Московский государственный технический университет
имени Н.Э. Баумана
(национальный исследовательский университет)

На правах рукописи

Степанова Мария Владимировна

Метод и алгоритмы назначения заданий в распределенной информационной системе Интернета вещей

Диссертация на соискание ученой степени
кандидата технических наук

Специальность 05.13.11 — Математическое и программное обеспечение вычислительных машин, комплексов и компьютерных сетей
(технические науки)

Научный руководитель:
д.т.н., профессор Пролетарский А.В.

Москва - 2022

Оглавление

Стр.

Стр.

Введение

Актуальность работы. Во многих сферах человеческой деятельности существует большое количество задач, требующих для своего решения существенных вычислительных ресурсов. Для решения таких задач широко применяются параллельные и распределенные вычислительные системы (такие как системы грид-вычислений, облачные вычисления, вычислительные кластеры и т. п.). В частности, использование распределенных вычислительных систем лежит в основе проектов @Home, которые для обработки данных научных исследований используют через Интернет простаивающие компьютеры пользователей [1, 2].

В настоящее время широкое развитие и распространение получили встраиваемые системы. В частности большую популярность набирают системы, называемые Интернет вещей (ИВ, IoT - Internet of Things), основной идеей которого является встраивание вычислительных модулей в объекты окружающего мира [3]. Такие системы позволяют собирать информацию о состоянии и о функционировании с большого количества объектов, передавая эти данные через сети связи для дальнейшей обработки и анализа [4]. Интернет вещей получил широкое распространение, в частности, в сфере здравоохранения для мониторинга состояния здоровья, обеспечения процессов жизнеобеспечения [5].

Тенденция распространения ИВ позволяет сделать предположение, что в ближайшее время количество подключенных к Интернету устройств ИВ превысят количество подключенных персональных компьютеров всех видов (включая ноутбуки, планшетные компьютеры) и мобильные телефоны (Рисунок В.1) [6].

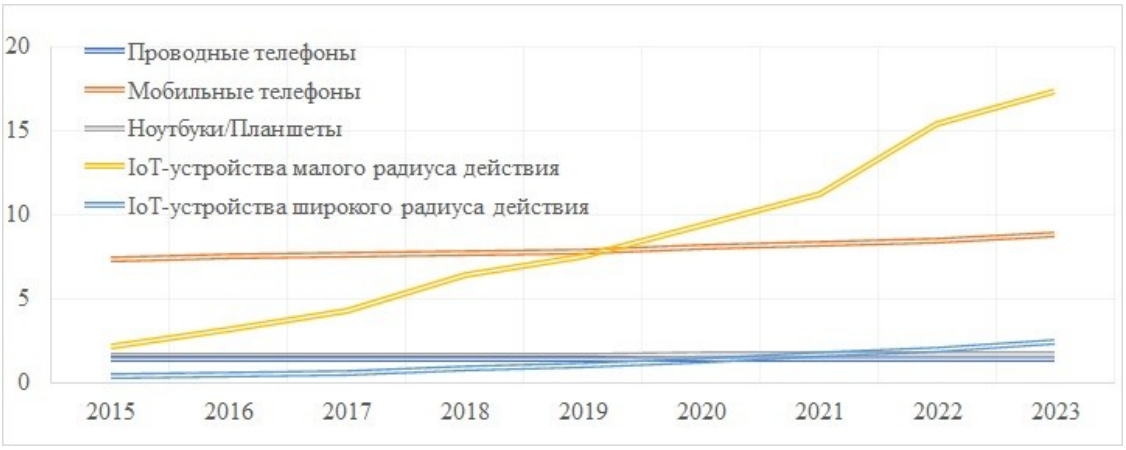

Рисунок В.1. Прогноз увеличения числа подключенных к Интернету устройств

Поскольку устройства ИВ являются вычислительными системами, то возможно построение распределенной информационной системы с вычислительными узлами на основе устройств ИВ (распределенная информационная система Интернета вещей, РИСИВ). Помимо проблем присущих классическим распределенным и параллельным вычислительным системам, в информационной системе на основе устройств ИВ также добавляются особенности встраиваемых систем: автономные источники питания, низкие скорости передачи данных, использование процессоров с низким энергопотреблением, высокие уровни помех в каналах связи, постоянное перемещение устройств, гетерогенность вычислительных узлов. Поэтому использование методов назначения заданий, которые применяются в классических распределенных и параллельных вычислительных системах, становятся неприменимым. Решение данной задачи возможно за счет использования методов, которые могут учитывать особенности устройств ИВ, а также принимать решение о назначении заданий вычислительным узлам с учетом полной неопределенности структуры информационной системы на основе устройств ИВ.

