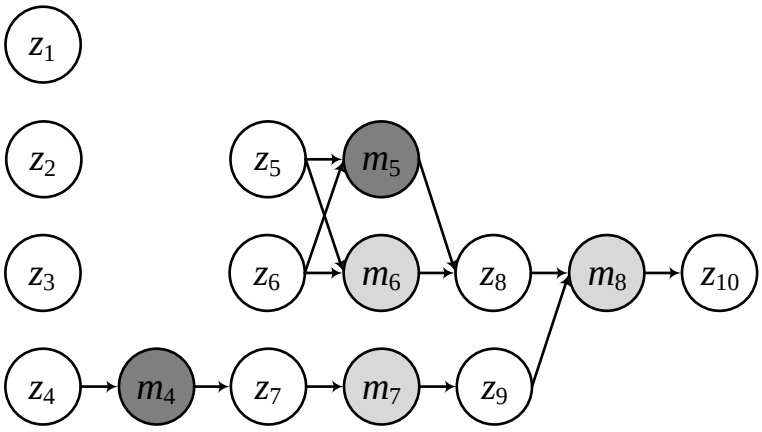


Рисунок 2.12 – Остаточная схема решения задачи

2.6.2. Отказы

Информационно-вычислительная модель системы диагностики представлена в виде следующей структуры $S =< O, P, T, Z, F, R, PR, L, I >$, где O – множество исследуемых объектов в узлах, P – множество измеряемых характеристик

(параметров) объектов, T – множество типов значений характеристик, Z – множество логических параметров, F – множество контрольно-диагностических операций (действий), R – множество типов операций, PR – множество продукций, L – журнал диагностики, I – периоды запуска диагностики вычислительного узла, соответствующие режимам эксплуатации узла в разные временные интервалы.

В системе метамониторинга рассматриваются следующие характеристики узлов, модулей и агентов: характеристики объемов вычислительной нагрузки компонентов узла (нагрузки процессоров, ядер, оперативной памяти (ОЗУ), сетевых элементов, систем хранения данных и других структурных элементов); характеристики физического состояния компонентов узла (температура процессоров и материнских плат, работоспособность систем бесперебойного питания, жестких дисков и других структурных элементов); характеристики процесса выполнения модуля (приоритет и статус процесса, использованное процессорное время, объем используемой ОЗУ, число обращений к жесткому диску и сетевым элементам и другие сведения); характеристики работы агента (аналогичные выполнению модуля на узле, дополненные характеристиками взаимодействия агента с другими агентами).

К основным типам отказов, идентифицируемых в системе метамониторинга относятся нештатное завершение вычислительного процесса; нехватка оперативной памяти для выполнения процесса; сбои операций чтения/записи данных; недоступность требуемого объема свободного места в системе хранения данных для записи результатов вычислений; отсутствие доступа к предметным базам данных; нарушение взаимодействия с другими узлами и агентами РВС (в том числе сбои передачи данных); отказы ОС узла, приводящие к потере его работоспособности; отказы аппаратных средств (систем охлаждения, сетевых интерфейсов, модулей памяти, процессоров и других устройств).

Виды, признаки и причины отказов объектов РВС с различной степенью их детализации описаны в [97]. В диссертации рассматриваются следующие основные

объекты РВС: вычислительные узлы, агенты и модули РППП.

В соответствии с выделенными объектами учитываются следующие отказы: отказ узла (узел находится в нерабочем состоянии либо не отвечает в течение заданного периода времени); отказ агента (агент не отвечает в течение заданного периода времени); отказ модуля (аварийное завершение выполнения модуля или некорректное вычисление его выходных параметров). Эти отказы обобщают различные причины и признаки неисправностей рассматриваемых объектов. Отказы приводят к необходимости выполнения модуля схемы решения задачи в резервном узле или другим агентом, передачи управления модулем иному агенту или выполнения другого модуля.

В отличие от известных средств мониторинга используемая система метамониторинга обладает следующими возможностями:

- возможность получения данных из самых популярных высокопроизводительных систем мониторинга (Ganglia, Nagios, Zabbix);
- широкий спектр традиционных и оригинальных функций для получения и сбора данных об аппаратных и программных компонентах узлов РВС;
- инструментарий высокого уровня для реализации функций в виде модулей на разных языках программирования для получения и сбора данных;
- специализированные инструменты для сбора и анализа данных инженерной инфраструктуры компьютерных кластеров и центров обработки данных;
- вспомогательные инструменты для унификации и агрегирования данных, полученных из разных источников;
- новые инструменты для автоматизированного экспертного анализа данных и генерации управляющих воздействий для устранения неисправностей, если это возможно;
- новые инструментальные средства для периодической проверки узлов, а также обнаружение, диагностика и частичное восстановление ошибок узлов;
- специальный API на основе открытых стандартов для обеспечения доступа к данным мониторинга для внешнего ПО.

2.6.3. Сценарии обработки отказов

МАС использует алгоритм перераспределения вычислительных ресурсов РВС для улучшения отказоустойчивости процесса решения задачи. Алгоритм включает следующие основные этапы работы:

I. Обнаружение отказа.

II. Обработка отказа.

III. Выбор сценария для обеспечения отказоустойчивости выполнения схемы решения задачи.

IV. Построение остаточной схемы решения задачи.

На первом шаге определение и исправление сбоя осуществляется средствами системы метамониторинга.

Далее агенты этой системы запускают процедуру обработки сбоя, включающую частичное устранение последствий и реконфигурацию множества доступных ресурсов. Кроме того, они собирают и передают необходимую информацию агентам виртуального сообщества.

На основе полученных данных агенты выбирают сценарий, обеспечивающий отказоустойчивость выполнения схемы решения задачи. Когда узел агента выходит из строя во время выполнения модуля, агент может принять независимое решение выполнить этот модуль на одном из резервных узлов. В остальных случаях агенты выбирают сценарий коллективно.

На последнем этапе агенты в процессе торгов борются за право выполнить модули остаточной схемы решения задачи в соответствии с выбранным сценарием по формулам (1)-(3). Формирование схемы осуществляется на основе методов редукции избыточных вычислений.

Рисунок 2.13 иллюстрирует сценарии обработки перечисленных выше отказов. Состав виртуального сообщества агентов, выполняющих схему решения задачи, обновляется после назначения новых агентов для выполнения модулей. Выделение ресурсов агентами осуществляется с учетом различий вычислительных характеристик узлов РВС при реализации многоуровневого параллелизма

алгоритма решения задачи.

Модуль может быть выполнен другим агентом в случае всех трех отказов. Кроме того, при отказе узла модуль может быть выполнен также в резервном узле. Дополнительно, при отказе модуля выполнявшейся схемы решения задачи, может быть выполнен модуль эквивалентной схемы решения задачи.

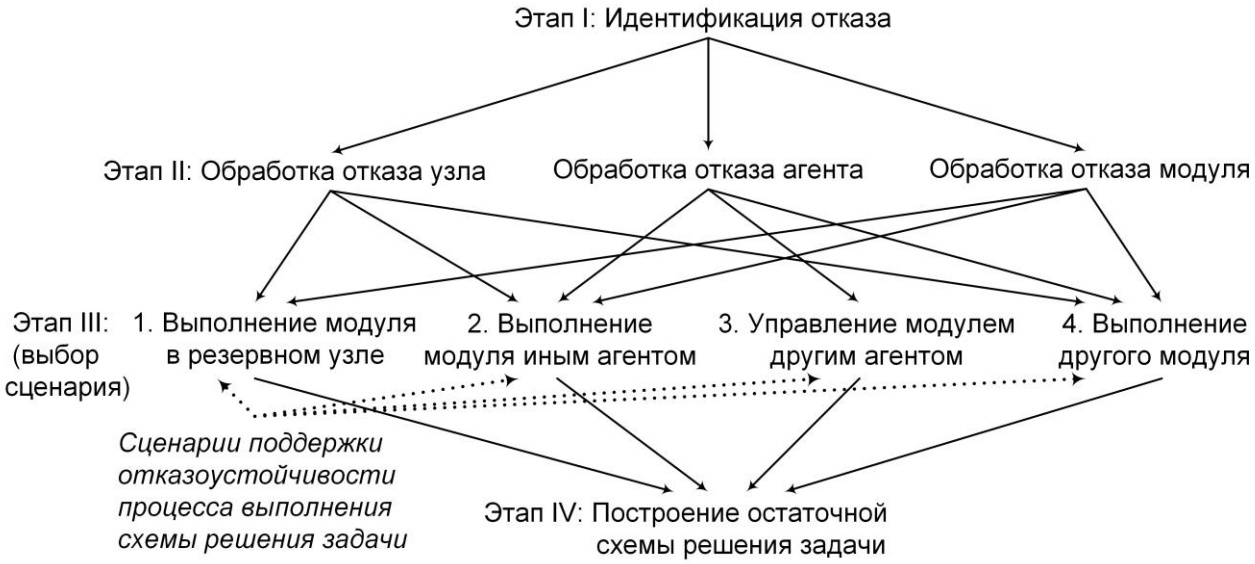


Рисунок 2.13 – Сценарии обработки отказов

Есть три возможных варианта назначения агента для выполнения модуля:

- централизованное (директивное) назначение модуля другому агенту, имеющему возможность выполнить данный модуль, координатором выполнения схемы решения задачи;
- децентрализованное назначение модуля путем проведения торгов между агентами, которые могут выполнить данный модуль;
- децентрализованное назначение модулей остаточной схемы решения задачи путем проведения торгов между агентами, которые могут принять участие в выполнении модулей остаточной схемы.

2.6.4. Алгоритм

Пусть Q – это множество простых и составных логических параметров, принимающих значения из множества $\{0,1,\theta\}$, $Q \subset Z$. Составной параметр

реализует логическое выражение, сформированное из простых параметров и логических операторов. Он не определен, если не определен хотя бы один из простых параметров. F и PR – это соответственно множества действий по обработке отказов и продукций, задающих условия применения действий: $pr_i : q_j \Rightarrow f_k$, где $q_j \Rightarrow f_k$ – ядро продукции, интерпретируемое как выбор действия $f_k \in F$ по обработке отказа в соответствии с условием $q_j \in Q$. Каждая продукция имеет приоритет.

Связи между продукциями и логическими параметрами в левых частях их ядер представлены булевой матрицей W размерности $n_p \times n_z$, где n_p – число продукций, n_z – число логических параметров. Элемент матрицы $w_{i,j} = 1$ означает, что продукция pr_i использует параметр q_j. Связи между продукциями и действиями представлены булевой матрицей C размерности $n_p \times n_f$, где n_f – число действий. Элемент матрицы $c_{i,k} = 1$ означает, что продукция pr_i описывает условия выполнения действия f_k. Зависимости между действиями представим булевой матрицей D размерности $n_f \times n_f$. Элемент матрицы $d_{i,j} = 1$ означает, что действие f_i зависит от действия f_j.

Параметры $x_1, x_2, x_3, x_4, x_5 \in \{0,1\}$ определяют соответственно работоспособность узла, наличие резервного узла, работоспособность модуля, наличие резервного модуля и работоспособность агента. Обозначим через n_{sc} номер сценария. Будем считать, что $n_{sc} = 0$ в случае полностью работоспособной ситуации и $n_{sc} = -1$, когда ни один из сценариев нельзя применить.

Работа алгоритма строится следующим образом:

1. Формулировка непроцедурной постановки задачи: «вычислить Y_0 по X_0», где X_0 – это множество параметров, содержащих информацию о диагностируемых объектах, а Y_0 – множество параметров, содержащих информацию о результатах диагностики.

2. Формирование множества $PR^* = \left\{ pr_{i_1}, pr_{i_2}, ..., pr_{i_q} \right\}$ продукций, для которых определены истинные значения параметров $q_{j_1}, q_{j_2}, ..., q_{j_l} : w_{i_h, j_h} = 1 \ \forall h = \overline{1,l}.$
3. Если $PR^* = \varnothing$, то переход на шаг 9 (задача неразрешима).
4. Иначе $\forall h = \overline{1,l}$:
   a. выполнение действия $f_k : c_{i_h,k} = 1, \ Z_k^{in} \subseteq X_0$;
   b. если действие f_k выполнено, то $X_0 = X_0 \cup Z_k^{out}$, $F = F \setminus \left\{ f_k \right\}$, $PR = PR \setminus \left\{ p_{i_h} \right\}$.
5. Если $Y_0 \subseteq X_0$, то переход на шаг 7.
6. Иначе переход на шаг 2.
7. Выбор сценария в соответствии с таблицей решений (таблица 7).
8. Построение остаточной схемы решения задачи.
9. Завершение работы алгоритма.

Таблица 7 – Таблица решений

x_1	x_2	x_3	x_4	x_5	n_{sc}
0	0	0	0	0	-1
0	0	0	0	1	-1
0	0	0	1	0	4
0	0	0	1	1	4
0	0	1	0	0	2
0	0	1	0	1	2
0	0	1	1	0	2
0	0	1	1	1	2
0	1	0	0	0	-1
0	1	0	0	1	-1
0	1	0	1	0	4
0	1	0	1	1	4
0	1	1	0	0	2
0	1	1	0	1	1
0	1	1	1	0	2
0	1	1	1	1	1

x_1	x_2	x_3	x_4	x_5	n_{sc}
1	0	0	0	0	-1
1	0	0	0	1	-1
1	0	0	1	0	4
1	0	0	1	1	4
1	0	1	0	0	3
1	0	1	0	1	0
1	0	1	1	0	3
1	0	1	1	1	0
1	1	0	0	0	-1
1	1	0	0	1	-1
1	1	0	1	0	4
1	1	0	1	1	4
1	1	1	0	0	3
1	1	1	0	1	0
1	1	1	1	0	3
1	1	1	1	1	0

Обработка предусловий продукций обеспечивает адаптацию работы алгоритма к текущему состоянию РВС и выбор наиболее подходящего сценария действий по устранению отказа. Для децентрализованного назначения модулей и резервирования узлов используются алгоритмы, предложенные в [98].

Пример 1. Рассмотрим пример идентификации отказа вычислительного узла, в котором выполняется модуль. В рамках концептуальной модели определим множества параметров, логических и вычислительных операций, а также продукций, определяющих условия выполнения вычислительных операций.

Множество параметров:

- z_1 – ОЗУ вычислительного узла;
- z_2 – модуль, выполняемый на узле;
- z_3 – тип проводимой диагностики;
- z_4 – общий объем ОЗУ;
- z_5 – используемый объем ОЗУ в процентах;

- z_6 – идентификационный номер неисправного модуля ОЗУ;
- z_7 – статус отправки уведомления администратору РВС;
- z_8 – статус отправки уведомления координатору виртуального сообщества агентов, запустившего модуль z_2 в узле;
- z_9 – статус команды для остановки выполнения модуля z_2;
- z_{10} – работоспособность узла;
- z_{11} – наличие резервного узла;
- z_{12} – работоспособность модуля z_2;
- z_{13} – наличие альтернативного модуля;
- z_{14} – работоспособность агента, представляющего узел;
- q_1 – признак проведения быстрой диагностики;
- q_2 – признак доступности неполного объема ОЗУ;
- q_3 – признак наличия неисправных модулей ОЗУ;
- q_4 – признак достижения критического значения объема используемой ОЗУ.

Множество логических операций (в списке параметров каждой операции ее входные и выходные параметры разделены знаком ‘;’):

- $f_1^l(z_3;q_1)$ – определение признака проведения быстрой диагностики;
- $f_2^l(z_4;q_2)$ – определение признака доступности неполного объема ОЗУ;
- $f_3^l(z_6;q_3)$ – определение признака наличия неисправных модулей ОЗУ;
- $f_4^l(z_5;q_4)$ – определение признака достижения критического значения объема используемой ОЗУ.

Для вычисления значений логических параметров $q_1 - q_4$ в операциях $f_1^l - f_4^l$ определены следующие условия:

$$q_1 = \begin{cases} 1, \textit{ если } z_3 = "quick", \\ 0 \textit{ в противном случае,} \end{cases}$$

$$q_2 = \begin{cases} 1, \textit{ если } z_4 \neq 2^{33}, \\ 0 \textit{ в противном случае}, \end{cases}$$

$$q_3 = \begin{cases} 1, \textit{ если } z_6 > 0, \\ 0 \textit{ в противном случае}, \end{cases}$$

$$q_4 = \begin{cases} 1, \textit{ если } z_5 > 99, \\ 0 \textit{ в противном случае}. \end{cases}$$

Множество вычислительных операций:

- $f_1(z_1; z_4, z_5)$ – получение информации о ОЗУ;
- $f_2(z_1; z_6)$ – идентификация неисправного модуля ОЗУ;
- $f_3(z_1, z_6; z_7)$ – отправка уведомления администратору РВС;
- $f_4(z_2, z_5; z_8, q_5 = 1)$ – отправка уведомления координатору виртуального сообщества, запустившего модуль z_2 в узле;
- $f_5(z_2, z_8; z_9, q_6 = 1)$ – остановка выполнения модуля z_2;
- $f_6(z_1, z_2, z_9; z_{10}, z_{11}, z_{12}, z_{13}, z_{14})$ – проверка статуса выполнения модуля z_2.

Множество продукций:

$$pr_1 : q_1 \Rightarrow f_1;$$

$$pr_2 : q_2 \Rightarrow f_2;$$

$$pr_3 : q_3 \Rightarrow f_3;$$

$$pr_4 : q_4 \Rightarrow f_4;$$

$$pr_5 : q_5 \Rightarrow f_5;$$

$$pr_6 : q_6 \Rightarrow f_6.$$

Пусть сформулирована следующая постановка задачи: «Зная $z_1, z_2, z_3 = "quick"$ вычислить z_{10}, z_{11}, z_{12}».

В результате планирования действий на множестве логических операций $f_1^l - f_4^l$ можно выполнить операцию f_1^l. В результате ее выполнения получаем $q_1 = 1$, что означает проведение быстрой диагностики.

Так как значения логических параметров $q_2 - q_6$ не определены, то можно применить только одну продукцию pr_1. В результате ее применения будет выполнена вычислительная операция f_1.

В результате ее выполнения множество исходных параметров будет расширено. С учетом вычисленных значений логических параметров оно будет включать следующие параметры: z_1, z_2, $z_3 = "quick"$, $q_1 = 1$, $z_4 = 2^{32}$, $z_5 > 99$. Данное множество не включает множество целевых параметров. Поэтому осуществляется переход к планированию на множестве логических операций.

В результате планирования действий на множестве логических операций, которые еще не выполнены, можно выполнить операции f_2^l и f_4^l. В результате их выполнения получаем $q_2 = 1$ и $q_4 = 1$.

Так как значения логических параметров q_3, q_5, q_6 не определены, то из числа не примененных продукций можно применить только продукции pr_2 и pr_4. В результате их применения будут выполнены вычислительные операции f_2 и f_4.

В результате их выполнения множество исходных параметров будет расширено. С учетом вычисленных значений логических параметров оно будет включать следующие параметры z_1, z_2, $z_3 = "quick"$, $q_1 = 1$, $z_4 = 2^{32}$, $z_5 > 99$, $q_2 = 1$, $q_4 = 1$, $z_6 > 0$, z_8, $q_5 = 1$. Данное множество не включает множество целевых параметров. Поэтому осуществляется переход к планированию на множестве логических операций.

В результате планирования действий на множестве логических операций, которые еще не выполнены, можно выполнить операцию f_3^l. В результате ее выполнения получаем $q_3 = 1$.

Так как значение логического параметра q_6 не определены, то из числа не примененных продукций можно применить только продукции pr_3 и pr_5. В результате их применения будут выполнены вычислительные операции f_3 и f_5.

В результате их выполнения множество исходных параметров будет

расширено. С учетом вычисленных значений логических параметров оно будет включать следующие параметры: z_1, z_2, $z_3 = "quick"$, $q_1 = 1$, $z_4 = 2^{32}$, $z_5 > 99$, $q_2 = 1$, $q_4 = 1$, $z_6 > 0$, z_8, $q_5 = 1$, $q_3 = 1$, z_7, z_9, $q_6 = 1$. Данное множество не включает множество целевых параметров. Поэтому осуществляется переход к планированию на множестве логических операций.

Все логические операции выполнены. Однако можно применить оставшуюся продукцию pr_6. В результате ее применения будет выполнена вычислительная операция f_6.

В результате ее выполнения множество исходных параметров будет расширено. С учетом вычисленных значений логических параметров оно будет включать следующие параметры: z_1, z_2, $z_3 = "quick"$, $q_1 = 1$, $z_4 = 2^{32}$, $z_5 > 99$, $q_2 = 1$, $q_4 = 1$, $z_6 > 0$, z_8, $q_5 = 1$, $q_3 = 1$, z_7, z_9, $q_6 = 1$, $z_{10} = 0$, $z_{11} = 1$, $z_{12} = 1$, $z_{13} = 0$, $z_{14} = 1$. Данное множество включает множество целевых параметров. Диагностика завершена. Работоспособность узла, модуля и агента, представляющего узел, а также наличие резервного узла и альтернативного модуля определены. В соответствии с таблицей решений (таблица 7) выбран 1-й сценарий.

Пример 2. Рассмотрим пример идентификации отказа вычислительного модуля поливариантной схемы решения задачи (рисунок 2.11). Воспользуемся определенными в примере 1 множествами параметров и определим следующие логические и вычислительные операции, а также продукции, определяющие условия выполнения вычислительных операций.

$f_1^l(z_3; q_1)$ – определение признака проведения быстрой диагностики;

$f_2^l(z_{15}; q_7)$ – определение признака превышения времени последнего успешного опроса состояния модуля;

$f_3^l(z_{16}; q_8)$ – определение признака работоспособности модуля схемы решения задачи;

q_7 – признак превышения времени последнего успешного опроса состояния модуля;

q_8 – признак работоспособности модуля;

q_9 – признак завершения работы модуля;

z_{15} – значение времени последнего успешного опроса состояния модуля;

z_{16} – идентификационный номер процесса аварийного модуля схемы решения задачи.

Для вычисления значений логических параметров q_1, q_7, q_8 в операциях $f_1^l - f_3^l$ определены следующие условия:

$$q_1 = \begin{cases} 1, \textit{если } z_3 = "quick", \\ 0 \textit{ в противном случае}, \end{cases}$$

$$q_7 = \begin{cases} 1, \textit{если } z_{15} > 5, \\ 0 \textit{ в противном случае}, \end{cases}$$

$$q_8 = \begin{cases} 0, \textit{если } q_7 = 1 \\ 1 \textit{ в противном случае.} \end{cases}$$

Множество вычислительных операций:

- $f_1(z_2; z_{15})$ – получение значения времени последнего успешного опроса состояния модуля;
- $f_2(z_1, z_2, z_{15}; z_{16})$ – идентификация неисправного модуля схемы решения задачи;
- $f_3(z_1, z_2, z_{15}; z_7)$ – отправка уведомления администратору РВС;
- $f_4(z_2, z_{16}; z_8)$ – отправка уведомления координатору виртуального сообщества, запустившего модуль z_2;
- $f_5(z_2, z_8; z_9, q_9 = 1)$ – остановка выполнения модуля z_2;
- $f_6(z_1, z_2, z_9; z_{10}, z_{11}, z_{12}, z_{13}, z_{14})$ – проверка статуса выполнения модуля z_2.

Множество продукций:

$$pr_1 : q_1 \Rightarrow f_1;$$

$$pr_2 : q_7 \Rightarrow f_2;$$

$$pr_3 : q_7 \Rightarrow f_3;$$

$$pr_4 : q_8 \Rightarrow f_4;$$

$$pr_5 : q_8 \Rightarrow f_5;$$

$$pr_6 : q_9 \Rightarrow f_6.$$

Пусть сформулирована следующая постановка задачи: «Зная $z_1, z_2, z_3 = "quick"$, вычислить z_{10}, z_{12}, z_{13}».

В результате планирования действий на множестве логических операций $f_1^l - f_3^l$ можно выполнить операцию f_1^l. В результате ее выполнения получаем $q_1 = 1$ что означает проведение быстрой диагностики.

Так как значения логических параметров $q_2 - q_9$ не определены, то можно применить только одну продукцию pr_1. В результате ее применения будет выполнена вычислительная операция f_1.

В результате ее выполнения множество исходных параметров будет расширено. С учетом вычисленных значений логических параметров оно будет включать следующие параметры: $z_1,\ z_2, z_3 = "quick",\ z_{15} > 5,\ q_1 = 1$. Данное множество не включает множество целевых параметров. Поэтому осуществляется переход к планированию на множестве логических операций.

В результате планирования действий на множестве логических операций, которые еще не выполнены, можно выполнить операцию f_2^l. В результате ее выполнения получаем $q_7 = 1$.

Так как значения логических параметров $q_2, q_3, q_4, q_5, q_6, q_8, q_9$ не определены, то из числа не примененных продукций можно применить только продукции pr_2 и pr_3. В результате ее применения будут выполнены вычислительные операции f_2 и f_3.

В результате их выполнения множество исходных параметров будет расширено. С учетом вычисленных значений логических параметров оно будет включать следующие параметры: $z_1,\ z_2, z_3 = "quick",\ z_7,\ z_{15} > 5,\ z_{16},\ q_1 = 1,\ q_7 = 1$.

Данное множество не включает множество целевых параметров. Поэтому осуществляется переход к планированию на множестве логических операций.

В результате планирования действий на множестве логических операций, которые еще не выполнены, можно выполнить операцию f_3^l. В результате ее выполнения получаем $q_8 = 0$.

Так как значения логических параметров $q_2, q_3, q_4, q_5, q_6, q_9$ не определены, то из числа не примененных продукций можно применить только продукции pr_4 и pr_5. В результате их применения будут выполнены вычислительные операции f_4 и f_5.

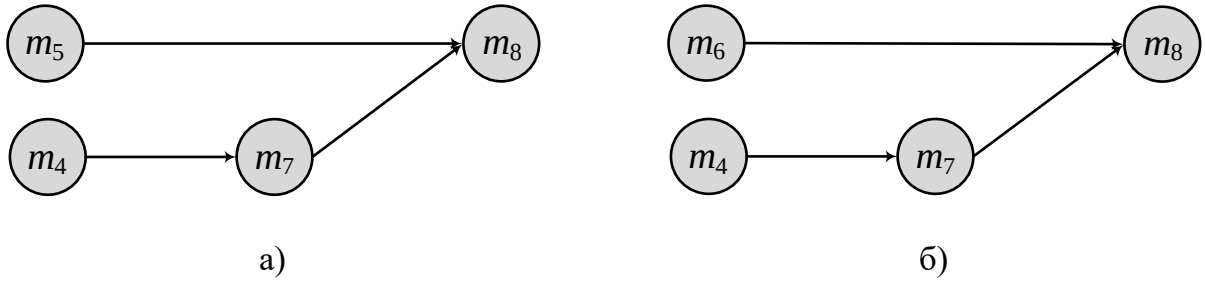
В результате их выполнения множество исходных параметров будет расширено. С учетом вычисленных значений логических параметров оно будет включать следующие параметры: z_1, z_2, $z_3 = "quick"$, z_7, z_8, z_9, $z_{15} > 5$, z_{16}, $q_1 = 1$, $q_7 = 1$, $q_8 = 1$, $q_9 = 1$. Данное множество не включает множество целевых параметров. Поэтому осуществляется переход к планированию на множестве логических операций.

Все логические операции выполнены. Однако можно применить оставшуюся продукцию pr_6. В результате ее применения будет выполнена вычислительная операция f_6.

В результате ее выполнения множество исходных параметров будет расширено. С учетом вычисленных значений логических параметров оно будет включать следующие параметры: z_1, z_2, $z_3 = "quick"$, z_7, z_8, z_9, $z_{10} = 1$, $z_{11} = 1$, $z_{12} = 0$, $z_{13} = 1$, $z_{14} = 1$, $z_{15} > 5$, z_{16}, $q_1 = 1$, $q_7 = 1$, $q_8 = 1$, $q_9 = 1$. Данное множество включает множество целевых параметров. Диагностика завершена. Работоспособность узла, модуля и агента, представляющего узел, а также наличие резервного узла и альтернативного модуля определены. В соответствии с таблицей решений (таблица 7) выбран 4-й сценарий.

Так как схема решения является поливариантной (рисунок 2.11), в результате обработки отказа выполнения модуля m_3 выполнится схема решения задачи с

альтернативным модулем m_6 вместо m_5 (рисунок 2.14).



а) б)

Рисунок 2.14. Фрагменты поливариантной схемы решения задачи: а) исходная схема, б) схема с альтернативным модулем

В примере 1 оценочное время на обнаружение сбоя во время диагностики в МАС составляет в среднем около 6 секунд. Condor DAGMan и Gridway затрачивают на устранение подобных отказов в среднем от 7 до 9 секунд при аналогичных с МАС настройках периодичности опроса состояния ресурсов.

В примере 2 только МАС сможет продолжить вычисления с запуском альтернативного модуля схемы решения задачи. Condor DAGMan и Gridway аварийно завершат выполнение задания. Оценочное время на обнаружение сбоя во время диагностики и перехода к новой схеме составляет в среднем около 8 секунд.

2.7. Методы и средства обучения агентов

Процесс обучения агентов строится на комплексном использовании методов концептуального моделирования, классификации заданий и параметрической настройки системы управления. В таблице 8 представлены агенты, а также методы, средства и субъекты их обучения.

Работа системы классификации заданий базируется на использовании концептуальной модели РВС и является ее дополнением. В системе классификации заданий, предложенной в [99], администраторы узлов на основе своих экспертных знаний определяет множество $H = \{h_1, h_2, ..., h_m\}$ возможных характеристик заданий (время решения задачи, размеры оперативной и дисковой памяти, число узлов, процессоров и ядер, режимы выполнения модулей и т.д.), а также области их допустимых значений.

Таблица 8 – Методы, средства и субъекты обучения агентов

Агент	Метод	Средство	Субъект
Агент формулировки постановки задачи и построения плана ее решения	Концептуальное моделирование предметной области Формулировка постановок задач и критериев эффективности их решения	Инструментальный комплекс Orlando Tools [31], расширение языка XML	Разработчик РППП Конечный пользователь РППП
Агент классификации заданий	Признаковое описание классов заданий	Система классификации заданий	Администраторы узлов РВС
Агент организации виртуального сообщества агентов	Спецификация соответствия классов заданий и ресурсов	Система классификации заданий	Администраторы узлов РВС
Агент, представляющий ресурсы РВС	Параметрическая настройка	Система имитационного моделирования РВС	Агент параметрической настройки
Агент параметрической настройки	Конфигурационная настройка Наблюдение за РВС	Система конфигурационной настройки МАС Система метамониторинга	Администратор РВС Агент мониторинга РВС
Агент мониторинга РВС	Конфигурационная настройка	Система конфигурационной настройки МАС	Администратор РВС
Агент диспетчеризации заданий в невыделенных ресурсах	Спецификация соответствия классов заданий и ресурсов	Система классификации заданий	Администратор РВС

Затем они формируют классы $C = \{c_1, c_2, ..., c_n\}$ заданий, обладающих наборами характеристик из определенного множества. Область допустимых значений характеристики при ее включении в тот или иной класс может быть конкретизирована.

Сформированным классам сопоставляются наиболее подходящие ресурсы для выполнения заданий, относящихся к этим классам (рисунок 2.15).

Администратор РВС задает значения параметров, определяющих намерения ресурсных агентов по выполнению заданий разных классов, нижние и верхние границы допустимого отклонения от средней нагрузки ресурсов агентов виртуальных сообществ, величины штрафов за отклонение от средней нагрузки (для агента параметрической настройки); состав и периодичность сбора информации, форматы данных, используемые системы мониторинга и контрольно-измерительные приборы, границы изменения измеряемых величин, а также применяемые при их достижении управляющие воздействия (для агента мониторинга); допустимые квоты по числу задач, времени их выполнения и числу используемых узлов, а также характеристики окон в расписании системы PBS Torque в невыделенных ресурсах (для агента диспетчеризации заданий).

Рисунок 2.15 – Схема формирования классов заданий и сопоставления им ресурсов

Для поддержки адаптивного обучения агентов разработана новая технология параметрической настройки мультиагентных алгоритмов управления РВС с целью оптимизации распределения ее ресурсов при выполнении заданий приложений пользователей. Схема параметрической настройки алгоритмов работы агентов распределения ресурсов представлена на рисунке 2.16. Агент-классификатор

идентифицирует классы заданий, на основе которых формируется виртуальное сообщество ресурсных агентов. Эти агенты распределяют ресурсы на основе тендера вычислительных работ, и алгоритмы их работы определяются вектором управляющих параметров, обеспечивающих агентам выбор оптимальной стратегии поведения. Эти управляющие параметры регулируются агентом параметрической настройки и отражают значения вектора варьируемых входных переменных имитационной модели РВС, соответствующих оптимальным значениям вектора ее наблюдаемых переменных, вычисляемых на основе проведения многовариантных расчетов. Нахождение значений управляющих параметров производится с помощью многокритериальных правил дискретного выбора [100].

Рисунок 2.16 – Схема параметрической настройки

Агент мониторинга реализован в системе мета-мониторинга [86]. Он предназначен для обеспечения субъектов РВС актуальной информацией о загрузке ее ресурсов, физическом состоянии вычислительного оборудования и устройств инженерной инфраструктуры. Важной особенностью применяемых агентов мониторинга в отличие от других систем является способность агентов анализировать и принимать управляющие воздействия непосредственно на вычислительном узле, на котором функционирует агент. Собираемые данные унифицируются, агрегируются и передаются экспертной системе для их анализа. В случае выявления критических событий выполняются необходимые функции исполнительной системы с целью применения управляющих воздействий для автоматического устранения неисправностей. При этом администратором может производиться предварительное обучение агентов мониторинга с учетом предназначения вычислительных узлов и выполняемых на них задач.

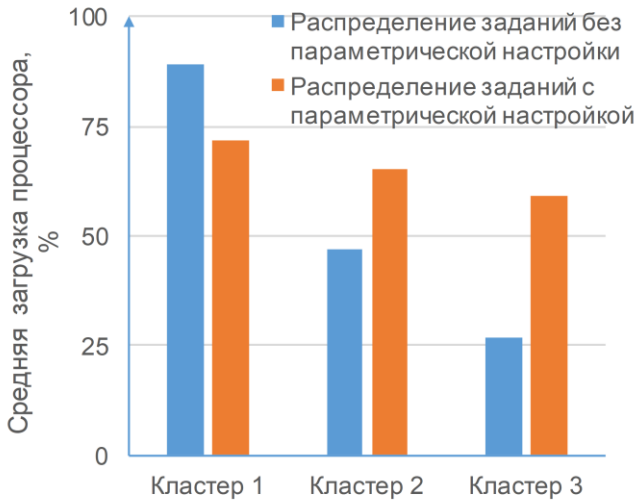


Рисунок 2.17 – Балансировка загрузки

Оценка действий ресурсных агентов регулируется системой штрафов отклонения от средней нагрузки ресурсов в их виртуальном сообществе. На основе анализа результатов распределения ресурсов агенты могут изменять намерения по выполнению заданий различных классов. Рисунок 2.17 демонстрирует результаты моделирования средней загрузки процессоров ресурсов потоком заданий,

выполненным в ЦКП ИСКЦ в период с 01.07.2018 по 31.10.2018. Эти результаты показывают, что параметрическая настройка агентов может существенно улучшить балансировку загрузки процессоров с учетом заданных критериев эффективности заданий.

В системе обучения агентов используется новая гибридная модель извлечения и применения проблемно-ориентированных знаний агентами в процессе управления потоками заданий. Разработчики приложений осуществляют концептуальное моделирование проблемной области и описывают специфику решаемых задач. Администраторы формируют знания путем добавления описаний системных объектов РВС и правил их работы в концептуальную модель, а также посредством определения конфигурационных настроек отдельных агентов и мультиагентной системы в целом. Знания разработчиков приложений и администраторов образуют базу знаний РВС, на основе которой создаются локальные базы знаний приложений и агентов, функционирующих в рамках РВС на основе локальных взаимодействий между собой.

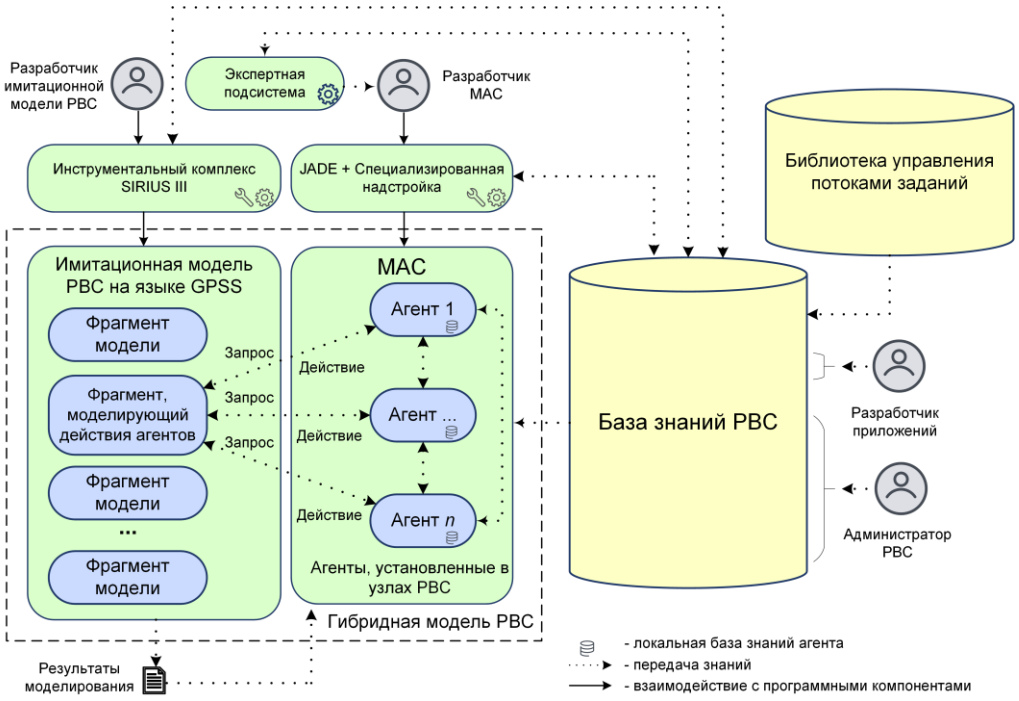


Рисунок 2.18 – Схема передачи знаний

На рисунке 2.18 приведена схема передачи знаний между субъектами,

генерирующими эти знания, агентами и программными комплексами.

Локальные базы знаний агентов формируются разработчиками МАС. В случае создания агентов на уровне приложения (агентов приложения или пользовательских агентов) его разработчик может выступать и в качестве разработчика МАС. Как правило, администратор разрабатывает агентов для управления вычислениями на уровне РВС. Специальные агенты извлекают дополнительные знания путем распознавания свойств приложений и их заданий, моделирования вычислительных процессов и прогнозирования интенсивности поступления и времени выполнения заданий. Дополнительные знания передаются агентам распределения ресурсов для выполнения заданий, генерируемых приложениями в РВС, с целью повышения эффективности управления вычислениями. Новая гибридная модель РВС организуется путем включения в имитационную модель средств взаимодействия с реальными агентами, размещенными в узлах РВС, что существенно повышает степень достоверности результатов моделирования по сравнению с отдельно взятым имитационным моделированием. Модели поведения реальных агентов строятся в процессе их разработки с экспертной поддержкой этого процесса, реализованной в рамках специализированной подсистемы.

2.8. Выводы

В данной главе рассмотрены вопросы, связанные с разработкой мультиагентных моделей, методов, алгоритмов и инструментальных средств управления РВС. В ней получены следующие основные результаты:

- представлена структура МАС для управления потоками заданий;
- формализована концептуальная модель такой системы;
- для данной модели определено понятие остаточной схемы решения задачи и формализован процесс ее построения;
- разработана новая автоматная модель поведения агентов;
- для данной модели разработан набор методов концептуального

моделирования предметной области функционирования агентов, динамического планирования их действий в разнородной РВС и автоматного программирования;

- разработан оригинальный мультиагентный алгоритм для построения остаточной схемы решения задачи и перераспределения ресурсов РВС в случае отказа ее программно-аппаратных средств;
- создана система машинного обучения агентов, базирующаяся на использовании гибридной модели представления и применения знаний.

Содержание данной главы отражено в публикациях [7, 8, 12-14, 16-19, 20–29, 34, 36].

Глава 3. Инструментальный комплекс

В третьей главе рассмотрена платформа JADE, используемая в рамках диссертации в качестве базового набора инструментов для построения агентной среды, и предложена надстройка над этой платформой, позволяющая автоматизировать процесс конструирования агентов. Представлена методика применения инструментального комплекса организации мультиагентного управления в РВС, организованной с использованием ресурсов ЦКП ИСКЦ и Иркутского научного центра СО РАН. В заключение сформулированы выводы по результатам исследований, представленных в данной главе.

3.1. Агентная платформа JADE

JADE представляет собой платформу для разработки мультиагентных систем. Она поддерживает FIPA-стандарты. Архитектура FIPA представлена на рисунке. 3.1. Платформа JADE может функционировать на распределенных вычислительных узлах, работающих под управлением различных ОС при наличии установленной системы Java не ниже 5-й версии.

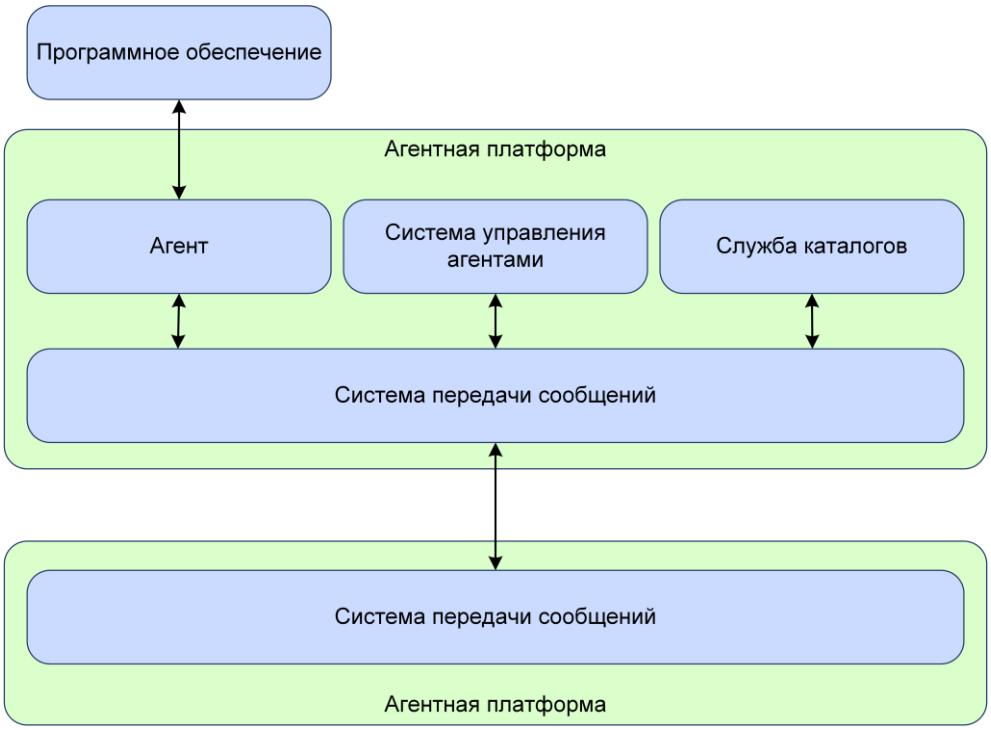


Рисунок 3.1 – Архитектура FIPA

JADE представляет собой ПО промежуточного слоя (рисунок 3.2), разработанное компанией Telecom Italia, предназначенное для создания распределенных мультиагентных приложений на основе транспортной архитектуры «точка-точка». Среда может динамично взаимодействовать с узлами, которые в терминологии JADE называются агентами. Агенты появляются и исчезают в системе в соответствии с потребностями и требованиями программной среды. Коммуникации между узлами не зависят от типа сети (проводная, беспроводная). Они являются полностью симметричными, и каждый узел может, как инициировать запросы, так и отвечать на них. Стандартные агенты создаются автоматически при запуске агентной платформы.

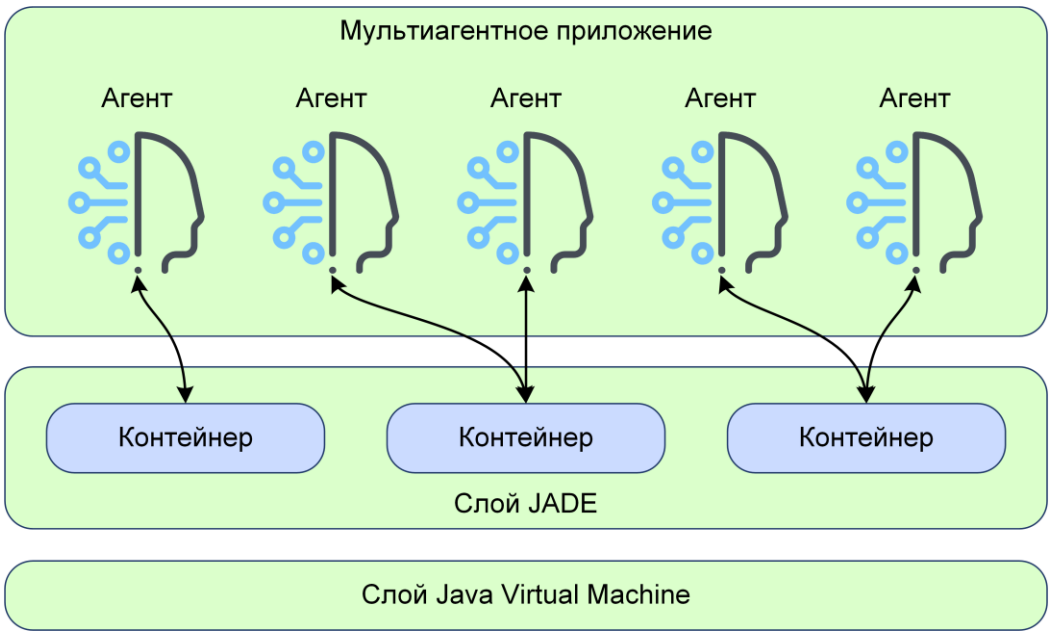


Рисунок 3.2 – Архитектура JADE

Платформа JADE состоит из следующих компонентов:

– среды выполнения агентов, которые регистрируются и работают под ее управлением;
– набора стандартных агентов: Agent Management System, Directory Facilitator и Remote Monitoring Agent;
– библиотек стандартных классов, которые используются для разработки агентных систем;

– набора графических утилит для администрирования и наблюдения за жизнедеятельностью активных агентов.

JADE полностью написан на языке программирования Java, что позволяет подключать его к любому проекту на этом языке и как следствие – является кроссплатформенным, может работать на беспроводных мобильных устройствах, и распространяется бесплатно по лицензии GNU Lesser General Public Licens (LGPL). JADE разработан в соответствии со спецификациями FIPA, благодаря чему агенты JADE могут обмениваться сообщениями с любыми другими агентами, поддерживающими данный стандарт.

С помощью среды исполнения агентов пользователю предоставляются базовые службы платформы. Данные службы должны быть запущены на устройстве до того, как на нем будет исполняться агент. Запущенный экземпляр JADE называется контейнером, так как он «содержит» агентов. Данного термина нет в стандарте FIPA. Он является собственной абстракцией JADE, введенной разработчиками, и удобным средством группировки агентов в платформе. Набор всех контейнеров называется платформой и представляет однородный слой, который скрывает от агентов (а также от разработчиков приложений) сложность и распределенный характер механизмов, расположенных на более низком уровне (аппаратное обеспечение, ОС, типы сетей, виртуальную машину Java).

Конструирование агентов МАС в JADE является весьма трудоемкой, рутинной работой, требующей высокой программистской квалификации и погружения во все тонкости представления и использования предметно-ориентированных знаний агентом. JADE предоставляет набор инструментов для управления жизненным циклом агентов и обеспечения базовых функций по взаимодействию с платформой. Взаимодействие агентов между собой и платформой основывается на обмене сообщений, формат которых регламентирован FIPA. К сожалению, платформой не гарантируется надежность и правильная очередность доставки сообщений.

3.2. Реализация автоматной модели поведения агентов с помощью встроенного класса FSMBehaviour

Разработка МАС основывается на автоматизированной реализации моделей поведения агентов. Такие модели должны быть гибкими и адаптируемыми к изменяющимся условиям вычислительной среды, предпочтениям владельцев ресурсов и критериям решения задач, указываемым пользователями. Качество реализации моделей повышается за счет сокращения накладных расходов на выявление и устранение ошибок в поведении агентов в процессе ее автоматизации.

Специальный класс FSMBehaviour обеспечивает в JADE поддержку разработки автоматной модели поведения агента. На его основе создаются два его экземпляра Parrent.FSMBehaviour и Child.FSMBehaviour, представляющие соответственно родительский и дочерний автоматы. Для реализации каждого из автоматов выполняется следующая совокупность действий:

- системный модуль концептуальной модели, реализующий поведение агента в его начальном состоянии для соответствующего автомата, регистрируется с помощью метода registerFirstState;
- системные модули концептуальной модели, реализующие поведение агента в его конечных состояниях для соответствующего автомата, регистрируются с помощью метода registerLastState;
- системные модули концептуальной модели, реализующие поведение агента в его промежуточных состояниях для соответствующего автомата, регистрируются с помощью метода registerState;
- правила переходов между состояниями регистрируются с помощью метода registerTransition;
- правила переходов между состояниями по умолчанию регистрируются с помощью метода registerDefaultTransition.

Выбор модулей, реализующих поведение агента в различных состояниях, и условий переходов между состояниями осуществляется с помощью специальных форм для работы с концептуальной моделью (рисунок 3.3). В данной форме

представлены начальные и конечные состояния (First State и Last State соответственно), а также промежуточные (State). Каждому состоянию присваивается номер (сквозная нумерация). Из выпадающего списка можно выбрать модули концептуальной модели, а в поле «State Trans. Condition» сформировать условие перехода в следующее состояние (поле «Next state»).

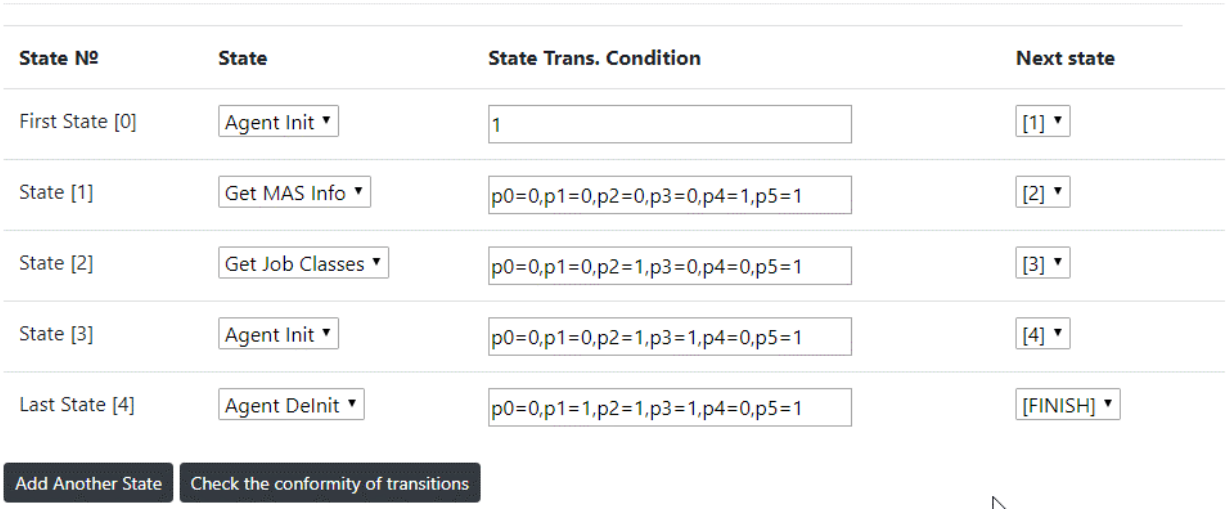


Рисунок 3.3 – Графический интерфейс надстройки над JADE для формирования поведения агента

По нажатию кнопки «Add Another State» добавляется состояние после текущего, а по нажатию кнопки «Add Another Last State» добавляется очередное конечное состояние. Программный код, реализующий автоматы, генерируется автоматически после проверки согласованности переходов (кнопка «Check the conformity of transitions»).

Известны различные надстройки для создания автоматов в JADE, такие как JADE Finite State Machine Engine [101] и Hierarchical State Machine Editor [102]. Однако программный код автоматов в рамках этих инструментов создается, как правило, в «ручном» режиме.

Так как в рамках диссертационного исследования состояния действия агентов реализуются с помощью заранее подготовленных системных модулей, то применение вышеупомянутых надстроек существенных преимуществ не предоставляет.

3.3. Инструментальный комплекс автоматизации разработки мультиагентной системы

Для решения проблем конструирования агентов, описанных в п. 3.1, в диссертации предложены инструментальные средства, позволяющие автоматизировать процесс разработки МАС, начиная от функционального наполнения агентов и заканчивая управлением жизненным циклом агентов. Методика организации МАС основывается на восходящем подходе к организационному проектированию подобных систем. JADE предоставляет базовый инструментарий, поэтому достаточно распространена практика разработки надстроек над агентными платформами, расширяющими их возможности [101-103].

Так как для реализации МАС зачастую нужны типовые системные функции поведения агентов, которые приходится реализовывать для каждого нового агента, в рамках диссертации разработана интеллектная надстройка над JADE, позволяющая частично автоматизировать и тем самым существенно упростить процесс конструирования агентов МАС на базе стандартных классов JADE. Данная надстройка включает два основных компонента: конструктор моделей поведения агентов и генератор их программного кода.

Предлагается подход к автоматизации конструирования агента МАС на базе стандартных классов JADE, позволяющий существенно упростить этот процесс. Данный подход реализуется следующим образом: с помощью разрабатываемого инструментария для организации проблемно-ориентированной МАС формируются множества состояний агента, условий (функций) переходов состояний агента и сообщений агента. На основе этих множеств строится граф переходов состояний агента. Затем создается база знаний агента путем конкретизации агрегированной модели РВС с учетом роли этого агента в МАС. Граф переходов состояний агента, база знаний, библиотека стандартных классов JADE и библиотека оригинальных алгоритмов функционирования агентов используются для синтеза абстрактной программы, специфицирующей поведение агента, а также для генерации

программного кода агента на языке Java.

Представленный в диссертации инструментарий для организации проблемно-ориентированных МАС является надстройкой над средствами системы JADE, существенно расширяющей возможности этой системы. Для реализации дополнительных методов стандартных классов JADE, представляющих функции конструируемых агентов МАС, используется библиотека «встроенных» оригинальных алгоритмов функционирования агентов. Алгоритмы адаптивного поведения агентов базируются на использовании ролевой модели и различных сценариев действий агентов в зависимости от текущего состояния РВС. Алгоритмы извлечения и применения проблемно-ориентированных знаний агентами реализуются на основе методов конкретизирующего программирования, а также с помощью экономических механизмов в процессе планирования вычислений и распределения ресурсов.

Эффективность распределенных алгоритмов взаимодействия агентов (например, алгоритмов выбора агента-координатора и проведения тендера) обеспечивается путем снижения временных затрат на обмен сообщениями за счет устранения избыточности известных алгоритмов аналогичного назначения, используемых на практике. При этом надежность таких алгоритмов взаимодействия агентов обеспечивается за счет применения систем логического времени и контрольных точек.

Основные особенности предложенной методологии заключаются в комплексном применении методов концептуального программирования, самоорганизации, адаптивного обучения и имитационного моделирования в процессе организации МАС. Рассматриваемые инструментальные средства обеспечивают, в отличие от известных инструментариев проблемную ориентацию и корректность функционирования агентов, а также частичную автоматизацию их конструирования.

Разработана библиотека алгоритмов функционирования агентов, реализующая основные поведенческие функции агентов: формулировка

постановок задач, планирование вычислений, распределение ресурсов, обеспечение отказоустойчивости вычислительных процессов. Предложена надстройка над JADE, которая генерирует агентов МАС на основе базовых библиотек JADE, специальных библиотек поведения агентов, агрегированной модели РВС и прикладных модулей решаемой задачи.

Данная надстройка снижает уровень требований, предъявляемых разработчикам МАС для решения прикладных программ, и позволяет достаточно быстро получить первый прототип агента, сокращая временные затраты пользователей-предметников МАС.

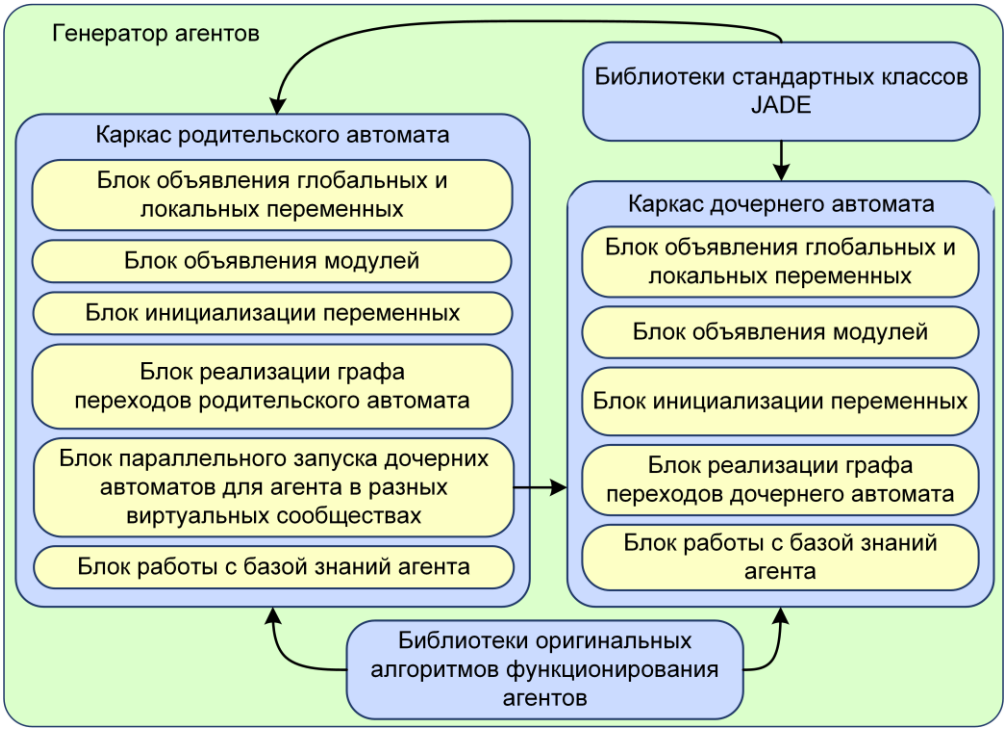


Рисунок 3.4 – Генерация программного кода агента

С помощью генератора агентов (рисунок. 3.4) разработчик МАС производит синтез абстрактной программы, специфицирующей поведение агента. Ее построение производится путем статического (на основе процедурной и непроцедурной постановок задач) или динамического планирования вычислений. Далее разработчик выполняет формирование программного кода агента на языке Java. Генератор использует граф переходов состояний агента, базу знаний,

библиотеку стандартных классов JADE и библиотеку оригинальных алгоритмов функционирования агентов. Генерация программного кода агента осуществляется в рамках каркасного подхода к конструированию программ.

Каркас родительского автомата включает:

- блок объявления глобальных и локальных переменных;
- блок объявления модулей;
- блок инициализации переменных;
- блок реализации графа переходов родительского автомата;
- блок параллельного запуска дочерних автоматов для агента в разных виртуальных сообществах;
- блок работы с базой знаний агента.

Каркас дочернего автомата включает:

- блок объявления глобальных и локальных переменных;
- блок объявления модулей;
- блок инициализации переменных;
- блок реализации графа переходов дочернего автомата;
- блок работы с базой знаний агента.

JADE предоставляет набор стандартных библиотек, обеспечивающих функционирование агентной платформы, в том числе обмен сообщениями. С помощью данных библиотек обеспечиваются базовые функции управления жизненным циклом агентов и взаимодействия с платформой. За наполнение агента поведением отвечает разработчик МАС.

Для реализации дополнительных методов стандартных классов JADE, представляющих функции (операции) агентов МАС, применяется библиотека «встроенных» оригинальных алгоритмов, оформленных в виде модулей, а также подключаются исполняемые модули пользователей (рисунок 3.5). Библиотека алгоритмов функционирования агентов включает следующие алгоритмы:

- формулировки постановок задач;

– планирования вычислений и распределения ресурсов;
– обеспечения отказоустойчивости вычислительных процессов;
– коммуникационного взаимодействия.

Рисунок 3.5 – Надстройка над JADE

Каждому агенту назначается модель поведения, связанная с имеющимися алгоритмами, а также передается полная информация о модели РВС. Сгенерированный таким образом агент в результате сборки и компиляции автоматически размещается на выбранном узле и подключается к МАС.

Гибкая модернизация алгоритмов функционирования агентов обеспечивается возможностью подключения «внешних» библиотек, включающих соответствующие алгоритмы, разработанные пользователями МАС (например, администраторами РВС). Процесс подключения «внешних» библиотек и разработки их алгоритмов определяется специальными интерфейсами и спецификациями.

При каркасном подходе (рисунок 3.6) каждый агент состоит из двух обязательных частей: каркаса и гнезд. Каркас представляет собой неизменяемую часть программы и включает в себя совокупность изменяемых частей – гнезд, в которые помещаются сменные модули пользователей.

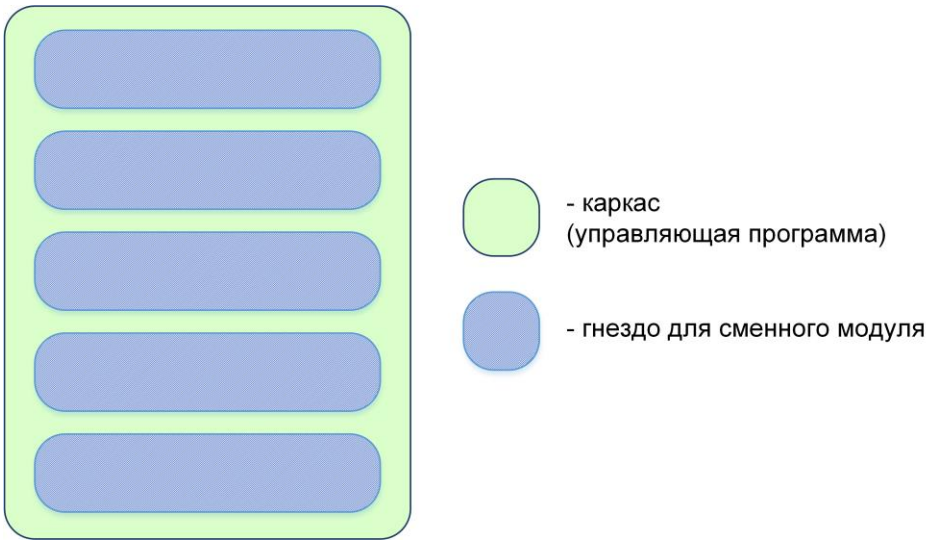


Рисунок 3.6 – Каркасный подход

3.4. Методика применения инструментального комплекса

Методика применения представленного инструментального комплекса для создания предметно-ориентированной МАС включает следующие основные этапы:

- установка и запуск агентной платформы JADE;
- настройка конфигурации МАС;
- разработка моделей поведения агентов;
- генерация программного кода агентов;
- размещение агентов и их подключение к JADE;
- подключение агентов к выбранному инструментальному комплексу (Orlando Tools).