Существующие методы назначения заданий, использующие генетические алгоритмы, классическое машинное обучение и другие, обладают рядом недостатков, такими как предположение об однородности вычислительных узлов,

либо требуют полную информацию о структуре соединений между вычислительными узлами распределенной системы.

Поэтому является *актуальной* задача создания метода назначения заданий вычислительным узлам, позволяющего использовать распределенную информационную систему на основе устройств ИВ как единую вычислительную систему, что позволит повысить степень использования вычислительных мощностей устройств ИВ для решения вычислительных задач, к которым можно отнести в частности:

- задачи имитационного моделирования [7, 8];

- научные вычисления по анализу данных экспериментов [9, 10];

- обработка мультимедиа-данных: видео, аудио, графических [11, 12], в том числе в системах САПР (система автоматизации проектных работ) и АСТПП (автоматизированная система технологической подготовки производства) [13, 14]);

- задачах цифровой обработки сигналов [15, 16];

- выполнения объемных вычислений, хранения и обработки данных эко-системы промышленного ИВ [17].

В качестве основы для метода назначения заданий узлам распределенной информационной системы выбрана модель на основе машинного обучения с подкреплением, которая лишена недостатков, присущих другим методам распределения заданий в распределенных и параллельных системах.

Цель исследования. Цель настоящей работы заключается в повышении эффективности инфраструктуры ИВ путем переноса вычислительной нагрузки на устройства ИВ за счет использования метода и алгоритмов назначения заданий вычислительным узлам распределенной информационной системы на основе устройств ИВ.

Объект исследования. Объектом исследования данной работы являются распределенная информационная система на основе устройств Интернета вещей.

Предмет исследования. Предметом исследования является процесс назначения вычислительных заданий узлам распределенной информационной системы на основе устройств Интернета вещей.

Положения, выносимые на защиту. В соответствии с пунктами 3, 8 и 9 области исследований паспорта специальности 05.13.11 «Математическое и программное обеспечение вычислительных машин, комплексов и компьютерных сетей»:

«3. Модели, методы, алгоритмы, языки и программные инструменты для организации взаимодействия программ и программных систем» (разработана модель распределения заданий по вычислительным узлам на основе модели машинного обучения с подкреплением, и программные инструменты выполнения программы на распределенной информационной вычислительной системе на основе объектов ИВ);

«8. Модели и методы создания программ и программных систем для параллельной и распределенной обработки данных, языки и инструментальные средства параллельного программирования» (разработан метод назначения заданий вычислительным узлам распределенной информационной системы на основе объектов ИВ);

«9. Модели, методы, алгоритмы и программная инфраструктура для организации глобально распределенной обработки данных» (разработаны алгоритмы и программное обеспечение для распределенной обработки данных на распределенной информационной системе на основе объектов ИВ)

на защиту выносятся следующие положения:

1. Модель назначения заданий по вычислительным узлам на основе модели машинного обучения с подкреплением.

2. Метод назначение заданий вычислительным узлам распределенной информационной системы Интернета вещей на основе машинного обучения с подкреплением, позволяющий динамически перенастраивать процедуру распределения заданий по вычислительным узлам распределенной

информационной системы, характеристики которых могут изменяться в широких пределах.

3. Алгоритмы распределенной информационной системы на основе объектов Интернета вещей, позволяющие назначать задания вычислительным узлам распределенной информационной системы в зависимости от их характеристик и степени доступности

4. Рекомендации по применению системы назначения заданий при работе с вычислительными узлами распределенной информационной системы на основе устройств Интернета вещей.