Выполнение перечисленных этапов сопровождается экспертной поддержкой принятия решений. В процессе своей работы агенты пополняют свои знания, полученные в процессе своего конструирования, посредством использования системы классификации заданий и приложений, имитационного моделирования

вычислительных процессов, параметрической настройки управляющих параметров алгоритмов функционирования агентов.

3.4.1. Установка и запуск агентной платформы JADE

Установка и запуск платформы JADE и процесс подключения к ней агентов детально рассматривается в Приложении Д. JADE разворачивается на управляющем узле системы и находится в запущенном состоянии для обеспечения запуска пользовательских агентов.

Как правило, эти действия выполнять пользователю необходимо только во время отладки разрабатываемых агентов, в остальных случаях команды выполняются в автоматическом режиме средством размещения агентов инструментального комплекса.

3.4.2. Настройка конфигурации мультиагентной системы

Разработчик МАС осуществляет:

– выбор необходимых ролей агентов из числа доступных;
– подключение внешних библиотек алгоритмов функционирования агентов;
– выбор инструментального комплекса для интеграции с разрабатываемой МАС и задание необходимых конфигурационных параметров для реализации такой интеграции.

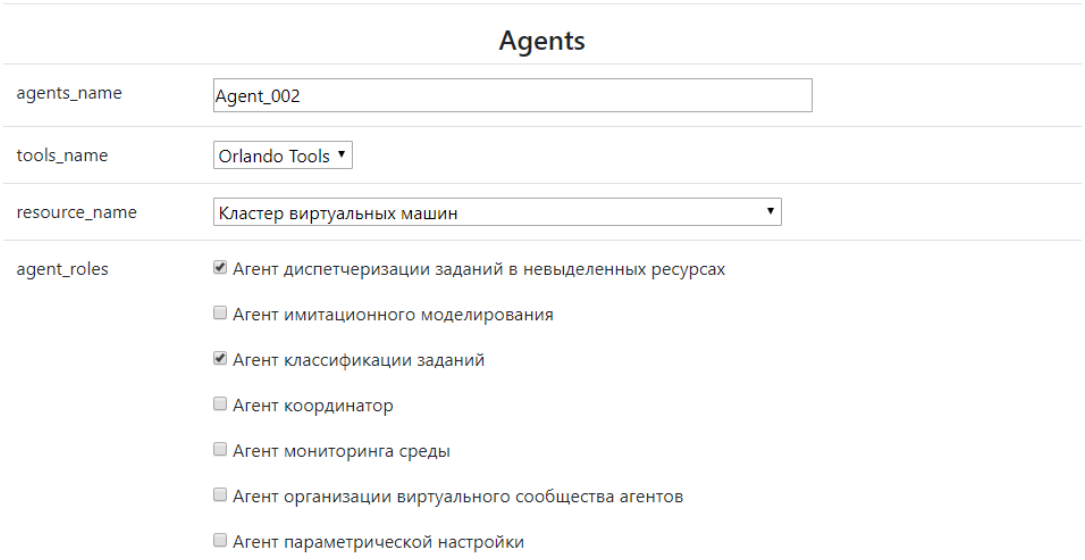


Рисунок 3.7 – Выбор ролей агента

С помощью формы создания нового агента (рисунок 3.7) назначается имя агента (поле agents_name), из выпадающего списка (поле tools_name) выбирается инструментальный комплекс, используемый агентом, а также из выпадающего списка с доступными для размещения вычислительными кластерами (поле resource_name) выбирается нужный ресурс.

В экспериментальной РВС доступны следующие вычислительные ресурсы:

– вычислительный кластер «Академик В.М. Матросов» (сегмент В);
– вычислительный кластер «Академик В.М. Матросов» (сегмент А);
– кластер виртуальных машин;
– кластер Иркутского государственного университета;
– кластер персональных компьютеров;
– пул выделенных ресурсов.

Каждому агенту допустимо назначение более одной роли, поэтому на форме создания агента предусмотрен список доступных ролей агента (поле agent_roles), позволяющий отметить галочкой необходимые роли.

Для каждого используемого инструментального комплекса предусмотрен редактор (рисунок 3.8), позволяющий назначить каждому из них пути до всех внешних библиотек, к которым будет открыт доступ агентам.

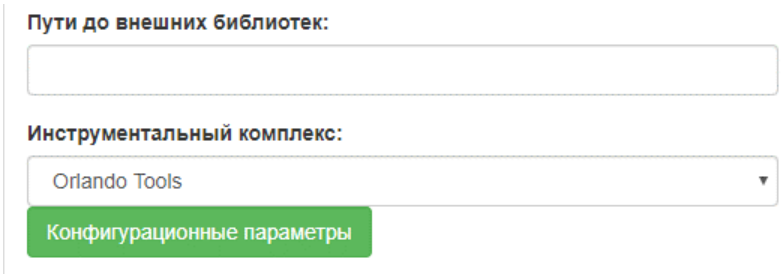


Рисунок 3.8 – Редактор инструментальных комплексов

Разработчик имеет возможность выбора типовых конфигураций МАС, сформированных для различных архитектур РВС на основе экспертного опыта ее администраторов.

3.4.3. Конструирование моделей поведения агентов

Функционирование агентов осуществляется с использованием парадигмы конечно-автоматного программирования. С помощью разработанного инструментального комплекса разработчик МАС выполняет концептуальное моделирование. В отличие от вычислительных моделей подобного назначения концептуальная модель РВС позволяет осуществить взаимосвязанное описание не только алгоритмических знаний предметных областей решаемых задач, но и знаний о программно-аппаратной инфраструктуре и административных политиках, определенных для ресурсов. В модель включены следующие компоненты знаний:

- вычислительные знания, содержащие информацию о прикладных модулях для решения задач, системных модулях для планирования вычислений, формирования заданий, распределения ресурсов, мониторинга вычислительных процессов и динамической декомпозиции задач, а также о вспомогательных модулях для препроцессорной и постпроцессорной обработки данных;
- схемные знания, включающие множество объектов (например, параметров и операций) для описания модульной структуры модели и алгоритмов исследования проблемной области;
- продукционные знания, определяющие правила применения операций и позволяющие конечным пользователям приложений выбирать лучшие алгоритмы в текущей вычислительной ситуации;
- инфраструктурные знания, представленные характеристиками программно-аппаратных объектов – узлов, каналов связи, сетевых устройств, топологии сети и других структурных элементов, а также сведениями об их надежности;
- административные знания о политиках в отношении ресурсов и их пользователей, включающие правила использования ресурсов, права и квоты для пользователей и их заданий, а также информацию о системах управления заданиями и дисциплинах их обслуживания.

С помощью редактора (рисунок 3.9) формируются новые состояния-действия агентов, где каждое состояние – вершина для графа переходов агента. Задается имя состояния (state_name), путь до файла описания функции переходов (state_function_path), условие перехода в данное состояние (значение функции 0 или 1, function_value) и путь до базы данных МАС.

State Editor

select	id	state_name	state_fucntion_path	fucntion_value	db_path
○	1	AS20	./mas/as20.json	0	db_mas
○	2	AS11	./mas/as11.json	1	db_mas
○	5	AS22	./mas/as25.json	0	db_mas

Рисунок 3.9 – Редактор состояний-действий агента

Особенностью концептуальной модели РВС является возможность использования системных объектов для описания схемных знаний о предметной области функционирования агентов. Разработчик МАС определяет и описывает следующие объекты: агенты, их роли, виртуальные сообщества и отношения между ними, а также состояния, функции и графы переходов автоматов. В качестве состояния агента используется состояние-действие – последовательность системных операций, выполняемых над полем системных параметров модели.

Затем разработчик автоматически конвертирует описание модели на XML, который используется в качестве языка описания вычислительной модели РВС в инструментальном комплексе Orlando Tools, предназначенном для разработки РППП. МАС интегрирована с системами управления данного инструментального комплекса. Таким образом, МАС и пакеты используют единую модель РВС.

3.4.4. Генерация программного кода агентов

Типовые роли агентов представлены в виде описания в xml-файле. Пользователь при конструировании агента либо использует существующие модели поведения агентов, либо загружает свои собственные. Редактор ролей агентов представлен на рисунке 3.10.

Все созданные роли становятся доступными в форме создания агентов (см. раздел 3.4.2). Для каждой роли необходимо задать имя роли (поле role_name), прикрепить xml-файл с описанием модели поведения (поле role_beh_model_path), указать используемую базу данных (поле role_tool_db), указать (при наличии) пути доступа к API взаимодействия с агентом, выполняющим функции создаваемой роли. Для указания пути к библиотекам, реализующим поведение агента в рамках создаваемой роли, используется поле role_func_lib_path.

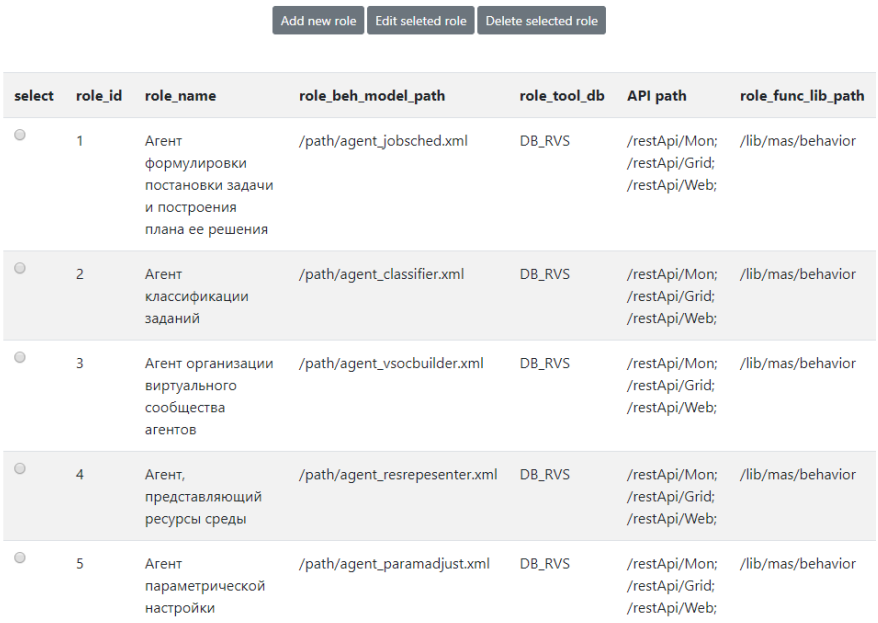


Рисунок 3.10 – Редактор ролей агентов

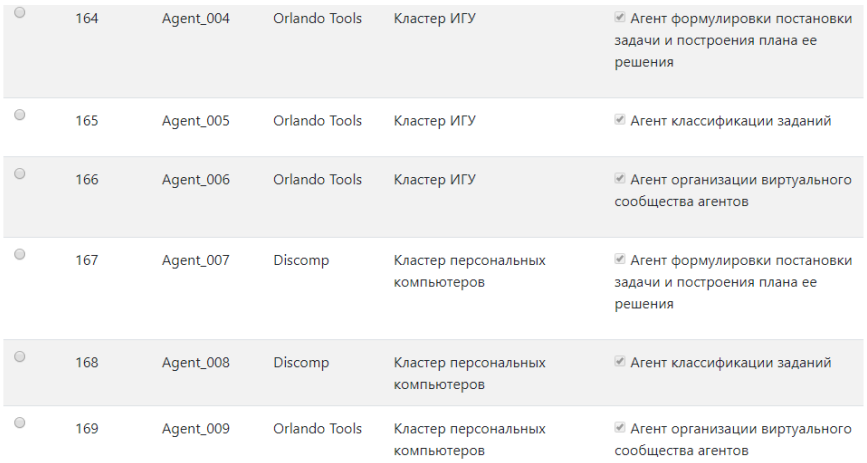


Рисунок 3.11 – Перечень агентов в редакторе

После описания необходимых ролей (моделей поведения) агентов

необходимо создать агентов и назначить им роли (см. раздел 3.4.2). Пример списка созданных агентов, назначенных им имен, выбранных инструментальных комплексов, ресурсов и ролей представлен на рисунке 3.11.

Следующим этапом является непосредственная генерация программного кода конструируемых агентов. На странице итоговой сборки новых агентов перечислены все доступные пользовательские агенты (рисунок 3.12). Для каждого агента доступна кнопка «Build», которая запускает генерацию кода нового агента на сервере. В зависимости от результата (Compiled, зеленый цвет – успешно; Error, красный цвет – произошла ошибка) становится доступной либо кнопка скачивания class-файла агента (Download Agent_001.class), либо кнопка скачивания вывода с ошибкой (Download out.log).

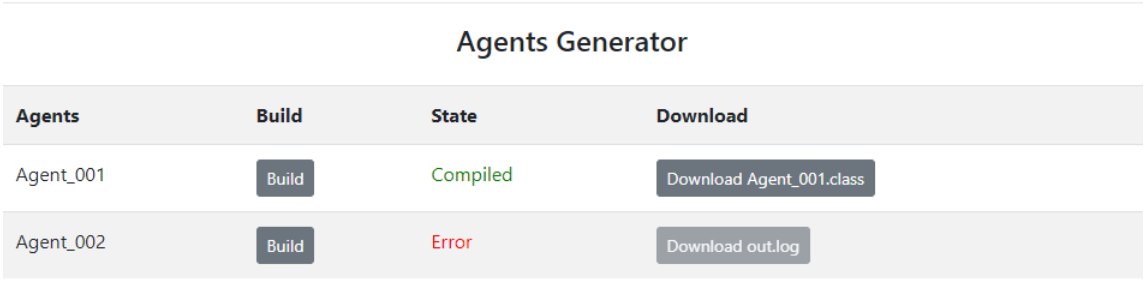


Рисунок 3.12 – Генерация программного кода конструируемых агентов

3.4.5. Размещение агентов и их подключение к JADE

После получения исходного кода агента его необходимо разместить на участвующих в вычислениях вычислительных узлах. Для авторизированных пользователей исходный код агентов автоматически сохраняется в домашнем каталоге сетевого хранилища. Актуальные версии библиотек JADE также расположены в домашнем каталоге в подпапке lib. Для подключения новых агентов к запущенной платформе JADE удобно использовать контейнеры JADE.

Агент после регистрации в платформе готов принимать и передавать сообщения и в зависимости от сообщений выполнять действия в соответствии с определенными ролями. При компиляции исходного кода агента из конфигурационного файла загружаются сведения о подключаемых инструментальных комплексах.

В веб-интерфейсе системы после завершения конструирования агентов появляется возможность проектирования новой МАС (рисунок 3.13). В данной форме задается имя новой МАС (поле mas_name), IP-адрес узла, на котором будет запущена управляющая платформа (mas_ip), порт, на котором создаваемая МАС будет принимать подключения новых агентов. В поле include_next_agents перечислены агенты, которых необходимо включить в создаваемую МАС.

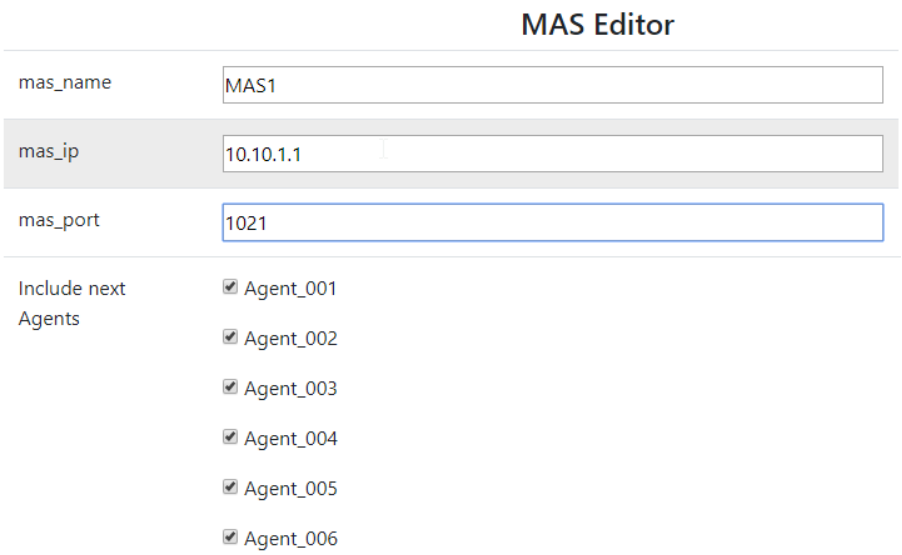


Рисунок 3.13 – Проектирование новой МАС

Для каждой создаваемой МАС происходит автоматическая проверка обеспечения минимально необходимых функциональных возможностей выбранных агентов для функционирования МАС (рисунок 3.14).

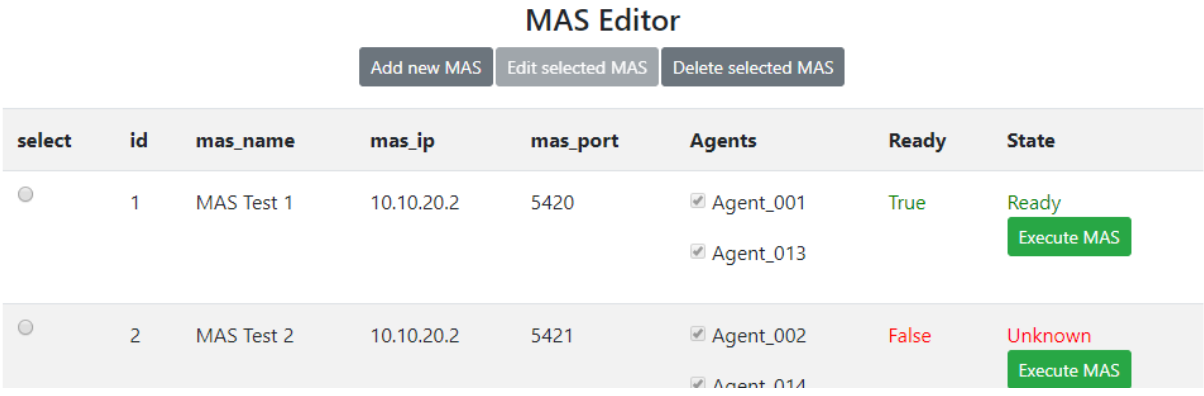


Рисунок 3.14 – Выполнение МАС

При удовлетворении условий работы MAC в столбце Ready отобразится статус True, подсвеченный зеленым цветом. В противном случае появится статус False, подсвеченный красным цветом. В столбце State отображается текущий статус MAC и доступна кнопка для запуска платформы создаваемой MAC (Execute MAS). Статус Ready говорит о том, что MAC готова к работе, статус Unknown сообщает о неопределенном состоянии системы.

3.4.6. Подключение агентов к инструментальному комплексу разработки распределенных пакетов прикладных программ

Следующим дополнительным шагом по конфигурации агентов является их подключение к инструментальному комплексу Orlando Tools. В веб-интерфейсе Orlando Tools происходит назначение агентам функций компонентов исполнительной подсистемы Orlando Tools. В примере, представленном на рисунке 3.15, переназначаются функции системы планирования агенту диспетчирования заданий в невыделенных ресурсах, а в качестве интерпретатора назначается агент, представляющий ресурсы РВС. С помощью инструментального комплекса Orlando Tools описывается модель предметной области, параметры и операции, а также модули, их реализующие (рисунок 3.16).

N	Компонент исполнительной подсистемы Orlando	Агент	Ресурс
1	Планировщик	Агент диспетчирования заданий в невыделенных ресурсах	☐ Пул выделенных ресурсов
			☑ Кластер персональных компьютеров
			☐ Кластер ИГУ
			☑ Кластер виртуальных машин
			☐ Вычислительный кластер «Академик В.М. Матросов» (сегмент В)
			☐ Вычислительный кластер «Академик В.М. Матросов» (сегмент А)
2	Интерпретатор	Агент, представляющий ресурсы среды	☑ Пул выделенных ресурсов
			☐ Кластер персональных компьютеров
			☐ Кластер ИГУ
			☐ Кластер виртуальных машин
			☑ Вычислительный кластер «Академик В.М. Матросов» (сегмент В)

Рисунок 3.15 – Переназначение функций инструментального комплекса

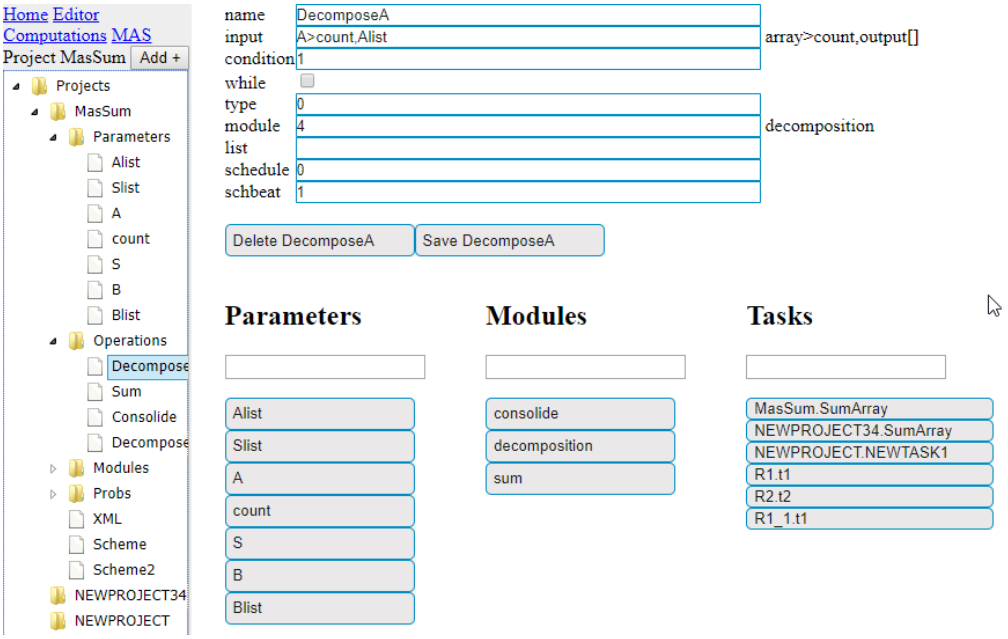


Рисунок 3.16 – Описание РППП в Orlando Tools

Более детально описание принципов построения РППП представлено в работе [25]. После формирования постановки задачи с помощью средства отправки готовых постановок задается имя нового задания, выбирается задача и предпочитаемая МАС (рисунок 3.17).

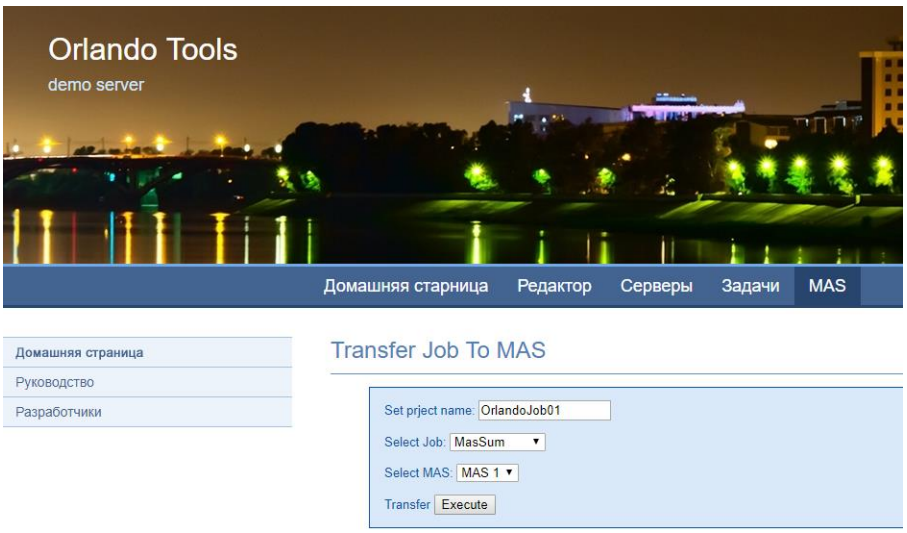


Рисунок 3.17 – Отправка заданий из Orlando Tools в МАС

3.4.7. Отправка пользовательских заданий на выполнение

После успешного запуска МАС, агент формулировки заданий переходит в состояние ожидания нового задания пользователя. Агент формулировки заданий

может принимать следующие виды постановок задач:

– в виде описания РППП из инструментального комплекса Orlando Tools;
– в виде направленного ациклического графа (Directed Acyclic Graph, DAG), представленного в формате описания заданий для Condor DAGMan или формате JSON;
– в процедурной или непроцедурной постановке.

С помощью редактора (рисунок 3.18) формируется новое задание пользователя для выбранной МАС (MAS). Указываются название задания (project_name), загружается новая, либо выбирается из имеющихся постановка задачи (executable_job, selected_job), источник входных данных (data from file) – при наличии, каталог для сохранения результатов вычислений, входные параметры командной строки (Input comand line args) – при наличии, а также требования к ресурсам (оперативная память, объем диска, архитектура процессора и ОС, предпочитаемый вычислительный ресурс). Готовое описание задания отправляется на выполнение кнопкой «Send».

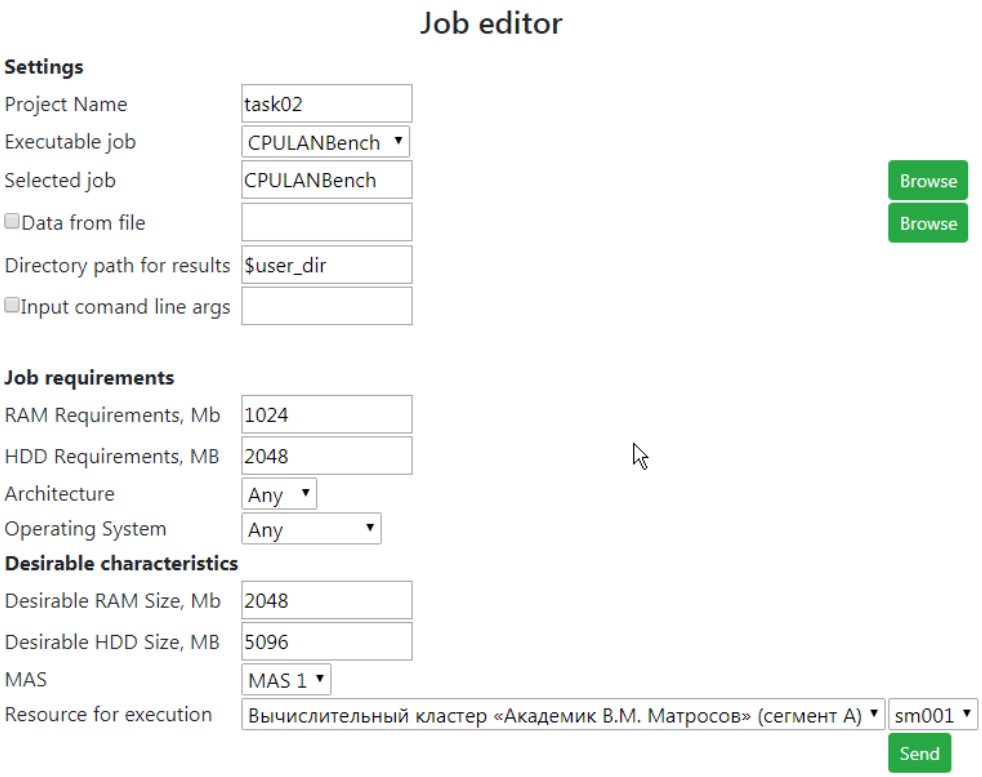
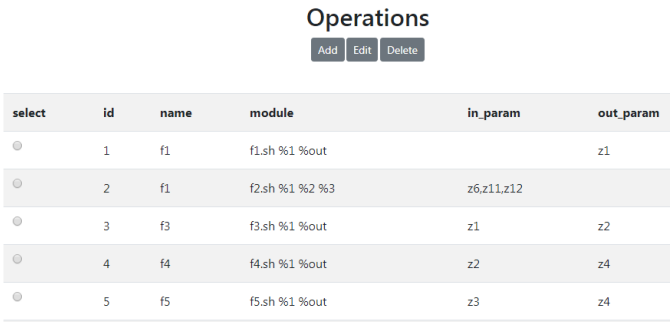


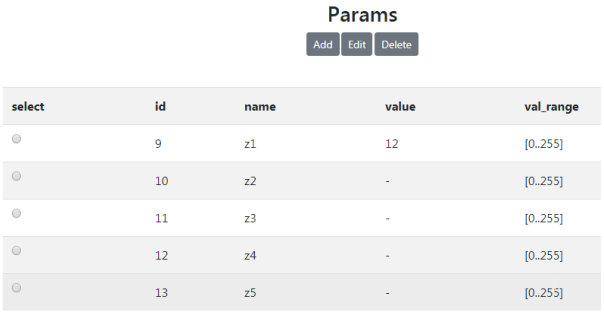
Рисунок 3.18 – Формирование нового задания пользователя

В том случае, если задание представлено в виде DAG, то его модули передаются агенту классификации заданий, затем агенту формирования ВС агентов с последующим проведением тендера вычислительных работ. Формат DAG определяется агентом автоматически.



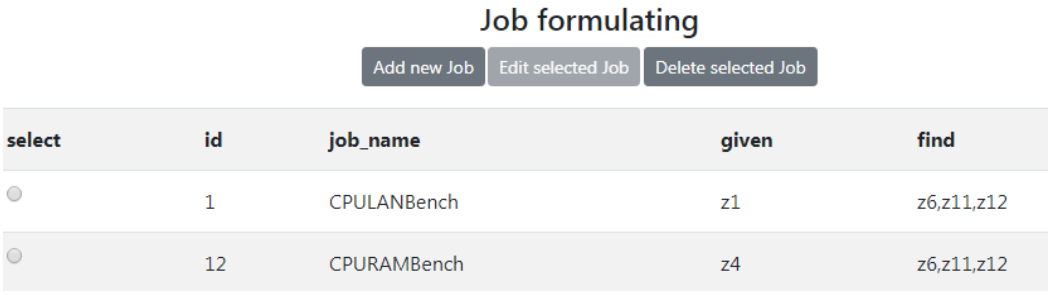
select	id	name	module	in_param	out_param
	1	f1	f1.sh %1 %out		z1
	2	f1	f2.sh %1 %2 %3	z6,z11,z12	
	3	f3	f3.sh %1 %out	z1	z2
	4	f4	f4.sh %1 %out	z2	z4
	5	f5	f5.sh %1 %out	z3	z4

Рисунок 3.19 – Описание операций решаемой задачи



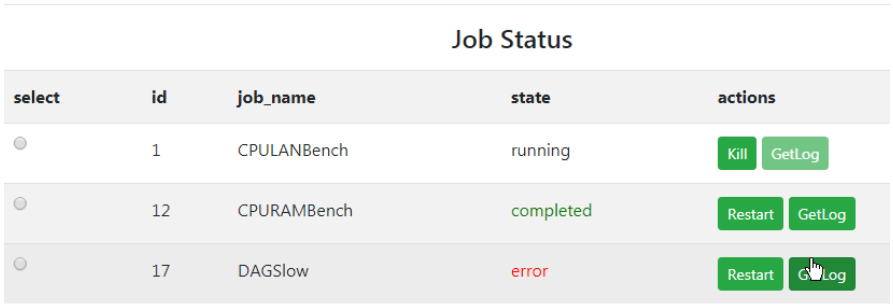
select	id	name	value	val_range
	9	z1	12	[0..255]
	10	z2	-	[0..255]
	11	z3	-	[0..255]
	12	z4	-	[0..255]
	13	z5	-	[0..255]

Рисунок 3.20 – Описание параметров решаемой задачи



select	id	job_name	given	find
	1	CPULANBench	z1	z6,z11,z12
	12	CPURAMBench	z4	z6,z11,z12

Рисунок 3.21 – Постановка задачи



select	id	job_name	state	actions
	1	CPULANBench	running	Kill GetLog
	12	CPURAMBench	completed	Restart GetLog
	17	DAGSlow	error	Restart GetLog

Рисунок 3.22 – Отслеживание очереди заданий МАС

При процедурной или непроцедурной постановке задачи пользователю необходимо описать операции (рисунок 3.19) и параметры (рисунок 3.20) решаемой задачи (модель предметной области), а затем сформулировать

постановку задачи (рисунок 3.21). Например, зная модель предметной области вычислить z_6, z_{11}, z_{12} по z_1. После этого задание отправляется на выполнение, где сначала происходит построение плана (схемы) решения задачи (метод прямой и обратной волны), и при успешном построении плана схема решения задачи передается агенту классификации.

Очередь запущенных заданий выбранной МАС отображается на соответствующей странице (рисунок 3.22). Для каждого задания доступен статус выполнения (running, completed, error), функции остановки и перезапуска (kill, restart), а также скачивания лог-файла (GetLog).

3.5. Выводы

В данной главе рассмотрен инструментальный комплекс организации МАС, автоматизации разработки МАС, а также методика применения предложенного инструментального комплекса. В ней получены следующие основные результаты:

- рассмотрена архитектура агентной платформы JADE;
- разработан инструментальный комплекс автоматизации разработки МАС, включающий специализированную надстройку над JADE;
- предложена методика применения предложенного инструментального комплекса.

Содержание данной главы отражено в публикациях [12, 13, 30–32, 39, 40].

Глава 4. Экспериментальный анализ

Четвертая глава посвящена экспериментальному анализу результатов практической апробации разработанных моделей, методов, алгоритмов и инструментальных средств мультиагентного управления потоками заданий при решении научных и прикладных задач в экспериментальной РВС, организованной с использованием ресурсов ЦКП ИСКЦ и других научных и образовательных организаций.

4.1. Экспериментальная среда

Экспериментальная РВС построена на базе вычислительного кластера «Blackford». В шасси V5000 в исполнении 5U установлены 10 вычислительных модулей семейства T-Blade V205S, разработанных компанией Т-Платформы [104], ведущим российским производителем в области высокопроизводительных вычислений. Каждый модуль обладает следующими характеристиками: два 16-ядерных процессора AMD Opteron 6276 «Bulldozer»/«Interlagos».

Указанная выше конфигурация позволяет использовать данные вычислительные модули для проведения крупномасштабных экспериментов. Кроме того, процессоры, имеющие 16 ядер и 64 GB ОЗУ каждый, обеспечивают возможность запуска виртуальных машин и, как следствие, организацию облачной инфраструктуры.

Экспериментальная РВС включает три пула вычислительных ресурсов. На четырех узлах пула 1 установлен автономный гипервизор VMWare ESXi 5.5 с бесплатной лицензией, а также гипервизоры Citrix XenServer [105], Qemu/KVM [106] и Oracle VM VirtualBox [107] управления гостевыми ОС и системой OpenStack [108] для управления виртуализированными ресурсами. На каждом из гипервизоров запущены готовые образы ОС на базе Windows и GNU/Linux, настроенные для определенных классов задач. Средства виртуализации обеспечивают возможность тонко конфигурировать ресурсы виртуальных машин, используемых для организации вычислений.

На шести узлах пула 2 установлена ОС Ubuntu 16.04. Эти узлы настроены и подготовлены для работы в составе кластера. В узлах установлено следующее системное ПО: системы управления заданиями HTCondor [109] и Portable Batch System / Terascale Open-Source Resource and QUEue Manager (PBS Torque) [110], библиотеки поддержки параллельного программирования Message Passing Interface (MPI) [111] и Open Multi-Processing (OpenMP) [112], компиляторы программ на языках Fortran, C, C++ и Java.

Пул 3 представлен кластером персональных компьютеров (ПК). ПК данного кластера являются невыделенными вычислительными ресурсами. Они могут быть использованы как в составе кластера, так и их непосредственными владельцами.

Состав пулов вычислительных ресурсов отражен в таблице 9. В ней для каждого пула указаны его пиковая производительность R_{max}, число n_n узлов, число n_p процессоров и число n_c ядер, а также используемая система управления заданиями (СУПЗ) и ОС. Детальное описание узлов пулов представлено в таблице 10.

Таблица 9 – Пулы вычислительных ресурсов экспериментальной РВС

Пул	R_{max}, Тфлопс	n_n/ n_p/ n_c	Системы управления заданиями	Поддерживаемые ОС
Пул 1	1.23	4 / 8 / 128	OpenStack	Windows, GNU/Linux
Пул 2	1.84	6 / 12 / 192	HTCondor, PBS Torque	Ubuntu 16.04
Пул 3	0.49	16 / 16 / 32	HTCondor, PBS Torque	Windows, GNU/Linux

К РВС могут быть подключены другие разнородные вычислительные ресурсы: ПК, высокопроизводительные серверы, кластеры ПК, системы хранения данных и др. Потоки заданий поступают в РВС через специальный шлюз, на котором установлен агент постановки задачи и метапланировщики (GridWay и Condor DAGMan) или из РППП [25], разработанных с помощью инструментального комплекса Orlando Tools [31, 113]. Схема мультиагентного

управления потоками заданий в экспериментальной РВС показана на рисунке 4.1.

Таблица 10 – Узлы пулов

Узел	Характеристики узла
Узел пула 1	2 процессора AMD Opteron 6276 (16 core, 2.3 GHz, 16 MB L3 cache, 64 GB ОЗУ DDR3-1600, 4 FLOP/cycle)
Узел пула 2	2 процессора AMD Opteron 6276 (16 core, 2.3 GHz, 16 MB L3 cache, 64 GB ОЗУ DDR3-1600, 4 FLOP/cycle)
Узел пула 3	1 processor Intel Core i3-4000M (2 core with hyper-threading, 2.4 GHz, 2 GB ОЗУ)

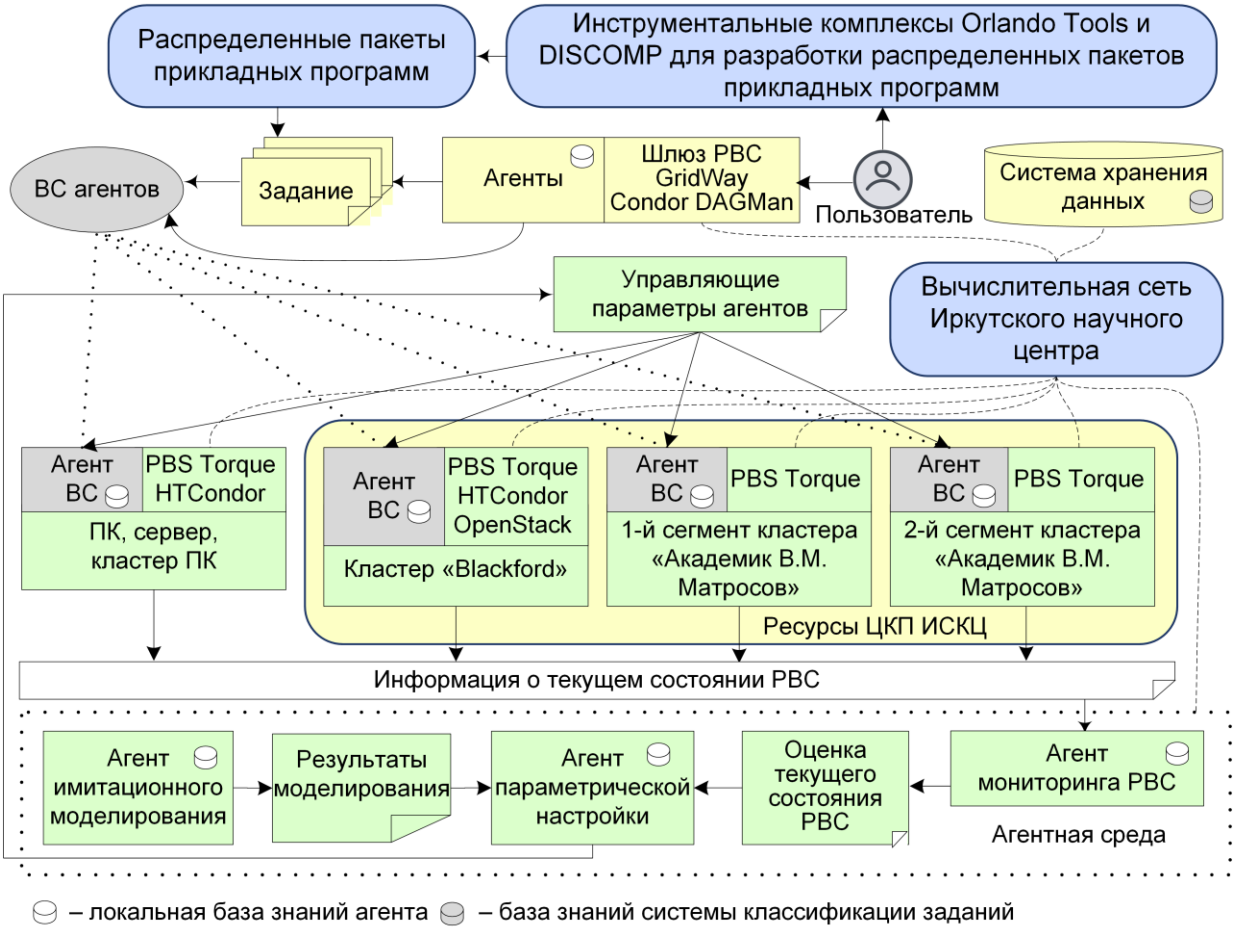


Рисунок 4.1 – Схема мультиагентного управления в экспериментальной РВС

Агенты постановки задачи, классификации заданий и организации виртуального сообщества размещаются в шлюзе РВС. Агент постановки задачи осуществляет формулировку задачи и построение плана ее решения. Агент классификации заданий определяет классы заданий, которым соответствуют модули плана решения задачи. Агенты, представляющие ресурсы и имеющие

возможность работы с требуемыми классами заданий, включаются в виртуальное сообщество агентом формирования виртуального сообщества, которое будет осуществлять выполнение поступившего задания.

В процессе проведения тендера вычислительных работ [90] агенты виртуального сообщества распределяют вычислительную нагрузку, связанную с выполнением задания, между собой.

Агент параметрической настройки способствует обучению агентов виртуального сообщества в процессе их функционирования. Для этого он использует информацию, предоставляемую агентами мониторинга и имитационного моделирования. Агентная среда реализована на основе платформы JADE.

Специальная гипервизорная оболочка [15] обеспечивает возможность выполнения виртуальных машин для заданий из экспериментальной РВС на узлах 1-го и 2-го сегментов кластера «Академик В.М. Матросов» ЦКП ИСКЦ. Агент диспетчеризации заданий в невыделенных ресурсах, функционирующий в рамках данной оболочки, осуществляет поиск свободных окон в расписании СУПЗ указанных выше сегментов, а также формирование и запуск специальных заданий по выполнению требуемых виртуальных машин. Схема запуска виртуальных машин и управления ими на узлах кластера «Академик В.М. Матросов» представлена на рисунке 4.2.

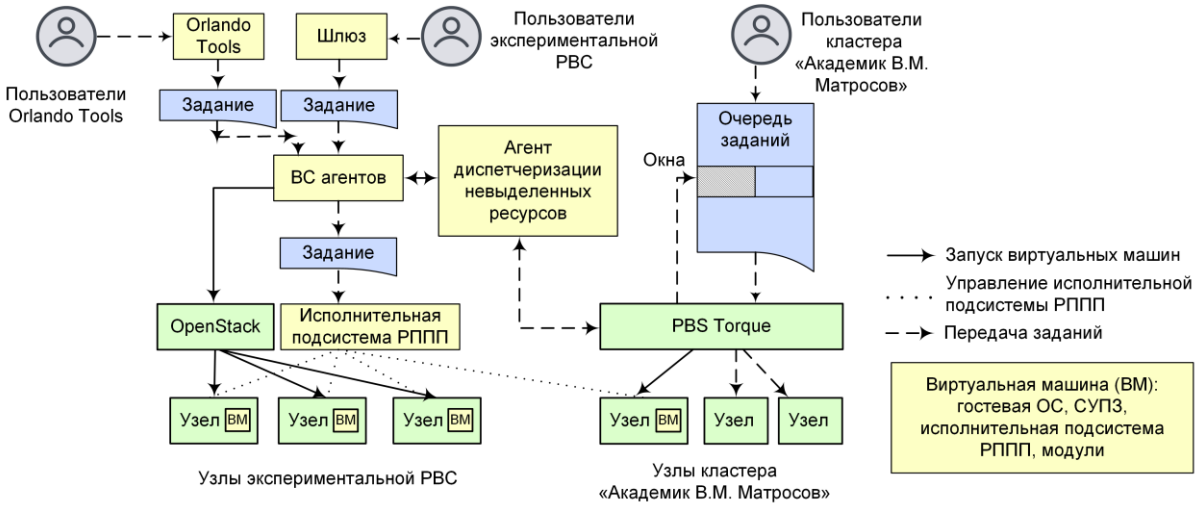


Рисунок 4.2 – Схема запуска виртуальных машин и управления ими

Все виртуальные машины подключаются к единой виртуальной среде, которая управляется исполнительной подсистемой РППП.

4.2. Сравнительный анализ трудоемкости построения мультиагентных систем

Целью сравнительного анализа трудоемкости построения МАС является сопоставление трудозатрат при использовании наиболее популярного инструментария JADE с трудозатратами, которые сокращаются посредством дополнительного применения разработанного в диссертации инструментального комплекса.

Как правило, каждый разработчик агентов с применением JADE выполняет следующие подготовительные этапы:

- подготовка рабочей среды для компиляции и запуска приложений на языке программирования Java;
- загрузка и установка среды разработки на Java;
- загрузка актуальной версии JADE и подключение к новому проекту на Java.

Данные этапы выполняются один раз, однако занимают времени от 15 минут до нескольких часов в зависимости от квалификации разработчика.

Следующие этапы связаны с непосредственной разработкой агентов зависят как от программистских навыков разработчика, так и от степени погружения в документацию JADE. В комплекте поставки JADE имеются демонстрационные версии агентов, с помощью которых можно понять, какие компоненты необходимы для написания простейших агентов. Создание простого агента, принимающего и передающего сообщения, в среднем занимает не больше часа.

Все дальнейшие шаги зависят от назначения агента. Рассмотрим набор необходимых операций для реализации агента, представляющего ресурсы РВС:

- обработка очереди сообщений в соответствии с логическими часами;
- обеспечение надежной доставки сообщений;
- опрос всех доступных агентов в платформе;

- взаимодействие с локальным менеджером ресурсов;
- формирование паспорта задания;
- принятие действий в случае сбоев;
- построение плана решения задачи;
- классификация модулей задачи;
- разработка каркасов родительских и дочерних автоматов;
- подключение внешних модулей пользователей и т.д.

В большинстве агентов повторяются вышеперечисленные операции, соответственно пользователям нет необходимости каждый раз заниматься разработкой всех системных операций, реализованных в предложенном инструментальном комплексе и надстройке над JADE. Реализация ролей агента занимает от двух недель и больше, в зависимости от сложности и состава команды разработчиков.

После успешной разработки агентов необходимо разместить на узлах, в зависимости от сложности РВС этот этап занимает от 20 минут до 1 часа.

На рисунках 4.3 и 4.4 приведены оценки трудозатрат в процессе создания агента, функционирующего на основе новой и существующей ролей соответственно. Данные оценки учитывают затраты на разработку каркасов родительского и дочернего автоматов, реализацию действий агента в составе МАС, подключение разработанного агента к платформе JADE и его размещение в экспериментальной РВС.

Применение разработанного в диссертации инструментального комплекса позволяет сократить трудозатраты на трех самых сложных этапах (разработка каркасов родительского и дочернего автоматов, реализация действий агента) как при реализации новой роли агента, так и при использовании существующей роли.

Небольшие затраты на подключение агента к JADE и его размещение в РВС примерно одинаковы во всех случаях. Это объясняется необходимостью выполнения типовых операций с платформой JADE.

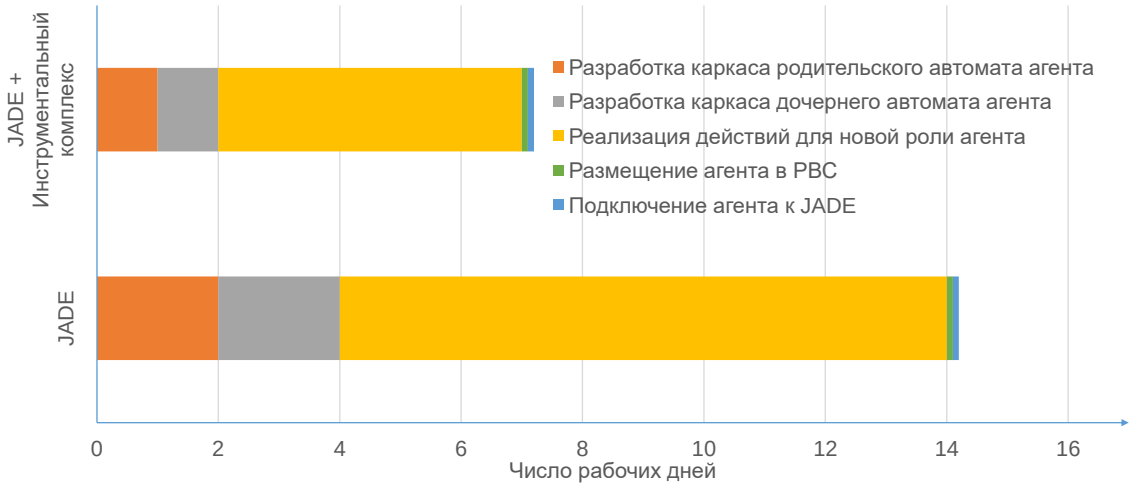


Рисунок 4.3 – Трудозатраты на создание агента с новой ролью

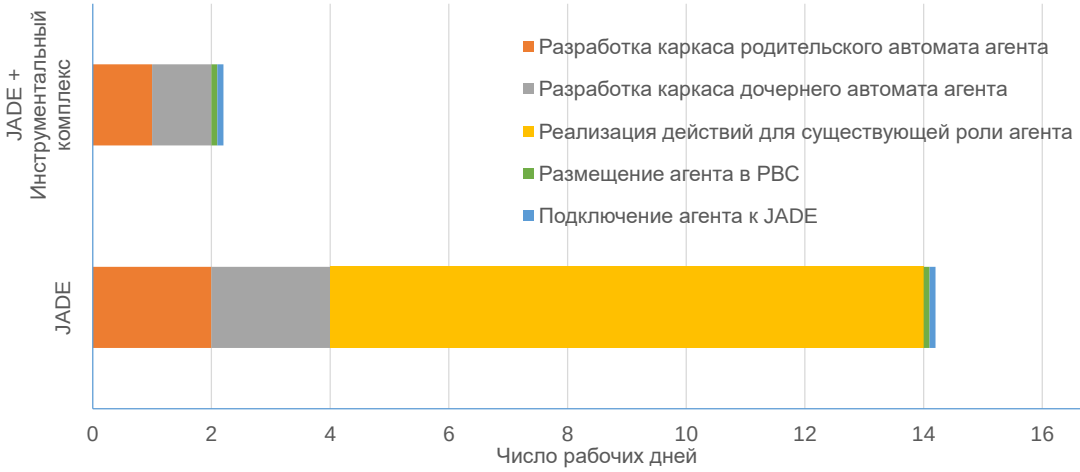


Рисунок 4.4 – Трудозатраты на создание агента с существующей ролью

Таким образом, эффективность применения инструментального комплекса при разработке отдельного агента не вызывает сомнений.

На рисунке 4.5 представлены сравнительные результаты разработки МАС, включающей 10 агентов. Пять агентов разрабатываются на основе существующих ролей, для других пяти агентов создаются новые роли. Для поддержки функционирования агентов разрабатываются база знаний и динамический планировщик их действий.

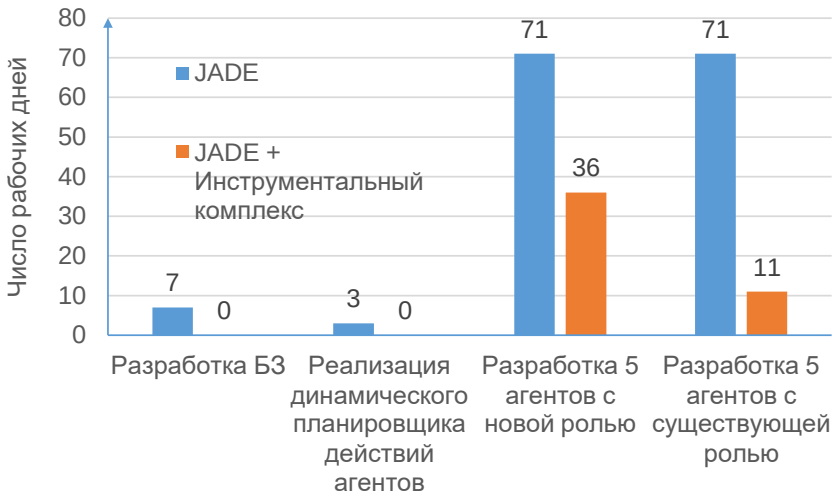


Рисунок 4.5 – Трудозатраты на создание МАС

Очевидно, что использование разработанного в диссертации инструментального комплекса в дополнение к инструментарию JADE позволяет существенно сократить трудозатраты на создание МАС.

Представленные в диссертации показатели трудоемкости построения МАС основаны на экспертных оценках разных разработчиков этой системы, полученных в процессе создания четырех ее версий на основе статистической информации [7, 83, 90]. Данные показатели согласуются с показателями, приведенными в известных работах [114, 115].

4.3. Вычислительные эксперименты

В данном разделе рассмотрены вычислительные эксперименты в рассмотренной выше РВС. Данные эксперименты отражают различные показатели качества функционирования МАС при управлении потоками заданий.

4.3.1. Отказоустойчивость

Проведен сравнительный анализ отказоустойчивости РВС при обработке потоков заданий с помощью полунатурного моделирования. С этой целью было разработано приложение, генерирующее синтетический поток заданий с использованием Standard Workload Format (SWF) на основе вычислительной истории. Статистика вычислений собрана в процессе решения типовых практических задач исследования направлений развития топливо-энергетических

комплексов России [116] и Вьетнама [117] с позиций обеспечения энергетической безопасности, а также оптимизации процессов складской логистики [118]. Модули приложения моделируют работу прикладных программ, использованных в процессе решения перечисленных выше задач. Они выполняют в РВС реальную загрузку ресурсов и обмен заданными объемами данных, соответствующие показателям выполнения реальных прикладных программ. Приложение включает 10 модулей. Характеристики их выполнения представлены в таблице 11.

Таблица 11 – Характеристики выполнения модулей приложения

Задача	Модуль	Среднее время выполнения, сек.	Средний объем данных, МБ	
			Входные	Выходные
Исследование направлений развития топливо-энергетических комплексов России и Вьетнама	m_1	130.01	50.12	54.01
	m_2	172.12	734.18	696.32
	m_3	10.07	368.43	389.71
	m_4	21.04	74.16	81.24
Оптимизация процессов складской логистики	m_5	12.73	0.04	0.05
	m_6	11.26	0.18	0.24
	m_7	24.83	0.09	0.11
	m_8	21.15	0.14	0.12
	m_9	19.71	0.03	0.08
	m_{10}	10.37	0.02	0.04

SWF-формат представлен в [119]. Он разделен на две основные части: описание моделируемой вычислительной системы и спецификация потока заданий, выполняемых в ней. SWF-формат включает набор предопределенных характеристик, каждая из которых имеет свой уникальный порядковый номер. Характеристики, используемые в первой части SWF-формата, могут иметь как числовые, так и текстовые значения. Характеристики, используемые во второй части SWF-формата, могут иметь только числовые значения. Детальное описание характеристик обеих частей SWF-формата приведено в Приложении Е.

Сгенерированный поток заданий был поочередно выполнен под управлением трех систем: метапланировщиков GridWay и CondorDAGMan, а также рассматриваемой в диссертации МАС с обучением агентов и без него.

В процессе работы систем моделировались типовые отказы их компонентов и узлов разнородной РВС. Интенсивность отказов составляла около одного сбоя в час. Такое число отказов превышает их реальную интенсивность для вычислительных кластеров. Однако такая частота отказов позволяет нагляднее продемонстрировать достоинства и недостатки каждой из систем.

Отказы прикладного ПО в данном примере не рассматриваются, так как решение вышеупомянутых научных и прикладных задач осуществлялось с помощью приложений, разработанных в инструментальном комплексе Orlando Tools, поддерживающего непрерывную интеграцию разрабатываемых программ, включая контроль их версий, отладку и тестирование в РВС. Таким образом, выполнение реального потока заданий, на основе которого был сгенерирован синтетический поток, осуществлялось без отказов ПО.

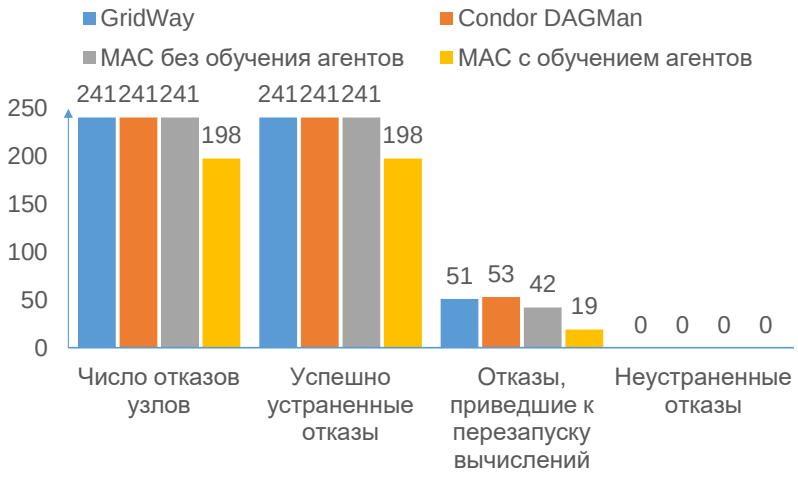


Рисунок 4.6 – Устранение отказов узлов

Рисунок 4.6 показывает успешную обработку отказов узлов всеми тремя системами. Число неустраненных отказов узлов равно 0 во всех четырех случаях.

Однако можно отметить ряд преимуществ мультиагентного управления вычислениями. Во-первых, оно обеспечило снижение числа рестартов по сравнению с традиционными метапланировщиками GridWay и Condor DAGMan. Такое снижение становится особенно очевидным при использовании системы обучения агентов. Во-вторых, обучение агентов позволило снизить число отказов узлов за счет назначения агентами (на основе полученных ими знаний) более

надежных узлов для выполнения заданий. Результаты, представленные на рисунке 4.6, демонстрируют существенное снижение числа отказов узлов под управлением МАС с обучением агентов более, чем на 17% по сравнению с тремя другими случаями.

Преимущества МАС по обработке отказов компонентов системы управления отражены на рисунке 4.7. В случаях возникновения таких отказов агент-лидер брал на себя управление узлом отказавшего агента и продолжал выполнение вычислений в данном узле вплоть до их завершения или момента восстановления отказавшего агента.

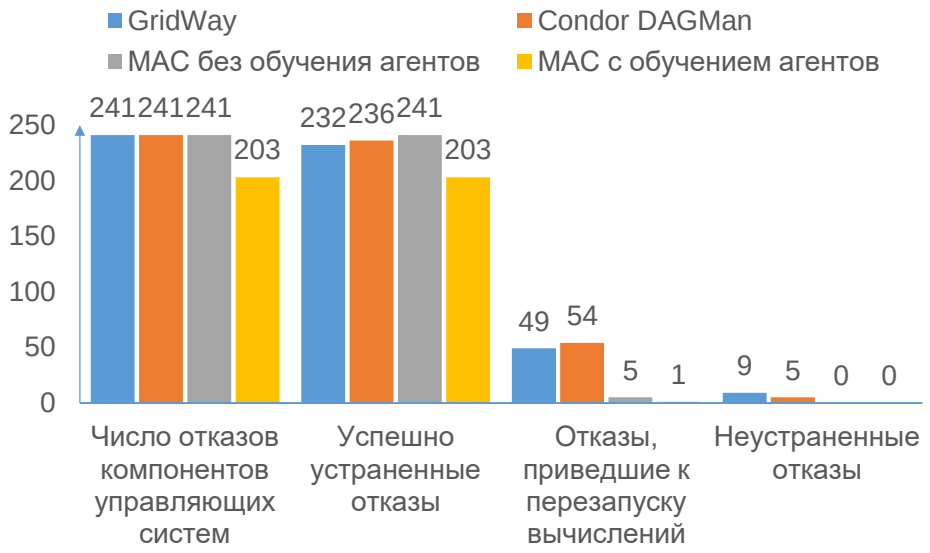


Рисунок 4.7 – Устранение отказов компонентов системы управления

В то же время GridWay и CondorDAGMan ожидали восстановления своих отказавших компонентов и в некоторых случаях перезапускали вычислительные процессы. Обучение агентов позволило улучшить результаты, полученные МАС, функционирующей без применения такого обучения.

Во многих случаях отказы компонентов управляющих систем явно или косвенно обусловлены отказами вычислительных узлов. Результаты, представленные на рисунке 4.7, демонстрируют существенное снижение числа отказов компонентов МАС с обучением агентов более, чем на 15% по сравнению с тремя другими случаями.

Число отказов, приведших к перезапуску вычислений, составляет 1 и 5

соответственно для МАС с обучением агентов и без него. В то же время число таких отказов для GridWay и CondorDAGMan значительно выше (49 и 54 отказа соответственно).

В работе МАС как с обучением агентов, так и без него нет неустраненных отказов компонентов систем. В процессе функционирования GridWay и Condor DAGMan можно видеть 9 и 5 неустраненных отказов соответственно.

Вычислительные эксперименты выполнены в экспериментальной разнородной РВС. В рамках экспериментов суммарное число ядер изменялось от 320 до 352 единиц. Разные узлы характеризовались различной степенью их надежности.

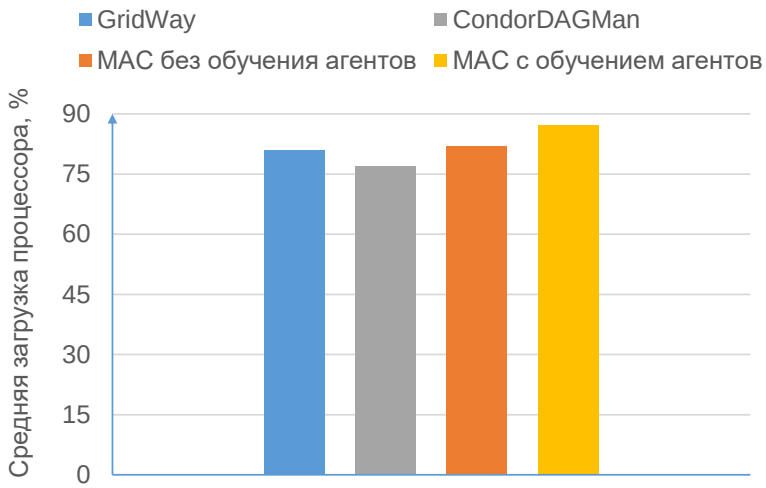


Рисунок 4.8 – Средняя загрузка процессора

В диссертационной работе эффективность использования ресурсов характеризуется средней загрузкой процессоров узлов при обработке потоков заданий разными управляющими системами (рисунок 4.8). Очевидно, что обучение агентов позволяет им выводить ненадежные узлы из РВС, повышая тем самым полезную вычислительную нагрузку на оставшиеся узлы и увеличивая среднюю загрузку процессора.

Средняя загрузка процессора узлов экспериментальной РВС под управлением систем GridWay и Condor DAGMan составляет соответственно 81% и 77%. Применение МАС с обучением агентов и без него позволило повысить данный показатель до 87% и 82% соответственно.

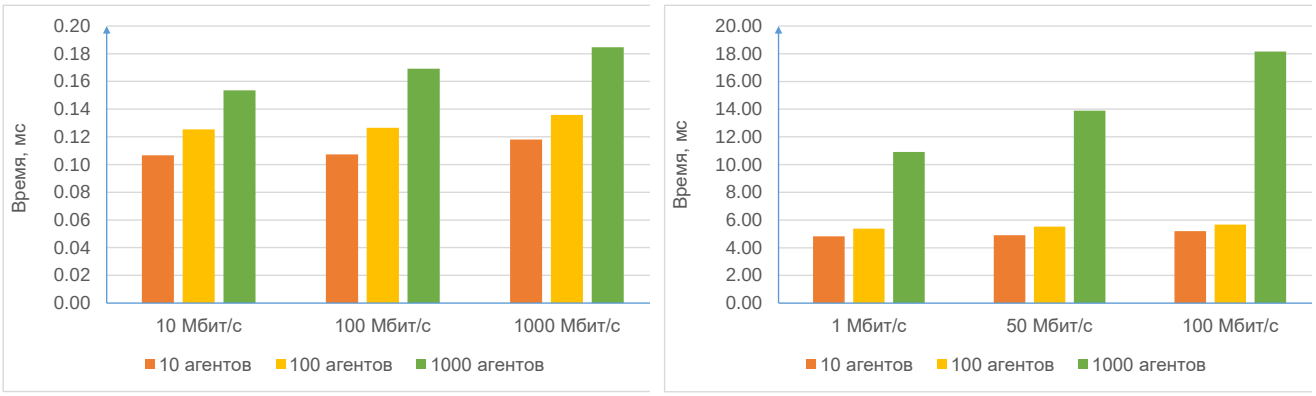
Новые результаты решения вышеупомянутых типовых практических задач [116–118] в целом подтвердили адекватность оценок отказоустойчивости и эффективности использования ресурсов под управлением МАС, полученные в данном разделе диссертационного исследования.

4.3.2. Надежность и эффективность передачи сообщений агентами

Проведен сравнительный анализ эффективности передачи сообщений агентами в различных типах сетей (локальная, городская и глобальная). Число агентов для каждого типа сети составило 10, 100 и 1000. В каждом тесте первый агент отправлял сообщения всем агентам МАС. Замерялось время между отправкой и получением ответного сообщения. Сообщение обрабатывается агентом в среднем за 0.05 мс. Локальная сеть характеризуется низкой латентностью сети и скоростью передачи до 1 Гбит/сек, в городской и глобальных сетях пропускная способность достигает 100 Мбит/сек, однако задержка составляет 4 мс и 50 мс соответственно. В каждом тесте создавалась искусственная загрузка сети.

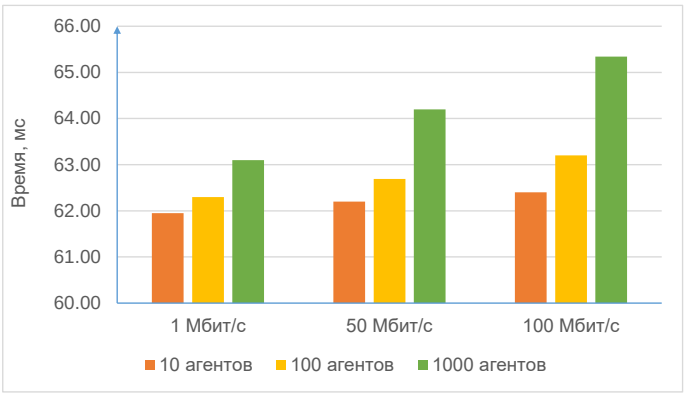
На рисунке 4.9 (а) представлен график среднего времени отправки сообщений в локальной сети со скоростью 1 Гбит/сек при общей загруженности коммуникационного канала в 10, 100 и 1000 Мбит/сек соответственно для 10, 100 и 1000 агентов. Результаты, представленные на данном рисунке, показывают, что время доставки одного сообщения для всех случаев не превышает 1 мс.

Результаты обмена сообщениями агентами, находящимися в городской сети, представлены на рисунке 4.9 (б). Наблюдались скачки времени доставки сообщений от 4 мс до 25 мс. Это связано с несинхронной отправкой сообщений. Потерь сообщений не было. Для 1000 агентов время доставки сообщения увеличилось значительно (с 5 до 18 мс) по сравнению со 100 агентами. Даже такая высокая нагрузка на сеть позволяет агентам быстро обменяться сообщениями. Отчасти это зависит от приоритизации трафика на оборудовании провайдера. Для создания нагрузки на сеть был запущен двусторонний обмен между серверами большими файлами.



а)

б)



в)

Рисунок 4.9 – Время передачи сообщений агентами в локальной (а), городской (б) и глобальной (в) сети

Рисунок 4.9 (в) отражает результаты обмена сообщениями агентами в глобальной сети (агенты расположены на географически удаленных узлах в разных городах). Результаты, представленные на данном рисунке, показывают, что минимальное время доставки сообщений составляет более 60 мс для 10 агентов и максимальное время для 1000 агентов – 65 мс. Существенное время затрачено на доставку сообщения по коммуникационным каналам, количество агентов и общая нагрузка на сеть значительно не повлияли на время доставки сообщений. Потери в доставке не наблюдались. Таким образом, результаты проведенных экспериментов показали надежность и эффективность обмена сообщениями между агентами МАС.

4.3.3. Оценка качества обслуживания очереди заданий с помощью мультиагентной системы

В настоящее время важной проблемой является повышение качества

обслуживания очередей заданий, представляющих собой взаимосвязанные схемы решения задачи, которые коррелируют с понятием рабочего процесса (англ., workflow [120]). В связи с этим в диссертации разработано приложение, включающее 20 тестовых модулей, которые моделируют работу прикладных программ ряда известных рабочих процессов. В их числе Montage [121], BLAST [122] и Broadband [123]. Характеристики выполнения модулей представлены в таблице 12. Тестовые модули приложения выполняют в РВС реальную загрузку ресурсов и обмен заданными объемами данных, соответствующие статистической информации о работе ПО реальных научных процессов.

Таблица 12 – Характеристики выполнения модулей рабочих процессов Montage, Blast, Broadband

Рабочий процесс	Модуль	Прикладная программа	Среднее время выполнения, сек.	Средний объем данных, МБ	
				Входные	Выходные
Montage	m_1	mProjectPP	1.73	2.05	8.09
	m_2	mDiffFit	0.66	16.56	0.64
	m_3	mConcatFit	143.26	1.95	1.22
	m_4	mBgModel	384.49	1.56	0.10
	m_5	mBackground	1.72	8.36	8.09
	m_6	mImgtbl	2.78	1.55	0.12
	m_7	mAdd	282.37	1102.57	775.45
	m_8	mShrink	66.10	411.50	0.49
	m_9	mJPEG	0.64	25.33	0.39
BLAST	m_{10}	Blast synteny	33.05	3.30	1.35
	m_{11}	Blast candidate	5.76	1.37	0.01
	m_{12}	Blast QRNA	1344.88	258.93	4.50
	m_{13}	Blast paralogues	4.54	1.20	1.24
Broadband	m_{14}	ucsb_createSRF	0.57	4.55	7.45
	m_{15}	urs_createSRF	0.43	0.00	1.96
	m_{16}	urs_lp_seisgen	84.92	43.65	0.63
	m_{17}	urs_hf_seisgen	3.06	5.54	3.31
	m_{18}	sdsu_hf_seisgen	196.40	938.28	1855.82
	m_{19}	ucsb_seisgen	68.32	2271.46	5.11
	m_{20}	rspectra	0.34	0.81	0.46

В данном примере рассматривается управление синтетическим потоком

заданий таких модельных научных рабочих процессов. Синтетический поток поочередно выполняется под управлением МАС, GridWay и Condor DAGMan в экспериментальной РВС.

Узлы кластера ПК, входящего в РВС, обладают пониженной степенью надежности и подвержены частым отказам. Учет агентами МАС степени надежности узлов позволяет им назначать наиболее надежные ресурсы для выполнения модулей заданий. Таким образом, агенты снижают число рестартов модулей и тем самым повышают показатели обслуживания очередей заданий.

На рисунке 4.10 приведены результаты сравнения процессов функционирования разработанной МАС с системами GridWay и Condor DAGMan с точки зрения теории массового обслуживания.

Эти результаты показывают существенное улучшение следующих четырех показателей качества обслуживания очереди заданий при использовании МАС:

- *a* – числа заданий с нулевым временем ожидания;
- *b* – среднего времени пребывания задания в очереди;
- *c* – коэффициента полезного использования ресурсов РВС;
- *d* – общего времени выполнения заданий потока.

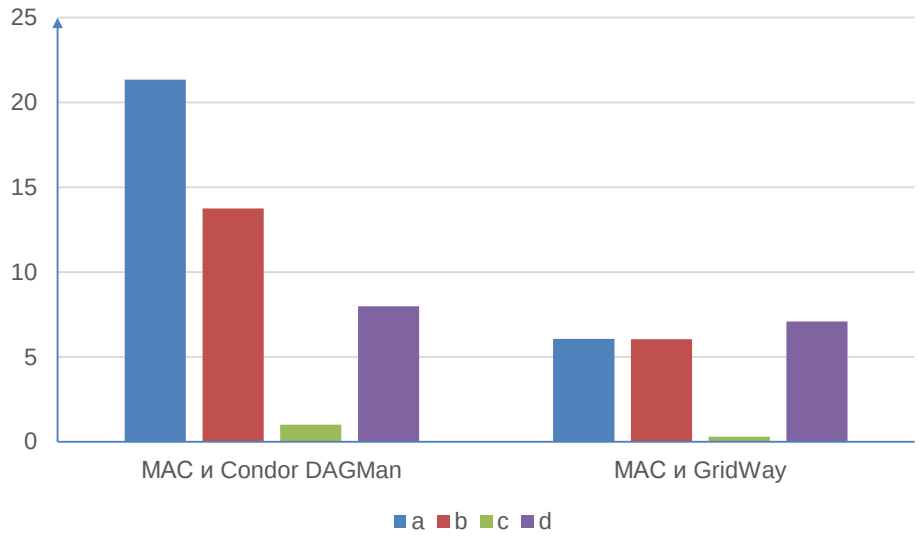


Рисунок 4.10 – Улучшение показателей качества обслуживания очереди заданий

Таким образом, улучшение перечисленных выше четырех показателей

обслуживания очередей заданий составляет от 1.01% до 21.34% по сравнению с системой Condor DAGMan и от 0.30% до 7.09% по сравнению с системой GridWay.

4.3.4. Балансировка загрузки вычислительных ресурсов

В рамках однородной вычислительной системы, например, на кластере достаточно эффективное планирование вычислений и распределение ресурсов может быть построено на базе СУПЗ. Как правило, СУПЗ поддерживают несколько очередей заданий. Задания распределяются по очередям в зависимости от категории пользователя, приоритета задания, класса задания и его требований к ресурсам.

Для эффективного использования вычислительных ресурсов необходима балансировка их нагрузки. Использование ресурсов не является эффективным, если одна часть узлов перегружена, в то время как другая часть узлов простаивает.

Поддержка миграции заданий между узлами позволяет сбалансировать нагрузку ресурсов. Миграция обеспечивает возможность прерывания, перемещения в другие узлы и повторного запуска задания. Она может быть обусловлена перегрузкой одних узлов и простоем других.

Однако организация миграции заданий между вычислительными кластерами с разными СУПЗ является нетривиальной проблемой в разнородной РВС. Применение МАС позволяет частично решить данную нетривиальную проблему.

В качестве примера рассмотрена обработка реального и синтетического потока заданий в экспериментальной РВС. Характеристики заданий синтетического потока (время поступления, число ядер, размер оперативной и дисковой памяти, время выполнения и др.) полностью соответствуют характеристикам заданий реального потока.

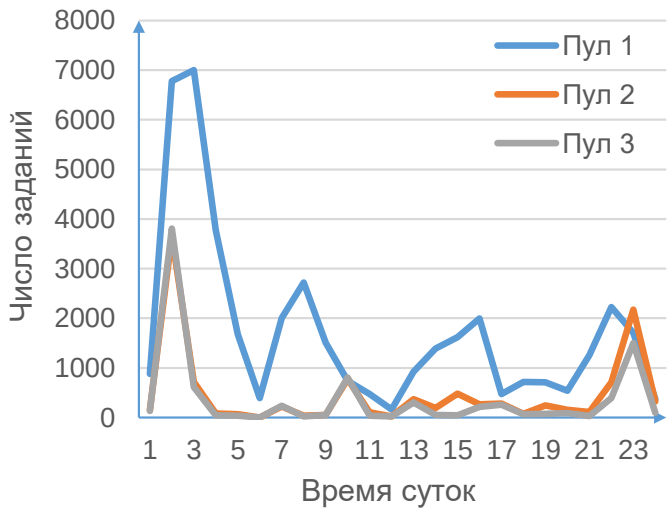


Рисунок 4.11 – Балансировка загрузки с помощью традиционных средств HTCondor и PBS Torque, используемых в пулах ресурсов РВС

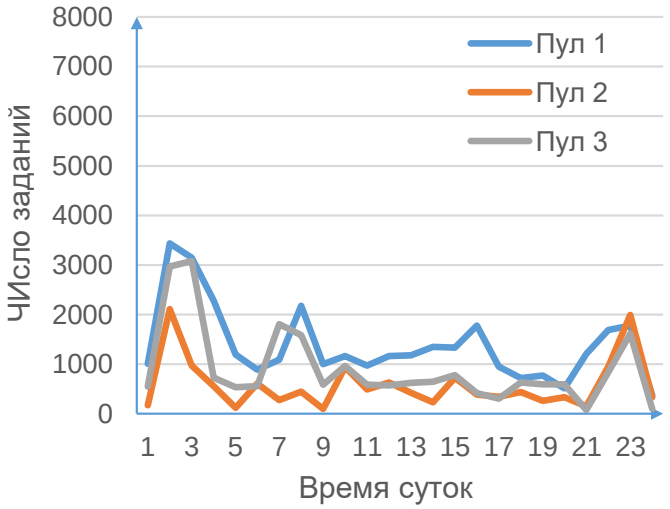


Рисунок 4.12 – Балансировка нагрузки с применением дополнительного уровня управления, реализуемого с помощью МАС

На рисунках 4.11 и 4.12 показано изменение распределения заданий реального потока в пулах 1-3 в течение суток по часам с помощью традиционных средств HTCondor и PBS Torque, а также с применением дополнительного уровня управления, реализуемого с помощью МАС для синтетического потока заданий.

С целью улучшения балансировки загрузки ресурсов МАС осуществляла перераспределение вычислительной нагрузки. Задания можно было назначать на ресурс, отличный от ресурса, который использовался в процессе обработки

реального потока заданий, только в том случае, если перераспределяемое задание сохраняло принадлежность определенному для него классу заданий.

Рисунок 4.13 демонстрирует улучшение средней загрузки процессора при управлении заданиями с помощью МАС по сравнению с традиционными СУПЗ, используемыми на практике в пулах 1-3. Такое улучшение получено за счет перераспределения заданий с небольшим временем выполнения и минимальным числом требуемых ядер из пула 1 в пулы 2 и 3.

Данное перераспределение заданий обеспечило продвижение и запуск на выполнение больших заданий в пуле 3, а также увеличение загрузки узлов пулов 2 и 3. В результате большое число заданий, которые оставались в очереди на практике в конце рассматриваемого периода времени, были выполнены при обработке синтетического потока заданий.

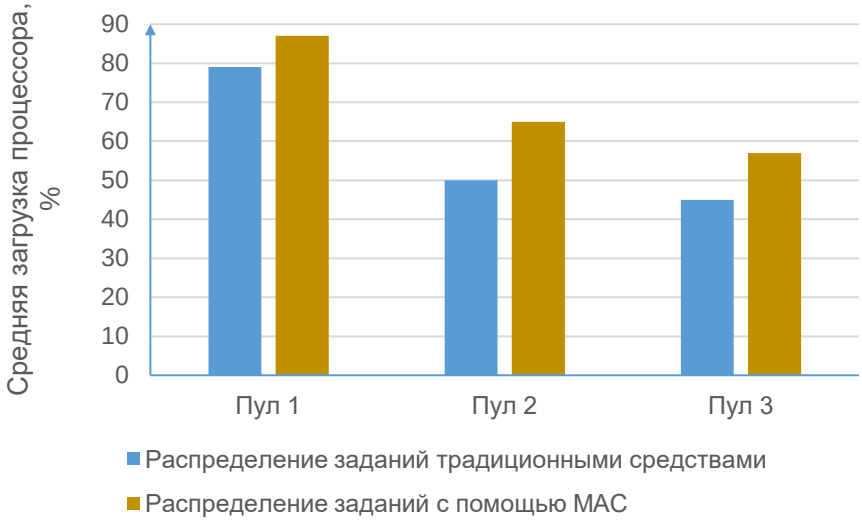


Рисунок 4.13 – Средняя загрузка процессора

В целом перераспределение заданий позволило повысить среднюю загрузку процессоров в пулах 1-3 на 8%, 15% и 12% соответственно. При этом улучшение среднеквадратического отклонения загрузки процессора по часам от средней загрузки в случае мультиагентного управления составило более 58%, 36% и 3% для пулов 1-3 соответственно по сравнению с использованием только систем HTCondor и PBS Torque. В свою очередь улучшение балансировки загрузки привело к сокращению времени выполнения потока заданий в целом на 17%.

4.3.5. Выявление и использование окон в расписании очередей заданий

В процессе работы гипервизорной оболочки (см. раздел 4.1) требуется определение свободных окон в расписании СУПЗ, установленной в узлах сегментов кластера «Академик В.М. Матросов», при запуске виртуальных машин для выполнения заданий из экспериментальной РВС. Это связано с большой погрешностью заказа времени в заданиях пользователей ЦКП ИСКЦ. Под окном понимаются незанятые ядра процессора вычислительного узла. Пример погрешностей времени заданий, заказанного двадцатью четырьмя пользователями ЦКП ИСКЦ, приведен в таблице 13.

Таблица 13 – Статистика времени выполнения заданий

Пользователь	Число заданий	Значение погрешности, сек.	
		Среднее значение	Среднеквадратическое отклонение
1	38	61307	49084
2	15	14192	15145
3	21	488372	480193
4	55	223717	212018
5	85	357498	344252
6	19	535278	470071
7	24	526475	522144
8	154	23381	22558
9	428	501837	413074
10	43	54108	42512
11	50	205890	198408
12	235	34718	33373
13	93	467255	445726
14	970	319049	316870
15	103	125644	118814
16	57	247568	224205
17	16	76483	71504
18	129	573595	549392
19	18	131701	147010
20	221	423511	391279
21	75	280807	231231
22	23	302757	273281
23	339	621142	605028
24	505	580807	564947

Очевидно, что среднее отклонение времени выполнения заданий, указанного в них пользователями, от реального времени выполнения этих заданий варьируется от 14192 до 621142 секунд. Погрешность определения окон на основе анализа вычислительной истории с учетом заданных допусков на запуск и завершение заданий составляет в среднем 35%.

В диссертации разработан специализированный алгоритм определения длительности окон в расписании СУПЗ. Блок-схема данного алгоритма представлена на рисунке 4.14.

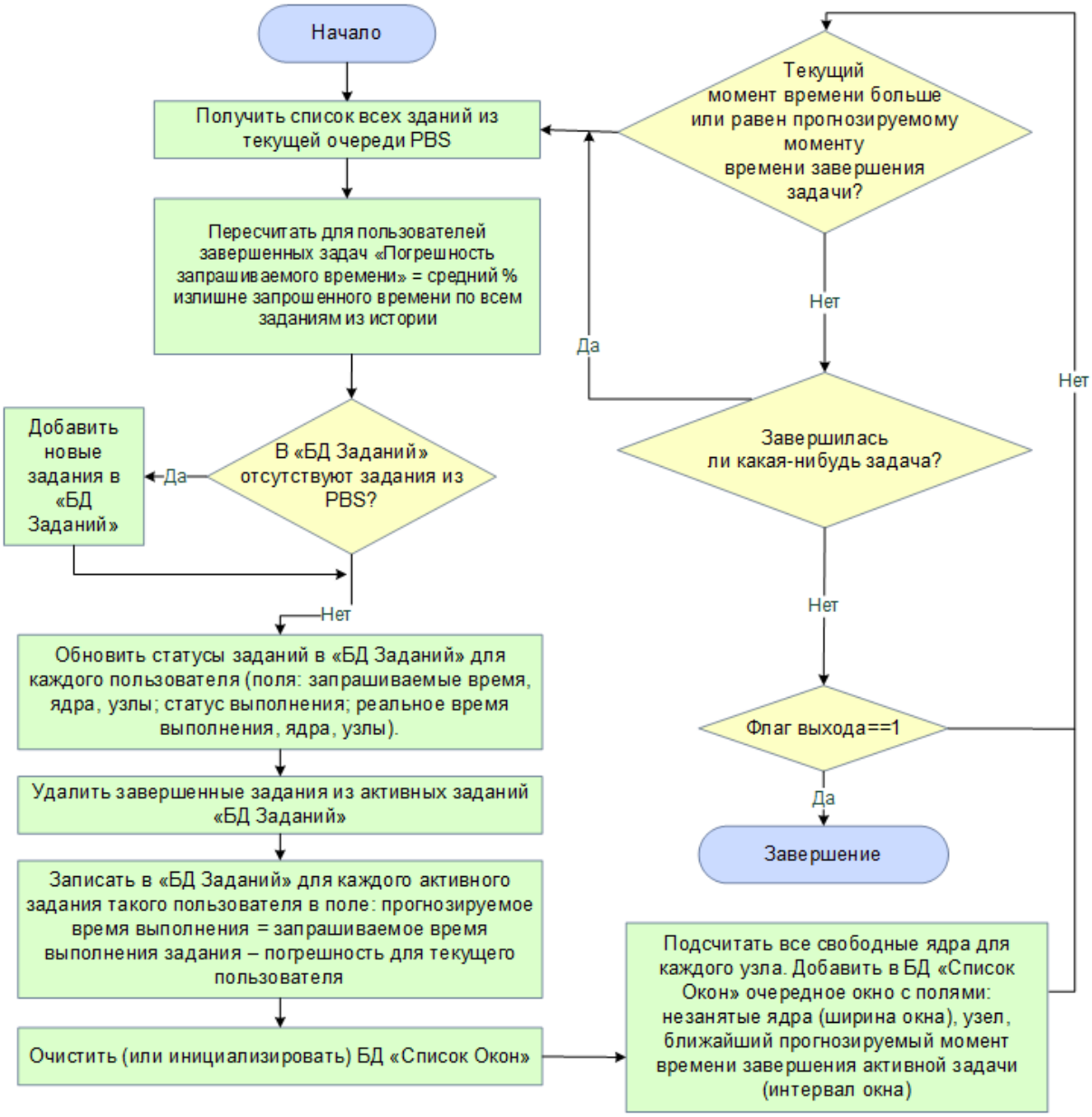


Рисунок 4.14 – Блок-схема алгоритма нахождения свободных окон

Применение алгоритма продемонстрировано на примере решения задачи развития энергетического сектора Вьетнама для оптимизации энергетической безопасности на определенный период [26]. РППП для решения данной задачи разработан с помощью инструментального комплекса DISCOMP [124]. На рисунке 4.15 представлены свободные окна, которые появлялись в расписании обслуживания заданий СУПЗ в процессе решения задачи. Зеленым цветом выделены окна, удовлетворяющие условиям выполнения заданий. Эти окна были использованы для ускорения вычислений. Использование удовлетворительных окон позволило ускорить процесс решения задачи на 16%.

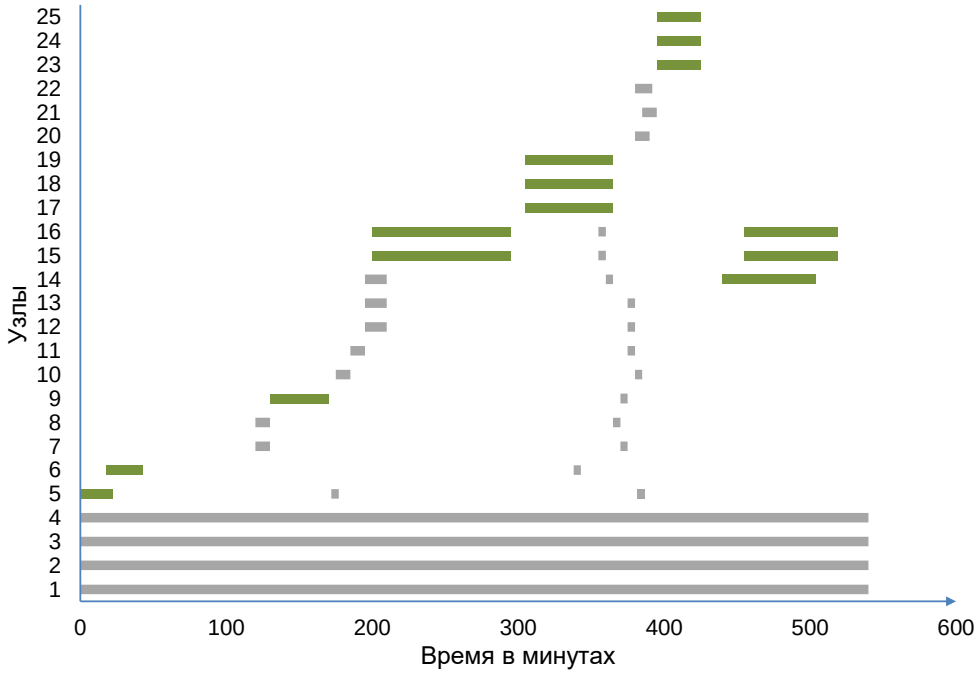


Рисунок 4.15 – Окна в расписании обслуживания заданий СУПЗ

4.4. Выводы

Основные результаты четвертой главы заключаются в практической апробации разработанных моделей, методов, алгоритмов и инструментальных средств мультиагентного управления потоками заданий при решении научных и прикладных задач в экспериментальной РВС, организованной с использованием ресурсов ЦКП ИСКЦ и Иркутского научного центра СО РАН.

Результаты экспериментального анализа показывают существенное повышение отказоустойчивости и эффективности управления потоками заданий,

включая функционирование самих агентов, при использовании системы машинного обучения, предложенной в диссертации. В частности, результаты проведенных экспериментов показали надежность и эффективность обмена сообщениями между агентами МАС.

Применение МАС позволяет улучшить балансировку вычислительных ресурсов, а также повысить среднюю загрузку процессоров узлов экспериментальной РВС.

Повышение эффективности использования ресурсов обуславливает сокращение времени решения научных и прикладных задач в РВС, а также других важных характеристик с точки зрения теории массового обслуживания.

Ускорение вычислений и повышение эффективности использования ресурсов во многом обусловлены использованием результатов анализа вычислительной истории агентами.

Преимущества предложенного в работе мультиагентного подхода к управлению потоками заданий в разнородной РВС по сравнению с известными метапланировщиками продемонстрировано в Приложении Ж на примере конкретного задания. Подзадания этого задания выполняются в общем синтетическом потоке заданий, сгенерированном на основе вычислительной истории, накопленной в процессе решения ряда научных и практических задач.

Основные результаты, полученные в четвертой главе, приведены в публикациях [7, 8, 15, 19, 29, 33, 35, 36, 37, 38, 40].

Заключение

В целом в диссертации представлен новый подход к организации управления потоками заданий в разнородной РВС на основе мультиагентных технологий. Характерной особенностью данного подхода является тесная интеграция моделей, методов и средств концептуального, имитационного, конкретизирующего и автоматного программирования, машинного обучения агентов (классификации заданий и параметрической настройки алгоритмов их работы агентов), организации распределенных вычислений и управления ими в процессе создания и применения предложенных в диссертации новых мультиагентных моделей, алгоритмов и системы управления потоками заданий в разнородной РВС, а также инструментального комплекса, поддерживающего их разработку.

В рамках диссертационного исследования получены следующие основные результаты:

1) разработана ролевая модель поведения агентов;
2) создана система их машинного обучения;
3) предложен мультиагентный алгоритм перераспределения ресурсов разнородной РВС в случае отказа ее программно-аппаратных средств;
4) реализован инструментальный комплекс построения МАС.

Результаты исследования направлены на развитие известных моделей поведения агентов, используемых для управления распределенными вычислениями. Они обеспечивают возможность перехода к ролевому ситуационному поведению агентов в процессе обработки потоков заданий, реализуемом на основе кооперации и соперничества в рамках динамически возникающих социальных организаций (виртуальных сообществ) агентов. Вышеупомянутые особенности поведения агентов позволяют существенно повысить качество мультиагентного управления в процессе решения общей задачи [125].

Все задачи, поставленные перед исследованием, были успешно выполнены.

Предложенные в диссертации оригинальные модели, методы, алгоритмы и инструментальные средства управления потоками вычислительных заданий в разнородных РВС, допускают свое естественное развитие и обобщение применительно к другим классам информационно-вычислительных и управляющих систем.

Список принятых сокращений

ВС – виртуальное сообщество

ИСКЦ – Иркутский суперкомпьютерный центр СО РАН

МАС – мультиагентная система

ОЗУ – оперативное запоминающее устройство (оперативная память)

ОС – операционная система

ПО – программное обеспечение

ППП – пакет прикладных программ

РВС – распределенная вычислительная система

РППП – распределенный пакет прикладных программ

СУПЗ – система управления прохождением заданий

ЦКП – Центр коллективного пользования

ЭВМ – электронно-вычислительная машина

ACL – Agent Communication Language

AppLeS – Application Level Scheduling

ARCOL – ARTIMIS COmmunication Language

ARTIMIS – The Advanced Regional Traffic Interactive Management and Information System

Condor DAGMan – Condor Directed Acyclic Graph Manager

COOL – Domain independent COOrdination Language

Cougaar – Cognitive Agent Architecture

DAG – Directed Acyclic Graph

FIPA – Foundation for Intelligent Physical Agents

FLOPS – Floating Operations Per Second

GAMA – General Agent Modeling Approach

HPC – High Performance Computing

JADE – Java Agent DEvelopment Framework

JADEx – Java Agent Development framework for programming intelligent software agents in XML

JVM – Java Virtual Machine

KQML – Knowledge Query and Manipulation Language

MAAG – Multi-Agent Architecture for Grid Environment

MaDKit – Multi Agent Development Kit

MAGE – Mobile Agent-based Grid Environment

MPI – Message Passing Interface

OpenMP – Open Multi-Processing

SWF – Standard Workloads Format

QoS – Quality of Service

Глоссарий[1]

Агент – аппаратная или программная сущность, способная действовать в интересах достижения целей, поставленных перед ним владельцем и/или пользователем.

Автономность – способность функционировать без вмешательства человека, осуществляя самоконтроль над своими действиями и внутренним состоянием.

Вычислительная единица РВС – программно-аппаратный ресурс, требуемый для решения некоторой вычислительной задачи, включающий одно ядро процессора, часть общей оперативной памяти и памяти жесткого диска вычислительного узла, а также системное ПО.

Вычислительная модель – совокупность значимых величин предметной области, их ограничений и информационно-логических связей между ними.

Виртуальный компьютер – программная среда, запускаемая на вычислительном узле и моделирующая физический компьютер (с ограниченным объемом памяти, процессорного времени, дискового пространства).

Вычислительный кластер – это совокупность вычислительных узлов, объединенных в рамках некоторой коммуникационной среды и представляющих с точки зрения пользователя единый аппаратный ресурс для решения определенного класса задач.

Вычислительный модуль – программная реализация определенного алгоритма в ППП, включающая исполняемую программу, спецификации по назначению, применению, формату входных и выходных формальных параметров,

---

1. Wooldridge, M.J. An Introduction to Multiagent Systems / M.J. Wooldridge. — Chichester: Wiley, 2002. — 340 p.
2. Аверкин, А.Н. Толковый словарь по искусственному интеллекту / А.Н. Аверкин, М.Г., Гаазе-Рапопорт, Д.А. Поспелов. — М.: Радио и связь, 1992. — 254 с.
3. Городецкий, В.И. Самоорганизация и многоагентные системы / В.И. Городецкий // Известия РАН. Теория и системы управления. — 2012. — № 2. — С. 92–120.
4. Гергель, В.П. Высокопроизводительные вычисления для многоядерных многопроцессорных систем: Учебник / В.П. Гергель. Издательство Нижегородского госуниверситета: Нижний Новгород, 2010. — 421 с.

времени выполнения и пр.

Вычислительный процесс – процесс исполнения схемы решения задачи в РВС, включающий трансляцию этой схемы во внутреннее представление, пошаговую интерпретацию и выполнение необходимых инструкций в узлах РВС.

Вычислительные ресурсы – средства для решения прикладных задач, формируемые путем интеграции вычислительных узлов РВС, коммуникаций, прикладного ПО и систем хранения данных.

Вычислительный эксперимент – метод изучения физических явлений, объектов и процессов их функционирования, базирующихся на построении математической модели и ее численном исследовании, позволяющем воспроизвести поведение исследуемого объекта в различных условиях или в различных модификациях.

Вычислительный узел РВС – программно-аппаратный ресурс, включающий модуль оперативной памяти, один или несколько процессоров, жесткий диск и системное ПО (комплекс программных средств, поддерживающих функционирование узла в РВС).

Желания агента – состояния, достижение которых является для агента желательным. Они могут быть противоречивыми, но агент может выбирать только их непротиворечивое подмножество.

Задание пользователя – это совокупность инструкций для пакета прикладных программ по решению прикладной задачи (постановок задачи), сформулированных на входном языке ИК, а также наборов входных данных с исходными значениями и наборов выходных данных для размещения результатов счета.

Инструментальный комплекс – это совокупность программных средств, обеспечивающих автоматизированную поддержку всех этапов создания и применения программ пользователя.

Коммуникационная среда – совокупность программно-аппаратных средств (сетевых устройств, транспортных протоколов) для обеспечения сетевого

взаимодействия двух и более компьютеров.

Концептуальная схема предметной области пакета прикладных программ – совокупность объектов данной предметной области (параметров и модулей) и отношений, определяющих взаимосвязь между этими объектами и характеризующихся набором заданных свойств и ограничений целостности.

Латентность – время между инициированием передачи данных в процессе посылки и прибытия первого байта в процессе приема.

Масштабируемость – это способность системы увеличивать вычислительную мощность без существенного снижения производительности за счет включения в состав РВС бóльшего числа вычислительных узлов.

Модуль – это объект концептуальной схемы предметной области пакета прикладных программ, реализующий абстрактную операцию над параметром данной предметной области.

Мультиагентная система – сеть слабо связанных решателей частных проблем (агентов), которые существуют в общей среде и взаимодействуют между собой для достижения тех или иных целей системы.

Наличие знаний у агента – обладание информацией о себе, о среде и о других агентах, а также правил применения решений на основе этой информации.

Намерения агента – то, что агент обязан сделать в соответствии со своим выбором или в силу своих обязательств по отношению к другим агентам.

Общественное поведение агента – способность функционировать в сообществе агентов, обмениваясь сообщениями с помощью некоторого языка коммуникаций.

Обязательства агента – те задачи, которые агент берет на себя по просьбе или по поручению других агентов в рамках кооперативных целей.

Пакет прикладных программ – это комплекс взаимосвязанных прикладных программ и средств системного обеспечения (программных и языковых), предназначенный для автоматизации решения определенного класса задач конкретной предметной области.

Параметр – это объект концептуальной схемы предметной области пакета прикладных программ, представляющий собой значимую характеристику данной предметной области.

Правдивость агента – свойство агента не манипулировать информацией, про которую ему заведомо известно, что она ложна.

Предметная область – это сфера научной, экономической, информационной или другой деятельности, являющаяся объектом исследования в пакетах прикладных программ.

Прикладное ПО – ПО, состоящее из отдельных прикладных программ и пакетов прикладных программ, предназначенных для решения различных задач пользователей, а также автоматизированных систем, созданных на основе этих прикладных программ.

Проактивность агента – способность действовать в упреждающей манере, в частности, генерировать новые цели и действовать рационально для их достижения, а не только реагировать на события.

Пропускная способность – это величина, обратная времени, необходимому для передачи одного байта, характеризующая скорость передачи больших объемов данных.

Распределенные вычисления – способ решения ресурсоемких вычислительных задач с использованием двух и более компьютеров, объединенных в сеть.

Распределенная вычислительная система – совокупность распределенных рабочих станций (вычислительных узлов), коммуникационной сети и промежуточного ПО, предназначенного для управления процессом решения задач в узлах этой вычислительной среды.

Распределенный пакет прикладных программ – пакет, прикладные программы которого размещены в различных узлах РВС.

Рациональность агента – свойство агента действовать так, чтобы достигнуть своих целей, а не избегать их достижения в рамках своих знаний и

убеждений.

Реактивность агента – способность воспринимать состояние среды и своевременно реагировать на ее изменения.

Ресурсоемкая задача – задача вычислительного характера, решение которой требует выполнения прикладного ПО и использования достаточно больших объемов вычислительных ресурсов (процессорного времени, оперативной памяти, дискового пространства).

Самоорганизация мультиагентной системы – динамический адаптивный процесс, приводящий к возникновению и поддержке структур агентов и их локальных взаимодействий без внешнего вмешательства.

Система управления прохождением заданий – системное ПО, предназначенное для поддержания очередей и планирования работ на вычислительном кластере.

Системное ПО – это комплекс программ, которые обеспечивают эффективное управление компонентами вычислительной системы, такими как процессор, память, каналы ввода-вывода, сетевое и коммуникационное оборудование и т. п.

Схема решения задачи – модель программы, отражающая информационно-логическую структуру вычислений в терминах понятий предметной области.

Убеждения агента – знания агента о недостоверной внешней среде и о других агентах, которые могут изменяться во времени и становиться неверными.

Цели агента – конкретное множество конечных и промежуточных состояний, достижение которых агент считает реализацией своих намерений.

Grid – согласованная, открытая и стандартизированная среда, которая обеспечивает гибкое, безопасное, скоординированное разделение ресурсов в рамках виртуальной организации. Grid состоит из географически распределенных ресурсов, средств телекоммуникаций ресурсов и связующего ПО, поддерживающего выполнение дистанционных операций, а также выполняющего функции контроля и управления ОС.

Литература

1. Lewis, P.R. Resource allocation in decentralised computational systems: an evolutionary market-based approach / P.R. Lewis, P. Marrow, X. Yao // Autonomous Agents and Multi-Agent Systems. — 2010. — Vol. 21, № 2. — P. 143–171. doi: 10.1007/s10458-009-9113-x

2. Kravari, K. A survey of agent platforms / K. Kravari, N. Bassiliades // Journal of Artificial Societies and Social Simulation. — 2015. — Vol. 18, № 1. doi: 10.18564/jasss.2661

3. GridWay Metascheduler [Электронный ресурс]. — Режим доступа: http://www.gridway.org (дата обращения: 21.09.2020).

4. Tannenbaum, T. Condor – A Distributed Job Scheduler / T. Tannenbaum et al. / Eds. T.L. Sterling, W. Gropp, E. Lusk // Beowulf Cluster Computing with Linux. — MA: The MIT Press, 2002. — 350 p.

5. Костромин, Р.О. Организация управления распределенными вычислениями в интегрированной кластерной системе / Р.О. Костромин // Материалы XVI Всерос. конф. молодых ученых по математическому моделированию и информационным технологиям. — Новосибирск: ИВТ СО РАН, 2015. — С. 79–80.

6. Костромин, Р.О. Модели, методы и средства управления вычислениями в интегрированной кластерной системе / Р.О. Костромин // Фундаментальные исследования. — 2015. — № 6, ч. 1. — С. 35–38.

7. Костромин, Р.О. Разработка и применение предметно-ориентированных мультиагентных систем управления распределенными вычислениями / А.Г. Феоктистов, Р.О. Костромин // Известия ЮФУ. Технические науки. — 2016. — № 11. — С. 65–75.

8. Костромин, Р.О. Средства разработки и применения проблемно-ориентированных мультиагентных систем управления распределенными вычислениями / А.Г. Феоктистов, Р.О. Костромин // Материалы 4-й Всерос. науч.-техн. конф. «Суперкомпьютерные технологии». — Ростов н/Д: Изд-во ЮФУ,

2016. — Т. 2. — С. 108–112.

9. Костромин, Р.О. Мультиагентный алгоритм перераспределения вычислительных ресурсов для остаточной схемы решения задачи в Grid / А.Г. Феоктистов, Р.О. Костромин // Современные наукоемкие технологии. — 2016. — № 9, ч. 2. — С. 244–248.

10. Костромин, Р.О. Обзор мультиагентных систем управления масштабируемыми приложениями / А.Г. Феоктистов, Р.О. Костромин // Фундаментальные проблемы науки: Сб. статей Междунар. науч.-практ. конф. — Уфа: АЭТЕРНА, 2016. — Ч. 1. — С. 72–76.

11. Костромин, Р.О. Обзор инструментальных средств организации мультиагентных систем / А.Г. Феоктистов, Р.О. Костромин // Фундаментальные проблемы науки: Сб. статей Междунар. науч.-практ. конф. — Уфа: АЭТЕРНА, 2016. — Ч. 1. — С. 68–72.

12. Костромин, Р.О. Разработка систем управления вычислениями в интегрированной кластерной системе на основе свойств самоорганизующихся систем и мультиагентного подхода / Р.О. Костромин // Российская цивилизация: история, проблемы, перспективы: Материалы XV молодежной науч.-практ. конф. — Иркутск: Сибирский поворот, 2016. — С. 137–143.

13. Костромин, Р.О. Методы и программные средства автоматизации создания агентов управления распределенными вычислениями / Р.О. Костромин // Ляпуновские чтения: материалы конф. — Иркутск: ИДСТУ СО РАН, 2016. — С. 45–46.

14. Kostromin, R. Knowledge Elicitation in Multi-Agent System for Distributed Computing Management / A. Feoktistov, A. Tchernykh, S. Gorsky, R. Kostromin // Proceedings of the 40th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO-2017). — Riejka: IEEE, 2017. — P. 1350–1355.

15. Kostromin, R. Virtualization of Heterogeneous HPC-clusters Based on OpenStack Platform / A. Feoktistov, I. Sidorov, R. Kostromin, V. Sergeev,

V. Bogdanova // Вестник Южно-Уральского гос. ун-та. Сер. Вычислительная математика и информатика. — 2017. — Т. 6, № 2. — С. 37–48.

16. Kostromin, R. Job Flow Management for Virtualized Resources of Heterogeneous Distributed Computing Environment / I. Bychkov, A. Feoktistov, I. Sidorov, R. Kostromin // Procedia Engineering. — 2017. — Vol. 201. — P. 534–542.

17. Костромин, Р.О. Интеллектная технология управления вычислениями в виртуализированной кластерной среде / И.В. Бычков, А.Г. Феоктистов, И.А. Сидоров, Р.О. Костромин и др. // Материалы 10-й Всерос. мультиконф. — Ростов н/Д: Изд-во ЮФУ, 2017. — Т. 3. — С. 84–86.

18. Костромин, Р.О. Методы и средства извлечения знаний в мультиагентной системе управления распределенными вычислениями / Р.О. Костромин, А.Г. Феоктистов, Ю.А. Дядькин // Материалы 10-й Всерос. мультиконф. — Ростов н/Д: Изд-во Южного федерального ун-та, 2017. — Т. 3. — С. 117–119.

19. Костромин, Р.О. Мультиагентная система управления распределенными вычислениями / А.Г. Феоктистов, Р.О. Костромин // ИТНОУ: Информационные технологии в науке, образовании и управлении. — 2017. — № 4 (4). — С. 18–22.

20. Костромин, Р.О. Извлечение знаний агентами в системе управления распределенными вычислениями / А.Г. Феоктистов, Р.О. Костромин // Информационные и математические технологии в науке и управлении. — 2017. — № 3. — С. 136–143.

21. Костромин, Р.О. Извлечение знаний агентами в самоорганизующейся системе управления распределенными вычислениями / Р.О. Костромин // Материалы XVIII Всерос. конф. молодых ученых по математическому моделированию и информационным технологиям. — Новосибирск: ИВТ СО РАН, 2017. С. 78–79.

22. Kostromin, R. Integration of Heterogeneous HPC-clusters Using OpenStack Platform / A. Feoktistov, I. Sidorov, V. Sergeev, V. Bogdanova, R. Kostromin // Параллельные вычислительные технологии (ПАВТ’2017): Короткие статьи и описания плакатов XI Междунар. конф. — Челябинск: Издательский центр

ЮУрГУ, 2017. — С. 90–99.

23. Костромин, Р.О. Подход к автоматизации создания самоорганизующейся мультиагентной системы для управления вычислениями в интегрированной HPC-среде / Р.О. Костромин // Материалы XIII Всерос. конф. молодых ученых «Моделирование, оптимизация и информационные технологии». — Иркутск: ИДСТУ СО РАН, 2017. — С. 40.

24. Костромин, Р.О. Модель извлечения знаний в процессе мультиагентного управления распределенными вычислениями / А.Г. Феоктистов, Р.О. Костромин // Ляпуновские чтения: материалы конф. — Иркутск: ИДСТУ СО РАН, 2017. — С. 55.

25. Kostromin, R. Development of Distributed Subject-Oriented Applications for Cloud Computing through the Integration of Conceptual and Modular Programming / A. Feoktistov, R. Kostromin, I. Sidorov, S. Gorsky // Proceedings of the 41th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO-2018). — Riejka: IEEE, 2018. — P. 256–261.

26. Kostromin, R. Multi-Agent Approach for Dynamic Elasticity of Virtual Machines Provisioning in Heterogeneous Distributed Computing Environment / A. Feoktistov, I. Sidorov, A. Tchernykh, V. Zorkalzev, S. Gorsky, R. Kostromin, I. Bychkov, A. Avetisyan // Proceedings of the International Conference on High Performance Computing and Simulation (HPCS-2018). — IEEE, 2018. — P. 909–916.

27. Kostromin, R. Agent Behavior Model for Distributed Computing Management in the Environment with Virtualized Resources / A. Feoktistov, R. Kostromin, A. Chernykh // Proceedings of the 41th International Convention on information and communication technology, electronics and microelectronics (MIPRO-2018). — Riejka: IEEE, 2018. — P. 1153–1158.

28. Костромин, Р.О. Управление заданиями в гетерогенной распределенной вычислительной среде на основе знаний / А.Г. Феоктистов, Р.О. Костромин, Ю.А. Дядькин // Вестник компьютерных и информационных технологий. — 2018. — № 2. — С. 10–17.

29. Костромин, Р.О. Мультиагентный алгоритм построения остаточной

схемы решения задачи в распределенных пакетах прикладных программ / А.Г. Феоктистов, Р.О. Костромин, И.А. Сидоров, С.А. Горский // Известия ЮФУ. Технические науки. — 2018. — № 8. — С. 59–69.

30. Костромин, Р.О. Обучение агентов на основе параметрической настройки их алгоритмов управления распределенными вычислениями / И.В. Бычков, А.Г. Феоктистов, И.А. Сидоров, А.В. Еделев, С.А. Горский, Р.О. Костромин // Информационные технологии и нанотехнологии: Сб. тр. IV Междунар. конф. и молодежной школы (ИТНТ-2018). — Самара: Новая техника, 2018. — С. 2237–2247.

31. Kostromin, R. Orlando Tools: Development, Training, and Use of Scalable Applications in Heterogeneous Distributed Computing Environments / A. Tchernykh, A. Feoktistov, S. Gorsky, I. Sidorov, R. Kostromin, I. Bychkov, O. Basharina, A. Alexandrov, R. Rivera-Rodriguez // Communications in Computer Information Science. — 2019. — V. 979. — P. 265–279.

32. Kostromin, R. Multi-agent Algorithm for Re-allocating Grid-resources and Improving Fault-tolerance of Problem-solving Processes / A. Feoktistov, R. Kostromin, I. Sidorov, S. Gorsky, G. Oparin // Procedia Computer Science. — 2019. — V. 150. — P. 171–178.

33. Костромин, Р.О. Непрерывная интеграция функционального наполнения распределенных пакетов прикладных программ / А.Г. Феоктистов, С.А. Горский, И.А. Сидоров, Р.О. Костромин, Е.С. Фереферов // Параллельные вычислительные технологии: Короткие статьи и описания плакатов XIII междунар. конф. (ПАВТ’2019). — Челябинск: Издательский центр ЮУрГУ, 2019. — С. 464.

34. Kostromin, R. Machine Learning in a Multi-Agent System for Distributed Computing Management / I. Bychkov, A. Feoktistov, R. Kostromin, I. Sidorov, A. Edelev, S. Gorsky // Data Science Information Technology and Nanotechnology 2018: CEUR-WS Proceedings. — 2018. — V. 2212. — P. 89–97.

35. Костромин, Р.О. Алгоритм выбора лидера виртуального сообщества агентов / А.Г. Феоктистов, Р.О. Костромин // Ляпуновские чтения: материалы

конф. — Иркутск: ИДСТУ СО РАН, 2018. — С. 88–90.

36. Kostromin, R. Multi-Agent Algorithm for Re-Allocating Grid-Resources and Improving Fault-Tolerance of Problem-Solving Processes / A. Feoktistov, R. Kostromin et al. // Program and abstracts of XIII International Symposium «Intelligent Systems – 2018» (INTELS'18). — St. Petersburg: LETI, 2018. — P. 37–38.

37. Костромин, Р.О. Мультиагентный алгоритм построения остаточной схемы решения задачи в гетерогенной распределенной вычислительной среде / А.Г. Феоктистов, Р.О. Костромин, И.А. Сидоров, С.А. Горский // Материалы 5-й Всерос. науч.-техн. конф. «Суперкомпьютерные технологии». — Ростов н/Д: Изд-во Южного федерального ун-та, 2018. — Т. 2. — С. 71–75.

38. Костромин, Р.О. Сравнительный анализ работы метапланировщиков ресурсов в процессе выполнения схемы решения задачи / А.Г. Феоктистов, Р.О. Костромин // Ляпуновские чтения: материалы конф. — Иркутск: ИДСТУ СО РАН, 2019. — С. 85.

39. Библиотека алгоритмов для эффективного извлечения и применения проблемно-ориентированных знаний агентами: Свидетельство о государственной регистрации программы для ЭВМ от 11.12.2017 № 2017663706 / А.Г. Феоктистов, Р.О. Костромин. — М.: Федеральная служба по интеллектуальной собственности (РОСПАТЕНТ), 2017.

40. Программа мониторинга очередей заданий в гетерогенной распределенной вычислительной среде: Свидетельство о государственной регистрации программы для ЭВМ от 10.04.2018 № 2018616092 / А.Г. Феоктистов, Р.О. Костромин. — М.: Федеральная служба по интеллектуальной собственности (РОСПАТЕНТ), 2018.

41. Wooldridge, M. An Introduction to Multiagent Systems / M. Wooldridge. — Chichester: Wiley, 2002. — 340 p.

42. Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence / Ed. G. Weiss. — Cambridge, MA, USA: MIT Press, 1999. — 648 p.

43. Массель, Л.В. Разработка многоагентных систем распределенного

решения энергетических задач с использованием агентных сценариев / Л.В. Массель, В.И. Гальперов // Известия Томского политехнического университета, инжиниринг георесурсов. — 2015. — Т. 5. — С. 45–53.

44. Wooldridge, M. Agent-based software engineering / M. Wooldridge // IEEE Proceedings Software Engineering. — 1997. Vol. 144(1). P. 26–37.

45. Городецкий, В.И. Многоагентные системы: современное состояние исследований и перспективы применения / В.И. Городецкий // Новости искусственного интеллекта. — 1996. — С. 44–59.

46. Городецкий, В.И. Многоагентные системы: основные свойства и модели координации поведения / В.И. Городецкий // Информационные технологии и вычислительные системы. — 1998. — С. 22–34.

47. Nwana, H.S. Software agents: an overview / H.S. Nwana // The Knowledge Engineering Review. — Cambridge University Press, 1996. — Vol. 11, № 3. P. 205–244.

48. Russell, S.J. Artificial Intelligence: A Modern Approach / S.J. Russell, P. Norvig. — Pearson Education, 2003. — 1132 p.

49. Городецкий, В.И. Промышленные применения многоагентных систем: прогнозы и реалии / В.И. Городецкий, П.О. Скобелев, О.Л. Бухвалов // Проблемы управления и моделирования в сложных системах: труды XVIII Междунар. конф. — Самара: Изд-во ООО «Офорт», 2016. — С. 137–162.

50. Городецкий, В.И. Современное состояние и перспективы индустриальных применений многоагентных систем / В.И. Городецкий и др. // Управление техническими системами и технологическими процессами. — 2017. — С. 94–157.

51. Ковтуненко, А.С. Создание распределенных информационно-управляющих систем на базе агентно-ориентированного подхода / А.С. Ковтуненко, В.А. Масленников // Материалы XII всерос. совещания по проблемам управления ВСПУ-2014. — Институт проблем управления им. В.А. Трапезникова РАН, 2014. — С. 8984–8994.

52. Городецкий, В.И. Самоорганизация и многоагентные системы /

В.И. Городецкий // Известия РАН. Теория и системы управления. — 2012. — № 2. — С. 92–120.

53. Васильев, В.И. Интеллектуальные системы управления: теория и практика / В.И. Васильев, Б.Г. Ильясов. — Москва: Радиотехника, 2009. — 392 с.

54. Массель, Л.В. Проектирование и разработка многоагентной системы оценивания состояний электроэнергетических систем / Л.В. Массель, В.И. Гальперов // Вестник Иркутского государственного технического университета. — 2015. — Т. 10, № 105. — С. 27–33.

55. Unified Modeling Language (UML) [Электронный ресурс]. — 2019. Режим доступа: https://www.uml.org/ (дата обращения: 21.09.2020).

56. Java [Электронный ресурс]. Режим доступа: https://www.java.com (дата обращения: 21.09.2020).

57. Camazine, S. Self-Organization in Biological Systems / S. Camazine et al. — Princeton University Press: USA, 2001. — 562 p.

58. Di Marzo Serugendo, G. Self-organization in multi-agent systems / G. Di Marzo Serugendo, M.-P. Gleizes, A. Karageorgos // The Knowledge Engineering Review. — Cambridge University Press, USA, 2005. — Vol. 20, № 2. — P. 165–189.

59. Ковтуненко, А.С. Агентно-ориентированная программная архитектура распределенной обработки и хранения потоковых данных / А.С. Ковтуненко, В.А. Масленников, С.С. Валеев // Proceedings 2nd International Conference «Intelligent Technology Information Process. Manag. — 2014. — P. 65–70.

60. Frey, J. Condor-G: A Computation Management Agent for Multi-Institutional Grids / J. Frey et al. // Cluster Computing. — 2002. — Vol. 5. — P. 237–246.

61. YarKhan, A. GridSolve: The Evolution of a Network Enabled Solver / A. YarKhan, J. Dongarra, K. Seymour // Grid-based problem solving environment. — 2007. — Vol. 239. — P. 215–224.

62. Laxmi, C.V.T.E.V. Application Level Scheduling (AppLeS) in Grid with Quality of Service (QoS) / C.V.T.E.V. Laxmi, K. Somasundaram // Journal of Grid Computing. — 2014. — Vol. 5, № 2. — P. 1–10.

63. Shi, Z. Advanced Artificial Intelligence / Z. Shi. — Hackensack: World Scientific Publishing, 2011. — 613 p.

64. Rezaee, A. A Multi-Agent Architecture for QoS Support in Grid Environment / A. Rezaee // Journal of Computer Science. — 2008. — Vol. 4, № 3. — P. 225–231.

65. Singh, A. Agent Based Framework for Scalability in Cloud Computing / A. Singh, M. Malhotra // International Journal of Computer Science Engineering and Technology. — 2012. — Vol. 3, № 4. — P. 41–45.

66. Unland, R. Software Agent-Based Applications, Platforms and Development Kits / R. Unland, M. Klusch, M. Calisti // Birkhauser Verlag, 2005. — 455 p.

67. Finin, T. KQML as an agent communication language / T. Finin et al. // Proceedings of the Third International Conference on Information and Knowledge Management. — ACM, 1994. — P. 456.

68. Labrou, Y. Semantics and Conversations for an Agent Communication Language / Y. Labrou, T. Finin / Eds. M. Huhns, M. Singh // Readings in Agents. — San Mateo, 1998. — P. 235–242.

69. Badica, C. Software Agents: Languages, Tools, Platforms / C. Badica, E. Al. // ComSIS Consort. — 2011. — Vol. 8, № 2. — P. 255–298.

70. Kantamneni, A. Survey of multi-agent systems for microgrid control / A. Kantamneni et al. // Engineering Applications of Artificial Intelligence. — 2015. — Vol. 45. — P. 192–203.

71. Gupta, R. A Survey on Comparative Study of Mobile Agent Platforms / R. Gupta, G. Kansal // International Journal of Computer Science Engineering and Technology. — 2011. — Vol. 3(3). — P. 1943–1948.

72. Acronymics Inc. AgentBuilder [Электронный ресурс]. Режим доступа: https://www.agentbuilder.com (дата обращения: 21.09.2020).

73. Delft University of Technology. AgentScape [Электронный ресурс]. Режим доступа: http://www.agentscape.de (дата обращения: 21.09.2020).

74. Raytheon BBN Technologies. Cognitive Agent Architecture [Электронный

ресурс]. Режим доступа: http://www.cougaar.world/ (дата обращения: 21.09.2020).

75. Bhamra, G.S. Intelligent Software Agent Technology: An Overview / G.S. Bhamra, A.K. Verma, R.B. Patel // International Journal of Computer Applications. — 2014. — Vol. 89, No. 2. — P. 19–31.

76. Kravari, K. EMERALD: A multi-agent system for knowledge-based reasoning interoperability in the semantic web / K. Kravari, E. Kontopoulos, N. Bassiliades // Lecture Notes in Computer Science. — 2010. — Vol. 6040. — P. 173–182.

77. Grignard, A. GAMA 1.6: Advancing the Art of Complex Agent-Based Modeling and Simulation / A. Grignard et al. // The 16th International Conference on Principles and Practices in Multi-Agent Systems (PRIMA). — 2013. — Vol. 8291. — P. 242–258.

78. Telecom Italia. Java Agent DEvelopment Framework [Электронный ресурс]. — Режим доступа: https://jade.tilab.com/ (дата обращения: 21.09.2020).

79. University H. JADEx [Электронный ресурс]. — Режим доступа: https://www.activecomponents.org/ (дата обращения: 21.09.2020).

80. MaDKit – Multiagent Development Kit [Электронный ресурс]. — Режим доступа: http://www.madkit.net (дата обращения: 21.09.2020).

81. Bellifemine, F. Jade – A Java Agent Development Framework / F. Bellifemine et al. // Multi-Agent Programming. — 2005. — Vol. 15. — P. 125–147.

82. Braubach, L. Jadex: A BDI agent system combining middleware and reasoning / L. Braubach, A. Pokahr, W. Lamersdorf / Eds. R. Unland, M. Klusch, M. Calisti // Software Agent-Based Applications, Platforms and Development Kits. — Basel-Boston-Berlin: Birkhauser-Verlag, 2005. — P. 143–168.

83. Богданова, В.Г. Мультиагентный подход к управлению распределенными вычислениями в кластерной Grid-системе / В.Г. Богданова и др. // Известия Российской академии наук. Теория и системы управления. — 2014. — № 5. — С. 95–105.

84. ЦКП Иркутский суперкомпьютерный центр СО РАН [Электронный ресурс]. — Режим доступа: http://hpc.icc.ru/ (дата обращения: 21.09.2020).

85. Bychkov, I. Conceptual Model of Problem-oriented Heterogeneous Distributed Computing Environment with Multi-agent Management / I. Bychkov et al. // Procedia Computer Science. — Elsevier, 2017. — Vol. 103. — P. 162–167.

86. Bychkov, I. Agent-based approach to monitoring and control of distributed computing environment / I. Bychkov et al. // Lecture Notes in Computer Science. — Springer, Cham, 2015. — Vol. 9251. — P. 253–257.

87. Опарин, Г.А. Инструментальная распределенная вычислительная САТУРН-среда / Г.А. Опарин, А.Г. Феоктистов // Программные продукты и системы. — 2002. — № 2. — С. 27–30.

88. Опарин, Г.А. Планирование схем решения задач в инструментальном комплексе САТУРН/ПЗ / Г.А. Опарин, Д.Г. Феоктистов // Компьютерная логика, алгебра и интеллектное управление: Тр. Всерос. школы. — Иркутск: Изд-во ИрВЦ СО РАН, 1994. — Т. 1. — С. 5–13.

89. Поликарпова, Н.И. Автоматное программирование / Н.И. Поликарпова, А.А. Шалыто. — СПб.: Питер, 2008. — 168 с.

90. Бычков, И. Мультиагентный алгоритм распределения вычислительных ресурсов на основе экономического механизма регулирования их спроса и предложения / И. Бычков и др. // Вестник компьютерных и информационных технологий. — 2014. — № 1. — С. 39–45.

91. Introducing JSON [Электронный ресурс]. — Режим доступа: https://www.json.org/json-en.html (дата обращения: 21.09.2020).

92. Tel, G. Introduction To Distributed Algorithms / G. Tel. — Cambridge University Press, 2000. — 596 p.

93. Бондаренко, А.А. Обеспечение отказоустойчивости высокопроизводительных вычислений с помощью локальных контрольных точек / А.А. Бондаренко, М.В. Якобовский // Вестник ЮУГУ. Сер. Вычислительная математика и информатика. — 2014. — Т. 3, № 3. С. 20–36.

94. Kumar Mishra, M. A Survey on Scheduling Heuristics in Grid Computing Environment / M. Kumar Mishra et al. // International Journal of Modern Education and

Computer Science. — 2014. — Vol. 6, № 10. — P. 57–83.

95. Qureshi, M.B. Survey on Grid Resource Allocation Mechanisms / M.B. Qureshi et al. // Journal of Grid Computing. — 2014. — Vol. 12, № 2. — P. 399–441.

96. Ershov, A.P. On Mixed Computation: Informal Account of the Strict and Polyvariant Computation Schemes / A.P. Ershov // Control Flow and Data Flow: Concepts of Distributed Programming. — Berlin A.O.: Springer-Verlag, 1985. — P. 107–120.

97. Balaji, P. Fault Tolerance Techniques for Scalable Computing / P. Balaji, D. Buntinas, D. Kimpe / Eds. L. Wang, A.Y. Zomaya, S.U. Khan // Scalable Computer Communication Theory and Practice. — Hoboken: Wiley-IEEE Press, 2013. — P. 212–245.

98. Feoktistov, A.G. Logical-probabilistic analysis of distributed computing reliability / A.G. Feoktistov, I.A. Sidorov // Proceedings of the 39th International Convention on information and communication technology, electronics and microelectronics (MIPRO-2016). — Riejka: IEEE, 2016. — P. 247–252.

99. Феоктистов, А.Г. Методология концептуализации и классификации потоков заданий масштабируемых приложений в разнородной распределенной вычислительной среде / А.Г. Феоктистов // Системы управления, связи и безопасности. — Общество с ограниченной ответственностью «Корпорация Интел Групп», 2015. — № 4. — С. 1–25.

100. Шоломов, Л.А. Логические методы исследования дискретных моделей выбора / Л.А. Шоломов. — М.: Наука, 1989. — 288 с.

101. Goh, S.K. JADE-FSM-Engine: A Deployment Tool for Flexible Agent Behaviours in JADE / S.K. Goh, M.B. Chhetri, R. Kowalczyk // International Conference on Intelligent Agent Technology (IAT’07). — IEEE, 2007. — P. 524–527.

102. Kessler, R. A Hierarchical State Machine using JADE Behaviours with Animation Visualization / R. Kessler et al. // — Technical report, University of Utah, 2004.

103. Ковтуненко, А.С. Многоагентная платформа распределенной обработки данных реального времени / А.С. Ковтуненко, С.С. Валеев, В.А. Масленников // Естественные и технические науки. — 2013. — Т. 2 (64). — С. 311–313.

104. Официальный сайт Т-Платформы [Электронный ресурс]. — Режим доступа: https://www.t-platforms.ru (дата обращения: 21.09.2020).

105. Citrix XenServer [Электронный ресурс]. — Режим доступа: https://www.citrix.com/ru-ru/products/citrix-hypervisor/ (дата обращения: 07.04.2019).

106. Qemu/KVM [Электронный ресурс]. — Режим доступа: https://www.qemu.org/ (дата обращения: 07.04.2019).

107. Oracle VM VirtualBox [Электронный ресурс]. — Режим доступа: https://www.virtualbox.org/ (дата обращения: 21.09.2020).

108. Bumgardner, V.K.C. OpenStack in action / V.K.C Bumgardner. — Manning Publications, 2016. — 384 p.

109. HTCondor [Электронный ресурс]. — Режим доступа: http://research.cs.wisc.edu/htcondor (дата обращения: 21.09.2020).

110. TORQUE Resource Manager [Электронный ресурс]. — Режим доступа: https://adaptivecomputing.com/cherry-services/torque-resource-manager/ (дата обращения: 21.09.2020).

111. Message Passing Interface (MPI) [Электронный ресурс]. — Режим доступа: https://www.open-mpi.org/ (дата обращения: 21.09.2020).

112. The OpenMP API specification for parallel programming [Электронный ресурс]. — Режим доступа: https://www.openmp.org/ (дата обращения: 21.09.2020).

113. Опарин, Г.А. Инструментальный комплекс ORLANDO TOOLS / Г.А. Опарин и др. // Программные продукты и системы. — 2007. — № 4. — С. 63–65.

114. Миньков, С.А. Мультиагентная среда для управления процессом распределенных вычислений / С.А. Миньков // Вестник ТГУ. Приложение. — 2006. — № 17. — С. 241–245.

115. Скобелев, П.О. Мультиагентные системы для управления ресурсами

предприятий в реальном времени [Электронный ресурс]. — 2011. Режим доступа: http://www.csedays.ru/application2011/program/skobelev (дата обращения: 21.09.2020).

116. Еделев, А.В. Применение распределенных вычислений для выявления критически важных объектов газотранспортной сети России / А.В. Еделев, С.М. Сендеров, И.А. Сидоров // Информационные и математические технологии в науке и управлении. — 2016. — № 1. — С. 55–62.

117. Edelev, A.V. The Approach to Find Rational Energy Development Ways in Terms of Energy Security Requirements / A.V. Edelev et al. // Proceedings of the International Conference of Science and Technology: 50th Anniversary of Electric Power University. — Hanoi, Vietnam, 2016. — P. 548–556.

118. Bychkov, I. Simulation Modeling in Heterogeneous Distributed Computing Environments to Support Decisions Making in Warehouse Logistics / I. Bychkov et al. // Procedia Engineering. — 2017. — Vol. 201. — P. 524–533.

119. Chapin, S.J. Benchmarks and Standards for the Evaluation of Parallel Job Schedulers / S.J. Chapin et al. // Job Scheduling Strategies for Parallel Processing. — Berlin: Springer, 1999. — Vol. 1659. — P. 66–89.

120. Belhajjame, K. A flexible workflow model for process-oriented applications / K. Belhajjame, G. Vargas-Solar, C. Collet // Proceedings of the Second International Conference on Web Information Systems Engineering. — IEEE Computer Soc, 2001. — Vol. 1, № 1. — P. 72–80.

121. Jacob, J.C. Montage: a grid portal and software toolkit for science-grade astronomical image mosaicking / J.C. Jacob et al. // International Journal Computer Science Engineering. — 2010. — Vol. 4, № 2. — P. 73–87.

122. Mathog, D.R. Parallel BLAST on split databases / D.R. Mathog // Bioinformatics. — 2003. — № 19 (14). — P. 1865–1866.

123. SCEC project, Southern California Earthquake Center [Электронный ресурс]. — Режим доступа: http://scec.usc.edu/scecpedia/Broadband_Platform (дата обращения: 21.09.2020).

124. Сидоров, И.А. Технология организации распределенных вычислений в инструментальном комплексе Discomp / И.А. Сидоров, Г.А. Опарин, А.Г. Феоктистов // Современные технологии. Системный анализ. Моделирование. — 2009. — № 2. — С. 175–180.

125. Поспелов, Д.А. От моделей коллективного поведения к многоагентным системам / Д.А. Поспелов // Программные продукты и системы. 2003. № 2. С. 39–44.

Приложение А

МИНОБРНАУКИ РОССИИ
федеральное государственное бюджетное
образовательное учреждение
высшего образования
«Иркутский государственный университет»
(ФГБОУ ВО «ИГУ»)
МЕЖДУНАРОДНЫЙ ИНСТИТУТ
ЭКОНОМИКИ И ЛИНГВИСТИКИ
(МИЭЛ)
ул. Улан-Баторская, д.6, г. Иркутск, 664082
Тел.: (3952) 52-11-40 Факс: (3952) 52-11-42
ОКПО 02068226, ОГРН 1033801008218,
ИНН/КПП 3808013278/380801001
www.id.isu.ru, e-mail: director@miel.isu.ru
06.05.2019 № 11-054

СПРАВКА

об использовании результата интеллектуальной деятельности
для автоматизации проведения научных исследований

Результат интеллектуальной деятельности «Библиотека алгоритмов для эффективного извлечения и применения проблемно-ориентированных знаний агентами» (Свидетельство об официальной регистрации программы для ЭВМ № 2017663706, авторы: Феоктистов А.Г., Костромин Р.О.) успешно использован в Международном институте экономики и лингвистики федерального государственного бюджетного образовательного учреждения высшего образования «Иркутский государственный университет» (МИЭЛ ФГБОУ ВО «ИГУ») в рамках НИР № 111-15-701 «Разработка и внедрение комплексов автоматизированного анализа и систематизации научных данных» (2017-2019 гг., научный руководитель к.х.н., проф. Дмитриев В.И.).

Использование данного результата интеллектуальной деятельности позволило повысить эффективность проведения практических экспериментов с использованием разнородной распределенной вычислительной среды и ускорило получение основных результатов НИР.

Директор МИЭЛ ИГУ, профессор О.В. Архипкин

Приложение Б

Свидетельства об официальной регистрации программ для ЭВМ

РОССИЙСКАЯ ФЕДЕРАЦИЯ

СВИДЕТЕЛЬСТВО

о государственной регистрации программы для ЭВМ

№ 2017663706

«Библиотека алгоритмов для эффективного извлечения и применения проблемно-ориентированных знаний агентами»

Правообладатель: ***Федеральное государственное бюджетное учреждение науки Институт динамики систем и теории управления имени В.М. Матросова Сибирского отделения Российской академии наук (RU)***

Авторы: ***Феоктистов Александр Геннадьевич (RU), Костромин Роман Олегович (RU)***

Заявка № **2017660709**
Дата поступления **24 октября 2017 г.**
Дата государственной регистрации
в Реестре программ для ЭВМ ***11 декабря 2017 г.***

Руководитель Федеральной службы по интеллектуальной собственности

Г.П. Ивлиев

РОССИЙСКАЯ ФЕДЕРАЦИЯ

СВИДЕТЕЛЬСТВО

о государственной регистрации программы для ЭВМ

№ 2018616092

«Программа мониторинга очередей заданий в гетерогенной распределенной вычислительной среде»

Правообладатель: ***Федеральное государственное бюджетное учреждение науки Институт динамики систем и теории управления имени В.М. Матросова Сибирского отделения Российской академии наук (RU)***

Авторы: ***Феоктистов Александр Геннадьевич (RU), Костромин Роман Олегович (RU)***

Заявка № **2018613483**
Дата поступления **10 апреля 2018 г.**
Дата государственной регистрации
в Реестре программ для ЭВМ ***23 мая 2018 г.***

Руководитель Федеральной службы по интеллектуальной собственности

Г.П. Ивлиев

Приложение В

Инструментарии для создания мультиагентных систем

Таблица В.1 – Общие характеристики мультиагентных платформ

Платформа	Назначение	Текущее состояние	Наличие бесплатной версии	Поддержка пользователей
Agent Factory	Агенты общего назначения	Развивается	+	Хорошая (документация, форум, e-mail рассылка)
AgentBuilder	Мультиагентные системы общего назначения	Проект завершен и не развивается	–	Хорошая (документация, перечень популярных вопросов, шаблоны и примеры, отчеты об ошибках, e-mail рассылка)
AgentScape	Крупномасштабные распределенные мультиагентные системы	Проект завершен и не развивается	+	Хорошая (форум, документация, e-mail рассылка)
AGLOBE	Моделирование в реальном времени	Проект завершен и не развивается	+	Средняя (документация, почта для связи)
AnyLogic	Общего назначения, распределенное агентное моделирование	Развивается	–	Высокая (расширенная документация, средства поддержки)
Cormas	Природные ресурсы и агентное моделирование	Развивается	+	Хорошая (форум, почта документация)

Платформа	Назначение	Текущее состояние	Наличие бесплатной версии	Поддержка пользователей
Cougaar	Крупномасштабные распределенные приложения	Развивается	+	Хорошая (перечень популярных вопросов, форум, документация, e-mail рассылка)
CybelePro	Крупномасштабные распределенные системы	Развивается	–	Хорошая (поддержка при покупке)
EMERALD	Распределенные приложения состоящие из автономных сущностей	Развивается	+	Средняя (документация, почта для связи)
GAMA	Крупномасштабное распределенное агентное моделирование	Развивается	+	Хорошая (форум, e-mail рассылка, документация, wiki, социальные сети)
INGENIAS Development Kit	Агенты общего назначения	Развивается	+	Средняя (документация, почта для связи)
JACK	Динамические и сложные среды	Проект завершен и не развивается	+	Высокая (расширенная документация, средства поддержки)
JADE	Распределенные приложения состоящие из автономных сущностей	Развивается	+	Высокая (перечень популярных вопросов, e-mail рассылка, документация,

Платформа	Назначение	Текущее состояние	Наличие бесплатной версии	Поддержка пользователей
				API, отчеты об ошибках)
Jadex	Распределенные приложения состоящие из автономных BDI-сущностей	Развивается	+	Средняя (документация, почта для связи)
JAMES II	Общего назначения, распределенное агентное моделирование	Развивается	+	Средняя (документация, почта для связи, wiki)
JAS	Агенты общего назначения	Проект завершен и не развивается	+	Хорошая (API, документация, инструкции, почта разработчиков)
Jason	Распределенные приложения состоящие из автономных BDI-сущностей	Развивается	+	Высокая (подробные инструкции, книги, API, новостная рассылка
JIAC	Крупномасштабные распределенные системы	Развивается	+	Хорошая (перечень популярных вопросов, документация, e-mail рассылка)
MaDKit	Агентное моделирование для мультиагентных систем	Развивается	+	Хорошая (документация, форум, почта для связи)
MASON	Событийно-ориентированное мультиагентное моделирование	Развивается	+	Средняя (документация, почта для связи)

Платформа	Назначение	Текущее состояние	Наличие бесплатной версии	Поддержка пользователей
NetLogo	Агентное моделирование	Развивается	+	Хорошая (перечень популярных вопросов, документация, почта для связи, инструкции, примеры)
Repast	Агентное моделирование	Развивается	+	Средняя (документация, почта для связи)
SeSAm	Агентное моделирование	Развивается	+	Хорошая (документация, wiki)
Swarm	Агенты общего назначения	Проект завершен и не развивается	+	Хорошая (документация, wiki, e-mail рассылка)

Таблица В.2 – Пользовательские характеристики мультиагентных платформ

Платформа	Поддерживаемые стандарты	Протокол обмена сообщениями	Средства разработки	Лицензия
Agent Factory	Частично FIPA	HTTP	Java, AFAPL, AgentSpeak	LGPL
AgentBuilder	KQML	KQML, CORBA, TCP/IP	KQML, Java, C,C++	Проприетарная, скидки для ученых
AgentScape	Внутренний стандарт платформы	Внутренний язык платформы	Java, XML	BSD
AGLOBE	Частично FIPA, GIS	ACL	Java	LGPL

Платформа	Поддерживаемые стандарты	Протокол обмена сообщениями	Средства разработки	Лицензия
AnyLogic	GIS, 3D	Внутренний язык платформы	Java, UML-RT	Коммерческая, академическая
Cormas	Внутренний стандарт платформы	P2P	SmallTalk	GPL
Cougaar	Внутренний стандарт платформы	Cougaar Message Transport Service	Java	Cougaar Open Source License (COSL)
CybelePro	Внутренний стандарт платформы	Внутренний язык платформы	Java	Коммерческая, академическая
EMERALD	FIPA, Semantic web standards	Асинхронный ACL	Java, JESS, RuleML, Prolog, XML, RDF	LGPL
GAMA	FIPA, GIS, 3D	ACL	GAML	GPL
INGENIAS Development Kit	AUML	P2P	Java, XML	CC By-SA GPLv2
JACK	FIPA	TCP/IP	Java, JACK Agent Language (JAL), XML	Коммерческая, академическая
JADE	FIPA, CORBA	Асинхронный ACL, MTPs, RMI, IIOP, HTTP, WAP	Java	LGPLv2
Jadex	FIPA, SOA, WSDL	HTTP	Java, XML	LGPLv2
JAMES II	Внутренний стандарт платформы	Зависит от плагинов	Java	JAMESLIC (совместимая с GPL)
JAS	Частично FIPA,	Внутренний	Java	LGPL

Платформа	Поддерживаемые стандарты	Протокол обмена сообщениями	Средства разработки	Лицензия
	GraphML, XML	язык платформы		
Jason	Частично FIPA	KQML	Java, AgentSpeak	LGPLv2
JIAC	Внутренний стандарт платформы	ActiveMQ	Java, XML	Apache License V2
MaDKit	UML	P2P	Java, C/C++, Python	GPL
MASON	Внутренний стандарт платформы	Внутренний язык платформы	Java	Свободная академическая
NetLogo	Внутренний стандарт платформы	Внутренний язык платформы	NetLogo	GPL
Repast	Внутренний стандарт платформы	P2P	Java, C#, C++, Lisp, Prolog, Python	New BSD
SeSAm	Внутренний стандарт платформы	Внутренний язык платформы	Java (plus plugin for ontologies)	LGPL
Swarm	Внутренний стандарт платформы	Внутренний язык платформы	Java	GPL

Таблица В.3 – Функциональные характеристики мультиагентных платформ

Платформа	Совместимые ОС	Производи-тельность	Масштаби-руемость	Популяр-ность
Agent Factory	Любая с JVM	Хорошая	Хорошая	Низкая
AgentBuilder	Windows, Linux, Sun Solaris	Хорошая	Хорошая	Средняя

Платформа	Совместимые ОС	Производи-тельность	Масштаби-руемость	Популяр-ность
AgentScape	Любая с JVM	Хорошая	Хорошая	Низкая
AGLOBE	Любая с JVM	Высокая	Высокая	Средняя
AnyLogic	Любая с JVM	Высокая	Высокая	Средняя
Cormas	Windows, Linux	Высокая	Хорошая	Средняя
Cougaar	Windows, Linux	Высокая	Высокая	Низкая
CybelePro	Любая ОС с JVM	Высокая	Высокая	Низкая
EMERALD	Любая с JVM	Высокая	Высокая	Низкая
GAMA	Windows, Linux, Mac OS	Хорошая	Хорошая	Низкая
INGENIAS Development Kit	Любая ОС с JVM	Хорошая	Хорошая	Средняя
JACK	Windows, Macintosh, Unix, generic Java, iPAQ	Высокая	Высокая	Высокая
JADE	Любая с JVM	Высокая	Высокая	Высокая
Jadex	Любая ОС с JVM	Высокая	Высокая	Высокая
JAMES II	Любая ОС с JVM	Высокая	Высокая	Средняя
JAS	Любая ОС с JVM	Хорошая	Средняя	Средняя
Jason	Windows, Linux, Mac OS	Хорошая	Хорошая	Высокая
JIAC	Любая ОС с JVM	Высокая	Высокая	Низкая
MaDKit	Любая ОС с JVM	Хорошая	Хорошая	Средняя
MASON	Windows	Хорошая	Средняя	Средняя
NetLogo	Любая ОС с JVM	Хорошая	Хорошая	Высокая
Repast	Любая ОС с JVM	Высокая	Хорошая	Средняя
SeSAm	Любая ОС с JVM	Хорошая	Хорошая	Средняя
Swarm	Windows, Linux	Средняя	Средняя	Средняя

Приложение Г

Таблицы истинности для функций, определяющих три или более переходов из одного состояния

Таблица Г.1. Таблица истинности функций переходов из состояния AS_{start} родительского автомата для агента формулировки постановки задачи и построения плана ее решения

q_{11}	q_3	q_2	q_1	f_1	f_2	f_3	f_{16}	f_7
0	0	0	0	0	0	0	0	1
0	0	0	1	0	1	0	0	0
0	0	1	0	1	0	0	0	0
0	0	1	1	0	1	0	0	0
0	1	0	0	0	0	1	0	0
0	1	0	1	0	1	0	0	0
0	1	1	0	1	0	0	0	0
0	1	1	1	0	1	0	0	0
1	0	0	0	0	0	0	1	0
1	0	0	1	0	1	0	0	0
1	0	1	0	1	0	0	0	0
1	0	1	1	0	1	0	0	0
1	1	0	0	0	0	1	0	0
1	1	0	1	0	1	0	0	0
1	1	1	0	1	0	0	0	0
1	1	1	1	0	1	0	0	0

Таблица Г.2. Таблица истинности функций переходов из состояния AS_{start} родительского автомата для агента классификации заданий

q_3	q_2	q_1	f_1	f_2	f_3	f_5
0	0	0	0	0	0	1
0	0	1	0	1	0	0
0	1	0	1	0	0	0
0	1	1	0	1	0	0
1	0	0	0	0	1	0
1	0	1	0	1	0	0
1	1	0	1	0	0	0
1	1	1	0	1	0	0

Таблица Г.3. Таблица истинности функций переходов из состояния AS_{start} родительского автомата для агента организации виртуального сообщества

q_{18}	q_3	q_2	q_1	f_1	f_2	f_3	f_{23}	f_6
0	0	0	0	0	0	0	0	1
0	0	0	1	0	1	0	0	0
0	0	1	0	1	0	0	0	0
0	0	1	1	0	1	0	0	0
0	1	0	0	0	0	1	0	0
0	1	0	1	0	1	0	0	0
0	1	1	0	1	0	0	0	0
0	1	1	1	0	1	0	0	0
1	0	0	0	0	0	0	1	0
1	0	0	1	0	1	0	0	0
1	0	1	0	1	0	0	0	0
1	0	1	1	0	1	0	0	0
1	1	0	0	0	0	1	0	0
1	1	0	1	0	1	0	0	0
1	1	1	0	1	0	0	0	0
1	1	1	1	0	1	0	0	0

Таблица Г.4. Таблица истинности функций переходов из состояния AS_{start} родительского автомата для агента, представляющего ресурсы среды

q_{23}	q_{12}	q_3	q_2	q_1	f_1	f_2	f_3	f_{18}	f_{17}	f_9
0	0	0	0	0	0	0	0	0	0	1
0	0	0	0	1	0	1	0	0	0	0
0	0	0	1	0	1	0	0	0	0	0
0	0	0	1	1	0	1	0	0	0	0
0	0	1	0	0	0	0	1	0	0	0
0	0	1	0	1	0	1	0	0	0	0
0	0	1	1	0	1	0	0	0	0	0
0	0	1	1	1	0	1	0	0	0	0
0	1	0	0	0	0	0	0	1	0	0
0	1	0	0	1	0	1	0	0	0	0
0	1	0	1	0	1	0	0	0	0	0
0	1	0	1	1	0	1	0	0	0	0
0	1	1	0	0	0	0	1	0	0	0
0	1	1	0	1	0	1	0	0	0	0
0	1	1	1	0	1	0	0	0	0	0
0	1	1	1	1	0	1	0	0	0	0
1	0	0	0	0	0	0	0	0	1	0
1	0	0	0	1	0	1	0	0	0	0

q_{23}	q_{12}	q_3	q_2	q_1	f_1	f_2	f_3	f_{18}	f_{17}	f_9
1	0	0	1	0	1	0	0	0	0	0
1	0	0	1	1	0	1	0	0	0	0
1	0	1	0	0	0	0	1	0	0	0
1	0	1	0	1	0	1	0	0	0	0
1	0	1	1	0	1	0	0	0	0	0
1	0	1	1	1	0	1	0	0	0	0
1	1	0	0	0	0	0	0	1	0	0
1	1	0	0	1	0	1	0	0	0	0
1	1	0	1	0	1	0	0	0	0	0
1	1	0	1	1	0	1	0	0	0	0
1	1	1	0	0	0	0	1	0	0	0
1	1	1	0	1	0	1	0	0	0	0
1	1	1	1	0	1	0	0	0	0	0
1	1	1	1	1	0	1	0	0	0	0

Таблица Г.5. Таблица истинности функций переходов из состояния AS_2 родительского автомата для агента, представляющего ресурсы среды

q_{21}	q_5	f_{36}	f_{31}	f_{19}
0	0	1	0	0
0	1	0	1	0
1	0	0	0	1
1	1	0	1	0
0	0	1	0	0
0	1	0	1	0
1	0	0	0	1
1	1	0	1	0

Таблица Г.6. Таблица истинности функций переходов из состояния AS_{start} дочернего автомата для агента, представляющего ресурсы среды и функционирующего в роли координатора виртуального сообщества

q_{25}	q_8	q_3	q_2	q_1	f_1	f_2	f_3	f_{29}	f_{28}	f_8
0	0	0	0	0	0	0	0	0	0	1
0	0	0	0	1	0	1	0	0	0	0
0	0	0	1	0	1	0	0	0	0	0
0	0	0	1	1	0	1	0	0	0	0
0	0	1	0	0	0	0	1	0	0	0
0	0	1	0	1	0	1	0	0	0	0
0	0	1	1	0	1	0	0	0	0	0

q_{25}	q_8	q_3	q_2	q_1	f_1	f_2	f_3	f_{29}	f_{28}	f_8
0	0	1	1	1	0	1	0	0	0	0
0	1	0	0	0	0	0	0	1	0	0
0	1	0	0	1	0	1	0	0	0	0
0	1	0	1	0	1	0	0	0	0	0
0	1	0	1	1	0	1	0	0	0	0
0	1	1	0	0	0	0	1	0	0	0
0	1	1	0	1	0	1	0	0	0	0
0	1	1	1	0	1	0	0	0	0	0
0	1	1	1	1	0	1	0	0	0	0
1	0	0	0	0	0	0	0	0	1	0
1	0	0	0	1	0	1	0	0	0	0
1	0	0	1	0	1	0	0	0	0	0
1	0	0	1	1	0	1	0	0	0	0
1	0	1	0	0	0	0	1	0	0	0
1	0	1	0	1	0	1	0	0	0	0
1	0	1	1	0	1	0	0	0	0	0
1	0	1	1	1	0	1	0	0	0	0
1	1	0	0	0	0	0	0	0	1	0
1	1	0	0	1	0	1	0	0	0	0
1	1	0	1	0	1	0	0	0	0	0
1	1	0	1	1	0	1	0	0	0	0
1	1	1	0	0	0	0	1	0	0	0
1	1	1	0	1	0	1	0	0	0	0
1	1	1	1	0	1	0	0	0	0	0
1	1	1	1	1	0	1	0	0	0	0

Таблица Г.7. Таблица истинности функций переходов из состояния AS_{start} дочернего автомата для агента, представляющего ресурсы среды и функционирующего в роли ресурсного агента

q_{17}	q_{15}	q_3	q_2	q_1	f_1	f_2	f_3	f_{34}	f_{33}	f_{32}
0	0	0	0	0	0	0	0	1	0	0
0	0	0	0	1	0	1	0	0	0	0
0	0	0	1	0	1	0	0	0	0	0
0	0	0	1	1	0	1	0	0	0	0
0	0	1	0	0	0	0	1	0	0	0
0	0	1	0	1	0	1	0	0	0	0
0	0	1	1	0	1	0	0	0	0	0
0	0	1	1	1	0	1	0	0	0	0
0	1	0	0	0	0	0	0	0	0	1
0	1	0	0	1	0	1	0	0	0	0

q_{17}	q_{15}	q_3	q_2	q_1	f_1	f_2	f_3	f_{34}	f_{33}	f_{32}
0	1	0	1	0	1	0	0	0	0	0
0	1	0	1	1	0	1	0	0	0	0
0	1	1	0	0	0	0	1	0	0	0
0	1	1	0	1	0	1	0	0	0	0
0	1	1	1	0	1	0	0	0	0	0
0	1	1	1	1	0	1	0	0	0	0
1	0	0	0	0	0	0	0	0	1	0
1	0	0	0	1	0	1	0	0	0	0
1	0	0	1	0	1	0	0	0	0	0
1	0	0	1	1	0	1	0	0	0	0
1	0	1	0	0	0	0	1	0	0	0
1	0	1	0	1	0	1	0	0	0	0
1	0	1	1	0	1	0	0	0	0	0
1	0	1	1	1	0	1	0	0	0	0
1	1	0	0	0	0	0	0	0	1	0
1	1	0	0	1	0	1	0	0	0	0
1	1	0	1	0	1	0	0	0	0	0
1	1	0	1	1	0	1	0	0	0	0
1	1	1	0	0	0	0	1	0	0	0
1	1	1	0	1	0	1	0	0	0	0
1	1	1	1	0	1	0	0	0	0	0
1	1	1	1	1	0	1	0	0	0	0

Приложение Д

Пример развертывания агентной платформы JADE и подключения к ней агентов

Платформа JADE запускается на одном из узлов РВС. Затем к ней подключаются агенты. На одном узле допустимо запускать несколько платформ при условии, что они работают на разных портах (по умолчанию используется порт 1099). Агенты для внутреплатформенных коммуникаций занимают порт 1875.

Для начала работы с JADE необходимо распаковать в рабочий каталог архив с актуальной версией JADE, загруженной с официального сайта [78]. На узле обязательно наличие установленной среды выполнения Java.

Чтобы запустить платформу JADE вместе с графическим интерфейсом (рисунок Д.1), который удобен для отладки работы МАС и управления агентами, необходимо выполнить следующую команду:

```
java -cp lib\jade.jar jade.Boot -gui -platform-id
MASPlatform1 -port 1100
```

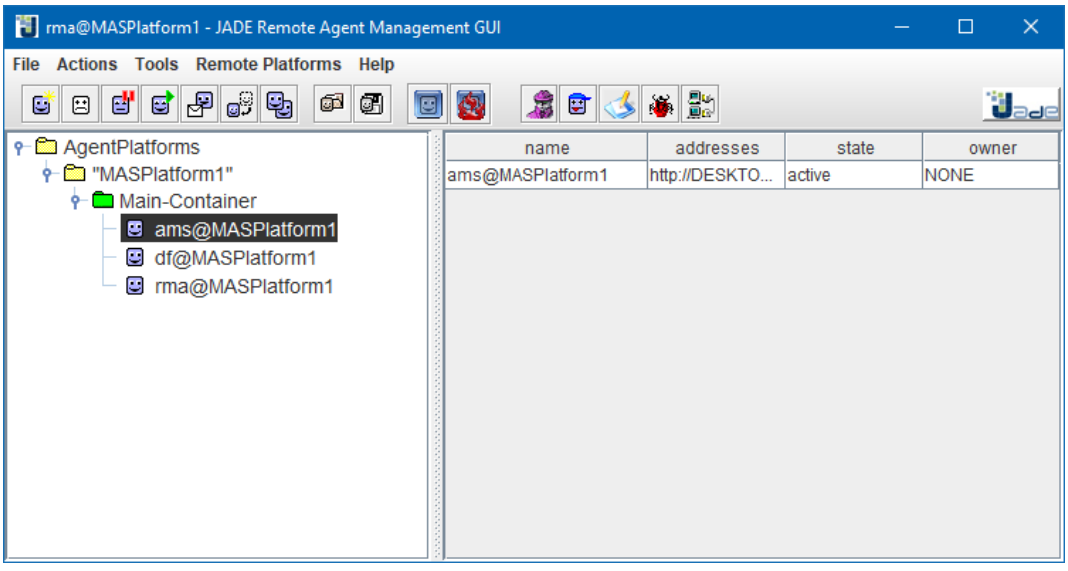


Рисунок Д.1 – Графический интерфейс системы управления агентами JADE

В результате ее выполнения создастся платформа с именем «MASPlatform1» и главный контейнер, который содержит набор базовых агентов и принимает

запросы через порт 1100, а также появится окно управления агентами.

В том случае, если возникает потребность в запуске дополнительного контейнера в созданной платформе, нужно выполнить команду:

```
java -cp lib\jade.jar jade.Boot -name PlatformName -
container -host 10.10.1.12 -port 1100
```

Параметры «host» и «port» не обязательные, с их помощью можно указать узел, на котором расположена платформа и ее порт. Важно заметить, что главный контейнер может быть только один, так как содержит базовых агентов, а создание дополнительных контейнеров не ограничено.

После запуска платформы JADE можно подключать пользовательских агентов. После написания программного кода поведения агента, необходимо скомпилировать его с подключением библиотек JADE с помощью команды:

```
javac -classpath lib\jade.jar -d bin src\mas\AgentName.java
```

Следующий шаг – подключение пользовательского агента к JADE. Добавить пользовательский агент MyAGents.MyAgent1 под именем AgentName в существующий контейнер на удаленный узел 10.10.1.12 с портом 1100 позволяет следующая команда:

```
java -cp lib\jade.jar;classes jade.Boot -host 10.10.1.12 -
port 1100 -agents AgentName:AgentName(args)
```

В некоторых случаях допустимо запускать платформу JADE с именем PlatformName и одновременно подключать агентов:

```
java -cp lib\jade.jar;classes;bin jade.Boot -name
PlatformName  -gui -agents AgentName:AgentName(args)
```

Приложение Е

Описание SWF-формата

SWF-формат состоит из двух частей: спецификации распределенной вычислительной среды и спецификации параметров задания.

Таблица Е.1 – Спецификация распределенной вычислительной среды

№	Параметр	Описание
1.	Version	Версия используемого формата SWF (2.2 – текущая версия)
2.	Computer	Имя производителя и модель кластера
3.	Installation	Территориальное место проведения эксперимента и имя кластера.
4.	Acknowledge	Персоны, которые проводят эксперимент с потоком заданий
5.	Information	Веб-сайт или e-mail, которые предоставят более подробную информацию о потоке заданий
6.	Conversion	Персона, конвертировавшая лог-файл в стандарт workload, и его e-mail
7.	MaxJobs	Общее число заданий в workload-файле
8.	MaxRecords	Общее число записей (строк) в workload-файле
9.	Preemption	Имеет четыре возможных значения: 'No' – задание является стандартным и представлено одной строкой в файле; 'Yes' – задание может быть разделено на несколько частей, каждое разделение представлено в виде отдельной строки в файле; 'Double' – задания могут быть разделены, и информация о них появляется дважды в файле (первый раз в качестве обобщенной информации в виде одной строки и второй раз, как последовательность строк, описывающих части данных

№	Параметр	Описание
		заданий); 'TS' – используется разделение времени, но нет подробных деталей
10.	UnixStartTime	Время начала ведения лога в формате Unix
11.	TimeZone	Не используется (заменен параметром TimeZoneString)
12.	TimeZoneString	Текущая временная зона
13.	StartTime	Время начала лог-файла в читаемом формате Tue Feb 21 18:44:15 IST 2006
14.	EndTime	Время окончания лог-файла в читаемом формате
15.	MaxNodes	Число вычислительных узлов кластера
16.	MaxProcs	Число процессоров кластера
17.	MaxRuntime	Максимально допустимое время выполнения задания на кластере
18.	MaxMemory	Максимально допустимое использование оперативной памяти (в килобайтах)
19.	AllowOveruse	Логический параметр, определяющий возможно ли заданиям использовать больше ресурсов, чем разрешено
20.	MaxQueues	Максимальное число очередей
21.	Queues	Словесное описание системных очередей для того, чтобы можно было определить какой номер очереди можно было бы использовать, в котором должно быть прокомментировано в какие очереди возможно поставить тот или иной тип задания (интерактивный, или стандартный)
22.	Queue	Описание очередей по формату «номер очереди, имя очереди»
23.	MaxPartitions	Число разделов
24.	Partitions	Словесное описание системных разделов для того, чтобы можно было определить какой номер раздела можно было бы использовать
25.	Partition	Описание отдельного системного раздела

№	Параметр	Описание
26.	Note	Заметки
27.	MaxCores	Число ядер кластера

Таблица Е.2 – Спецификация параметров задания

№	Параметр	Описание	Область значений
1.	Job Number	Номер задания	[1; JN]
2.	Submit Time	Время передачи задания на выполнение в секундах	[1; ST]
3.	Wait Time	Время ожидания перед выполнением задания в секундах	[1; WT] или -1
4.	Run Time	Время выполнения задания в секундах	[1; RT]
5.	Number of Allocated Processors	Число выделенных процессоров, которые использует задание	[1; NAP] или -1
6.	Average Time Used by Processor	Среднее время использования процессора в секундах	[1; APT] или -1
7.	Used Memory	Размер используемой оперативной памяти в килобайтах	[1; UM]
8.	Requested Number of Processors	Число процессоров, запрашиваемых для выполнения задания	[1; RP]
9.	Requested Time	Время, запрашиваемое для выполнения задания	[1; RT] или -1
10.	Requested Memory	Размер запрашиваемой оперативной памяти в килобайтах	[1; RM] или -1
11.	Status	Статус выполнения задания: 0 – ошибка выполнения задания, 1 – задание выполнено, 2 – частичное выполнение задания будет продолжено, 3 – последнее	

№	Параметр	Описание	Область значений
		частичное выполнение задания выполнено, 4 – ошибка при последнем частичном выполнении задания, 5 – выполнение задания отменено	
12.	User ID	Идентификатор пользователя, запустившего задание	[1; UID] или -1
13.	Group ID	Идентификатор группы, к которой принадлежит пользователь	[1; GID] или -1
14.	Executable (Application) Number	Номер выполняемого приложения	[1; EN] или -1
15.	Queue Number	Номер очереди	[1; QN] или -1
16.	Partition Number	Номер системного раздела	[1; PN] или -1
17.	Number of Preceding SubTasks	Число предшествующих подзаданий	[1; PJN] или -1
18.	Think Time from Preceding Job	Временная задержка после выполнения предыдущего задания и началом выполнения текущего задания	[1; TTPJ] или -1

Приложение Ж

Пример обработки задания под управлением МАС,
а также метапланировщиков GridWay и Condor DAGMan

На рисунке Ж.1 приведен пример схемы решения задачи, где $m_1 - m_6$ – это модули схемы, а $d_1 - d_9$ – данные, передаваемые между модулями.

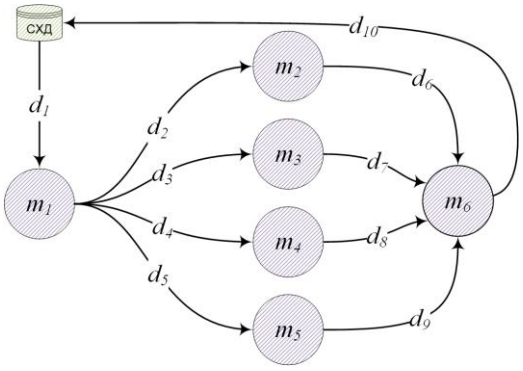


Рисунок Ж.1 – Схема решения задачи.

Для выполнения данной схемы решения задачи сгенерировано задание. Оно содержит 6 подзаданий, каждое из которых специфицирует процесс выполнения одного из модулей схемы. Подзадания классифицированы в соответствии с указанными в них требованиями по выполнению модулей. Подзадания выполняются в общем синтетическом потоке заданий, сгенерированном на основе вычислительной истории, накопленной в процессе решения ряда научных и практических задач.

Данный поток, включающие вышеупомянутые подзадания, был поочередно выполнен под управлением трех систем: метапланировщиков GridWay и Condor DAGMan, а также предложенной МАС. Перечисленные метапланировщики осуществляют управление потоками заданий на уровне РВС, распределяя их между ресурсами среды, которые работают под управлением локальных СУПЗ. Функционирование метапланировщика определяется его конфигурационными параметрами, определяющими число заданий для обработки за одну итерацию

планирования, периодичностью процессов планирования, поиска свободных ресурсов, обновления сведений о ресурсах, обновления сведений о статусах заданий и других операций.

Распределение потока заданий в МАС выполняется на основе тендера вычислительных работ. Тендер проводится в несколько раундов при поступлении задания в ВС. В каждом раунде распределяются модули очередного уровня ярусно-параллельной схемы решения задачи. Эффективность работы тендера базируется на реализации следующих решений:

- предварительной классификации заданий, обеспечивающей формирование виртуального сообщества агентов (участников тендера), каждый из которых удостоверен в наличии и заинтересован в выполнении соответствующих его ресурсам подзаданий вычислительной работы;
- разбиения схемы решения задачи, представленной ярусно-параллельной формой, на подсхемы, каждая из которых включает один ярус схемы, и проведения тендера для каждого яруса в отдельности, что позволяет агентам учитывать в процессе торгов для подзаданий некоторого яруса результаты распределения ресурсов для выполнения подзаданий предыдущих ярусов.

В таблице Ж.1 представлены характеристики процесса обработки заданий сравниваемыми метапланировщиками GridWay, Condor DAGMan и МАС. Все метапланировщики выполняют поздадания в асинхронном режиме по готовности данных. Однако МАС в отличие от GridWay и Condor DAGMan сразу ставит подзадания в локальные очереди, поддерживает непосредственную пересылку данных между агентами, распределяет подзадания по ресурсам с учетом зависимости по данным в DAG и прогнозирования времени выполнения подзаданий.

Таблица Ж.1 – Характеристики процесса обработки заданий метапланировщиками

Параметр	GridWay	Condor DAGMan	MAC
Постановка подзаданий в локальную очередь	В порядке общей очереди по готовности данных	В порядке общей очереди по готовности данных	При поступлении задания
Передача данных между подзаданиями	Через локальное хранилище; через общую систему хранения данных	Через локальное хранилище или общую систему хранения данных	Непосредственная пересылка данных между агентами
Выполнение поздаданий в асинхронном режиме по готовности данных	Да	Да	Да
Учет зависимости по данным в DAG	Нет	Нет	Да
Прогнозирование времени выполнения подзаданий	Нет	Нет	На основе вычислительной истории или результатов профилирования модулей

Преимущества распределения подзаданий одного задания по ограниченным ресурсам ВС под управлением МАС показано на примере схемы, приведенной

ранее на рисунке Ж.1.

На рисунке Ж.2 схематично представлен процесс обработки заданий в МАС, где каждый прямоугольник представляет модуль подзадания, поступающий из общей очереди ВС в локальную очередь СУПЗ. Модули передаются на выполнение на ресурсы по мере их освобождения. Фиолетовой штриховкой обозначены модули задания для выполнения вышеупомянутой схемы решения задачи. Красной штриховкой обозначены прочие задания потока. На рисунке Ж.2 ресурсы (кластеры) условно представлены прямоугольниками К1, К2, К3 и К4. Соответствующие процессы обработки заданий в GridWay и Condor DAGMan отражены на рисунках Ж.3 и Ж.4.

На разных этапах обработки задания на рисунках Ж.2- Ж.4 приводятся время t_0 начала обработки задания, время t_q нахождения подзаданий в общей очереди, время t_d передачи данных и время t_k завершения выполнения задания.

Результаты выполнения задания, приведенные в таблице Ж.2, демонстрируют очевидное преимущество МАС.

Таблица Ж.2 – Показатели выполнения задания

Параметр	t_0	t_q	t_d	t_k	Время выполнения задания
GridWay	15:00:00	00:07:29	00:01:18	15:12:49	12 мин. 49 сек.
Condor DAGMan	14:00:00	00:07:29	00:01:17	14:12:21	12 мин. 21 сек.
МАС	17:00:00	00:00:16	00:00:10	17:09:26	9 мин. 26 сек.

Обработка синтетического потока заданий под управлением МАС позволила ускорить процесс вычислений на 35.8 и 30.92% по сравнению с GridWay и Condor DAGMan соответственно. Применение МАС позволило также повысить среднюю загрузку процессора узлов ВС на 6 и 5% относительно GridWay и Condor DAGMan.

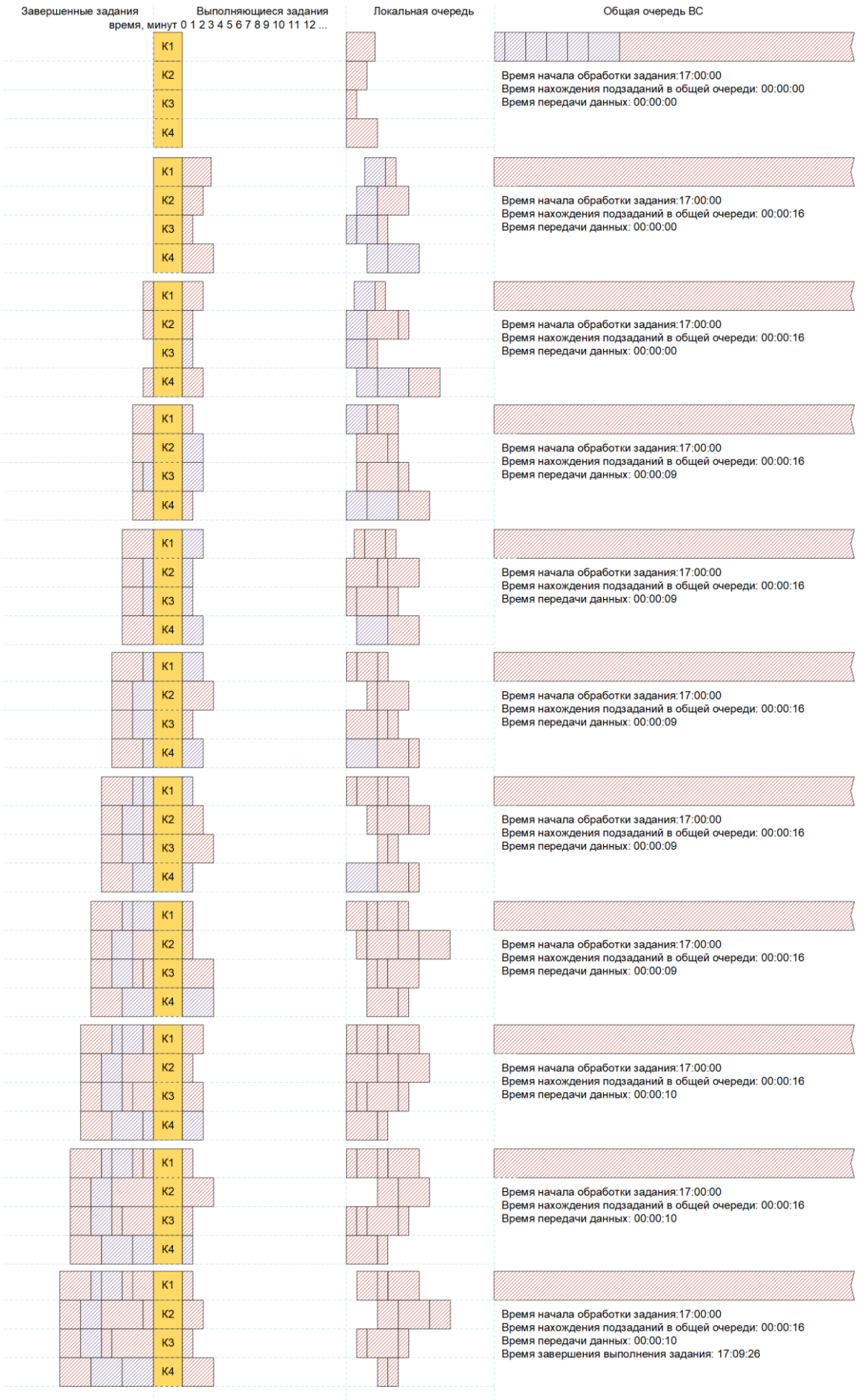


Рисунок Ж.2 – Обработка очереди заданий в МАС

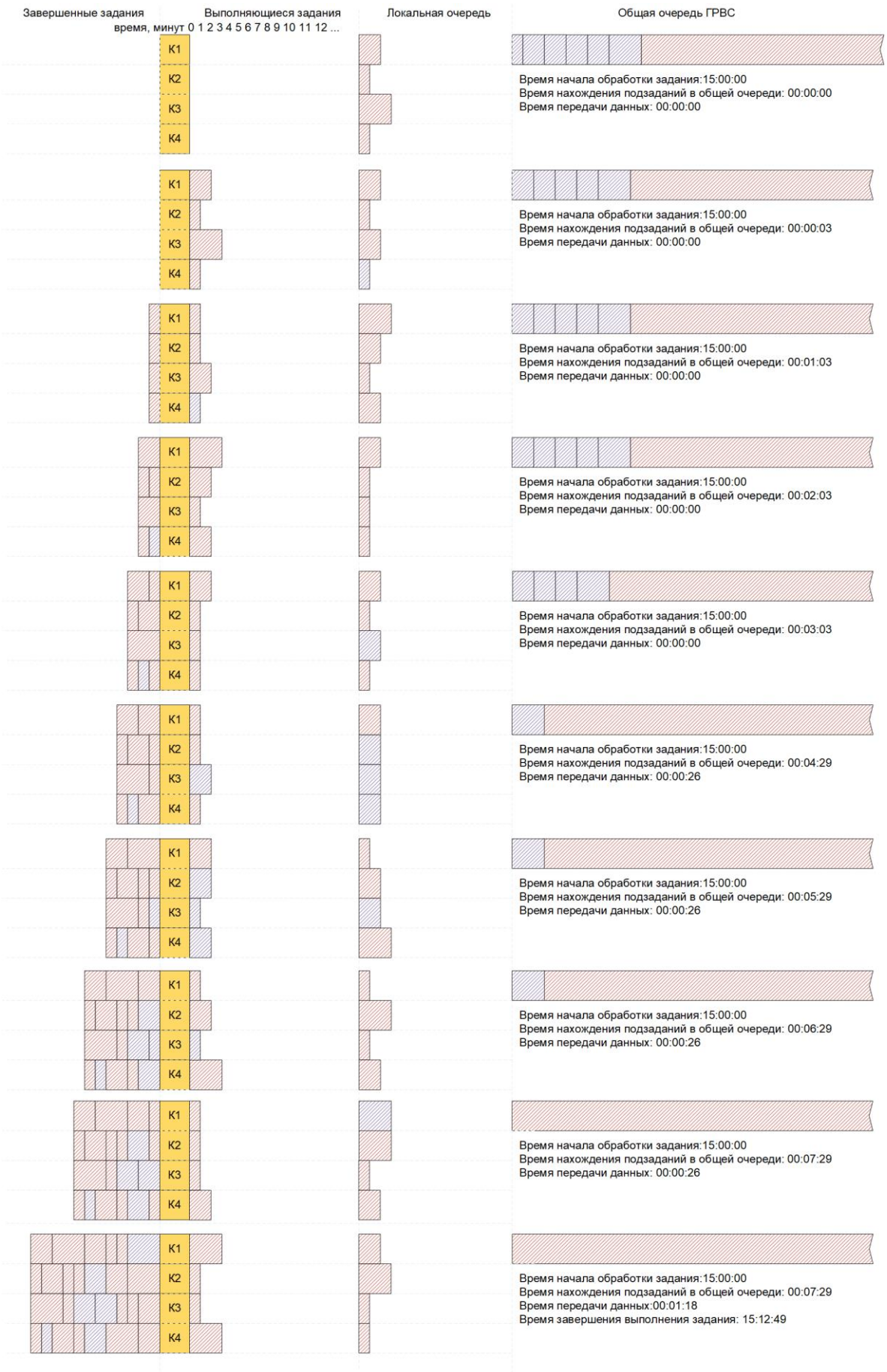


Рисунок Ж.3 – Обработка очереди заданий в GridWay

Рисунок Ж.4 – Обработка очереди заданий в Condor DAGMan

Ниже приведен фрагмент журнала операций этапа распределения модулей задания под управлением MAC.

```
Добавление нового агента в виртуальное сообщество:
id: 1
Имя: Agent_01
Ресурс: Кластер ПК - Сегмент 1
Надежность: 97
Коэффициент производительности: 90
Координатор: false

Добавление нового агента в виртуальное сообщество:
id: 2
Имя: Agent_02
Ресурс: Кластер ПК - Сегмент 2
Надежность: 96
Коэффициент производительности: 85
Координатор: false

Добавление нового агента в виртуальное сообщество:
id: 3
Имя: Agent_03
Ресурс: Кластер ВМ - Сегмент 1
Надежность: 95
Коэффициент производительности: 80
Координатор: false

Добавление нового агента в виртуальное сообщество:
id: 4
Имя: Agent_04
Ресурс: Кластер ВМ - Сегмент 2
Надежность: 99
Коэффициент производительности: 70
Координатор: true

Добавление нового агента в виртуальное сообщество:
id: 5
Имя: Agent_05
Ресурс: Кластер ПК - Сегмент 3
Надежность: 98
Коэффициент производительности: 65
Координатор: false

Добавление нового агента в виртуальное сообщество:
id: 6
Имя: Agent_06
Ресурс: Кластер ВМ - Сегмент 3
Надежность: 94
Коэффициент производительности: 110
Координатор: false
```

```
============================================================
Прогноз времени выполнения агентом Agent_01 модуля DAGJ01_m01 97.9
сек.  Коэффициент производительности: 90%
Класс slow определен для времени выполнения 97.9 сек.
Класс slow: ставка = 0.16

Прогноз времени выполнения агентом Agent_01 модуля DAGJ01_m02 79.2
сек.  Коэффициент производительности: 90%
Класс slow определен для времени выполнения 79.2 сек.
Класс slow: ставка = 0.16

Прогноз времени выполнения агентом Agent_01 модуля DAGJ01_m03 71.5
сек.  Коэффициент производительности: 90%
Класс slow определен для времени выполнения 71.5 сек.
Класс slow: ставка = 0.16

Прогноз времени выполнения агентом Agent_01 модуля DAGJ01_m04 63.8
сек.  Коэффициент производительности: 90%
Класс slow определен для времени выполнения 63.8 сек.
Класс slow: ставка = 0.16

Прогноз времени выполнения агентом Agent_01 модуля DAGJ01_m05 56.1
сек.  Коэффициент производительности: 90%
Класс normal определен для времени выполнения 56.1 сек.
Класс normal: ставка = 0.18

Прогноз времени выполнения агентом Agent_01 модуля DAGJ01_m06 46.2
сек.  Коэффициент производительности: 90%
Класс normal определен для времени выполнения 46.2 сек.
Класс normal: ставка = 0.18

============================================================
Прогноз времени выполнения агентом Agent_02 модуля DAGJ01_m01 102.35
сек.  Коэффициент производительности: 85%
Класс не найден: ставка = 1.0

Прогноз времени выполнения агентом Agent_02 модуля DAGJ01_m02 82.8
сек.  Коэффициент производительности: 85%
Класс slow определен для времени выполнения 82.8 сек.
Класс slow: ставка = 0.15

Прогноз времени выполнения агентом Agent_02 модуля DAGJ01_m03 74.75
сек.  Коэффициент производительности: 85%
Класс slow определен для времени выполнения 74.75 сек.
Класс slow: ставка = 0.15

Прогноз времени выполнения агентом Agent_02 модуля DAGJ01_m04 66.7
сек.  Коэффициент производительности: 85%
Класс slow определен для времени выполнения 66.7 сек.
Класс slow: ставка = 0.15
```

```
Прогноз времени выполнения агентом Agent_02 модуля DAGJ01_m05 58.65
сек.  Коэффициент производительности: 85%
Класс normal определен для времени выполнения 58.65 сек.
Класс normal: ставка = 0.25

Прогноз времени выполнения агентом Agent_02 модуля DAGJ01_m06 48.3
сек.  Коэффициент производительности: 85%
Класс normal определен для времени выполнения 48.3 сек.
Класс normal: ставка = 0.25

============================================================
Прогноз времени выполнения агентом Agent_03 модуля DAGJ01_m01 106.8
сек.  Коэффициент производительности: 80%
Класс не найден: ставка = 1.0

Прогноз времени выполнения агентом Agent_03 модуля DAGJ01_m02 86.4
сек.  Коэффициент производительности: 80%
Класс slow определен для времени выполнения 86.4 сек.
Класс slow: ставка = 0.85

Прогноз времени выполнения агентом Agent_03 модуля DAGJ01_m03 78.0
сек.  Коэффициент производительности: 80%
Класс slow определен для времени выполнения 78.0 сек.
Класс slow: ставка = 0.85

Прогноз времени выполнения агентом Agent_03 модуля DAGJ01_m04 69.6
сек.  Коэффициент производительности: 80%
Класс slow определен для времени выполнения 69.6 сек.
Класс slow: ставка = 0.85

Прогноз времени выполнения агентом Agent_03 модуля DAGJ01_m05 61.2
сек.  Коэффициент производительности: 80%
Класс slow определен для времени выполнения 61.2 сек.
Класс slow: ставка = 0.85

Прогноз времени выполнения агентом Agent_03 модуля DAGJ01_m06 50.4
сек.  Коэффициент производительности: 80%
Класс normal определен для времени выполнения 50.4 сек.
Класс normal: ставка = 0.75

============================================================
Прогноз времени выполнения агентом Agent_04 модуля DAGJ01_m01 115.7
сек.  Коэффициент производительности: 70%
Класс не найден: ставка = 1.0

Прогноз времени выполнения агентом Agent_04 модуля DAGJ01_m02 93.6
сек.  Коэффициент производительности: 70%
Класс slow определен для времени выполнения 93.6 сек.
Класс slow: ставка = 1.0

Прогноз времени выполнения агентом Agent_04 модуля DAGJ01_m03 84.5
сек.  Коэффициент производительности: 70%
```

```
Класс slow определен для времени выполнения 84.5 сек.
Класс slow: ставка = 1.0

Прогноз времени выполнения агентом Agent_04 модуля DAGJ01_m04 75.4
сек.  Коэффициент производительности: 70%
Класс slow определен для времени выполнения 75.4 сек.
Класс slow: ставка = 1.0

Прогноз времени выполнения агентом Agent_04 модуля DAGJ01_m05 66.3
сек.  Коэффициент производительности: 70%
Класс slow определен для времени выполнения 66.3 сек.
Класс slow: ставка = 1.0

Прогноз времени выполнения агентом Agent_04 модуля DAGJ01_m06 54.6
сек.  Коэффициент производительности: 70%
Класс normal определен для времени выполнения 54.6 сек.
Класс normal: ставка = 0.9

============================================================
Прогноз времени выполнения агентом Agent_05 модуля DAGJ01_m01 120.15
сек.  Коэффициент производительности: 65%
Класс не найден: ставка = 1.0

Прогноз времени выполнения агентом Agent_05 модуля DAGJ01_m02 97.2
сек.  Коэффициент производительности: 65%
Класс slow определен для времени выполнения 97.2 сек.
Класс slow: ставка = 1.0

Прогноз времени выполнения агентом Agent_05 модуля DAGJ01_m03 87.75
сек.  Коэффициент производительности: 65%
Класс slow определен для времени выполнения 87.75 сек.
Класс slow: ставка = 1.0

Прогноз времени выполнения агентом Agent_05 модуля DAGJ01_m04 78.3
сек.  Коэффициент производительности: 65%
Класс slow определен для времени выполнения 78.3 сек.
Класс slow: ставка = 1.0

Прогноз времени выполнения агентом Agent_05 модуля DAGJ01_m05 68.85
сек.  Коэффициент производительности: 65%
Класс slow определен для времени выполнения 68.85 сек.
Класс slow: ставка = 1.0

Прогноз времени выполнения агентом Agent_05 модуля DAGJ01_m06 56.7
сек.  Коэффициент производительности: 65%
Класс normal определен для времени выполнения 56.7 сек.
Класс normal: ставка = 0.8

============================================================
Прогноз времени выполнения агентом Agent_06 модуля DAGJ01_m01 80.1
сек.  Коэффициент производительности: 110%
Класс slow определен для времени выполнения 80.1 сек.
```

```
Класс slow: ставка = 0.1

Прогноз времени выполнения агентом Agent_06 модуля DAGJ01_m02 64.8
сек.  Коэффициент производительности: 110%
Класс slow определен для времени выполнения 64.8 сек.
Класс slow: ставка = 0.1

Прогноз времени выполнения агентом Agent_06 модуля DAGJ01_m03 58.5
сек.  Коэффициент производительности: 110%
Класс normal определен для времени выполнения 58.5 сек.
Класс normal: ставка = 0.2

Прогноз времени выполнения агентом Agent_06 модуля DAGJ01_m04 52.2
сек.  Коэффициент производительности: 110%
Класс normal определен для времени выполнения 52.2 сек.
Класс normal: ставка = 0.2

Прогноз времени выполнения агентом Agent_06 модуля DAGJ01_m05 45.9
сек.  Коэффициент производительности: 110%
Класс normal определен для времени выполнения 45.9 сек.
Класс normal: ставка = 0.2

Прогноз времени выполнения агентом Agent_06 модуля DAGJ01_m06 37.8
сек.  Коэффициент производительности: 110%
Класс normal определен для времени выполнения 37.8 сек.
Класс normal: ставка = 0.2

Отправка ставок координатору:

Agent_01: Ставка за модуль DAGJ01_m01: 0.16
Agent_02: Ставка за модуль DAGJ01_m01: 1.0
Agent_03: Ставка за модуль DAGJ01_m01: 1.0
Agent_04: Ставка за модуль DAGJ01_m01: 1.0
Agent_05: Ставка за модуль DAGJ01_m01: 1.0
Agent_06: Ставка за модуль DAGJ01_m01: 0.1

Agent_01: Ставка за модуль DAGJ01_m02: 0.16
Agent_02: Ставка за модуль DAGJ01_m02: 0.15
Agent_03: Ставка за модуль DAGJ01_m02: 0.85
Agent_04: Ставка за модуль DAGJ01_m02: 1.0
Agent_05: Ставка за модуль DAGJ01_m02: 1.0
Agent_06: Ставка за модуль DAGJ01_m02: 0.1

Agent_01: Ставка за модуль DAGJ01_m03: 0.16
Agent_02: Ставка за модуль DAGJ01_m03: 0.15
Agent_03: Ставка за модуль DAGJ01_m03: 0.85
Agent_04: Ставка за модуль DAGJ01_m03: 1.0
Agent_05: Ставка за модуль DAGJ01_m03: 1.0
Agent_06: Ставка за модуль DAGJ01_m03: 0.2

Agent_01: Ставка за модуль DAGJ01_m04: 0.16
Agent_02: Ставка за модуль DAGJ01_m04: 0.15
```

```
Agent_03: Ставка за модуль DAGJ01_m04: 0.85
Agent_04: Ставка за модуль DAGJ01_m04: 1.0
Agent_05: Ставка за модуль DAGJ01_m04: 1.0
Agent_06: Ставка за модуль DAGJ01_m04: 0.2

Agent_01: Ставка за модуль DAGJ01_m05: 0.18
Agent_02: Ставка за модуль DAGJ01_m05: 0.25
Agent_03: Ставка за модуль DAGJ01_m05: 0.85
Agent_04: Ставка за модуль DAGJ01_m05: 1.0
Agent_05: Ставка за модуль DAGJ01_m05: 1.0
Agent_06: Ставка за модуль DAGJ01_m05: 0.2

Agent_01: Ставка за модуль DAGJ01_m06: 0.18
Agent_02: Ставка за модуль DAGJ01_m06: 0.25
Agent_03: Ставка за модуль DAGJ01_m06: 0.75
Agent_04: Ставка за модуль DAGJ01_m06: 0.9
Agent_05: Ставка за модуль DAGJ01_m06: 0.8
Agent_06: Ставка за модуль DAGJ01_m06: 0.2

DAGJ01_m01 отправлен агенту Agent_06 по цене: 0.16 при его ставке:
0.1
DAGJ01_m02 отправлен агенту Agent_06 по цене: 0.15 при его ставке:
0.1
DAGJ01_m03 отправлен агенту Agent_02 по цене: 0.16 при его ставке:
0.15
DAGJ01_m04 отправлен агенту Agent_02 по цене: 0.16 при его ставке:
0.15
DAGJ01_m05 отправлен агенту Agent_01 по цене: 0.2 при его ставке:
0.18
DAGJ01_m06 отправлен агенту Agent_01 по цене: 0.2 при его ставке:
0.18
```