Метод назначения. Для достижения поставленной в работе цели использовались следующие методы исследования: теория принятия решений; теория принятия решений в условиях неопределенности; системный анализ; анализ процесса решения вычислительных задач на распределенных системах; анализ методов машинного обучения; анализ методов машинного обучения с подкреплением; моделирование взаимодействия агент-среда применительно к данной задаче; экспериментальный анализ результатов работы модели распределения заданий на основе машинного обучения с подкреплением.

Персоналии. Существенный вклад в область машинного обучения с подкреплением внесли М.Л. Цетлин, Р. Саттон и другие.

В области высокопроизводительных вычислительных систем (параллельного и распределенного типа) общего и специального назначения внесли В. Воеводин, Вл.Воеводин, О.М. Брехов, Э. Таненбаум, М. ван Стеен, В.П. Гергель, Р.Г. Стронгин и другие.

В области многопараметрической оптимизации и обработки сигналов А.П. Карпенко, В.В. Сюзев, А. Оппенгейм, Р. Шафер и другие.

В области компьютерных сетей и Интернета вещей И.П. Иванов, Б.В. Костров и другие.

Научная новизна. Научная новизна полученных в диссертации результатов теоретических и экспериментальных исследований заключается в следующем:

- разработана модель назначения заданий вычислительным узлам на основе модели машинного обучения с подкреплением, которая, в отличие от уже существующих, позволяет работать на распределенной информационной системе, узлы которой являются устройствами Интернета вещей;

- разработан новый метод назначения заданий вычислительным узлам распределенной информационной системы Интернета вещей на основе машинного обучения с подкреплением, позволяющий, в отличие от известных, динамически перенастраивать процедуру распределения заданий по вычислительным узлам распределенной информационной системы, характеристики которых могут изменяться в широких пределах;

- разработаны алгоритмы и программное обеспечения для распределенной информационной системы на основе устройств Интернета вещей, позволяющие назначать задания вычислительным узлам распределенной информационной системы в зависимости от их характеристик и степени доступности;

- разработаны научно-обоснованные рекомендации по применению системы назначения заданий при работе с вычислительными узлами распределенной информационной системы на основе устройств Интернета вещей, позволяющей повысить эффективность инфраструктуры Интернета вещей для решения вычислительных задач за счет применения динамической модели машинного обучения с подкреплением.

Практическая значимость исследования. На основе метода, разработанного в диссертационной работе, создана система назначения заданий вычислительным узлам, предназначенная для решения вычислительных задач с помощью распределенной информационной системы на основе устройств Интернета вещей.

Анализ модели машинного обучения с подкреплением позволил получить алгоритм, который впервые был применен для назначения заданий вычислительным узлам, реализованных на основе устройств Интернета вещей. В результате задания, являющиеся составными частями комплексной

вычислительной задачи, стало возможным распределять и выполнять на узлах, характеристики которых существенно меняются с течением времени, при этом, изменяя параметры алгоритма, можно изменять поведение системы в соответствие с текущими состояниями вычислительных узлов. Адаптивность алгоритма позволила уменьшить влияние изменения параметров вычислительных узлов на процесс решения вычислительной задачи, и, следовательно, использовать в качестве вычислительных узлов устройства Интернета вещей.

Внедрение результатов исследований. Полученные в диссертационной работе результаты внедрены в компании ООО «ЦПР РТСофт», и в учебном процессе.

Для использования в учебном процессе кафедры «Компьютерные системы и сети» (ИУ6) и кафедры «Инновационное предпринимательство» (ИБМ7) МГТУ им. Н.Э. Баумана разработана распределенная информационная система на основе устройств ИВ, позволяющая проводить назначение заданий вычислительным узлам и демонстрировать все этапы работы машинного обучения с подкреплением, и приложение, выполняющее построение двухмерных изображений по трехмерной модели на основе алгоритма трассировки лучей. Результаты работы использованы при изучении дисциплин «Распределенные высоконагруженные вычислительные системы», «Технические средства хранения и обработки больших данных», «Облачные технологии» и «Информационные системы в управлении».

Апробация работы. Результаты диссертационной работы представлены и прошли апробацию на следующих конференциях:

1. Международная конференция «eLearning and Software for Education Conference», Румыния, 2019 г.

2. Международная научная конференция Математические Методы в Технике и Технологиях ММТТ-33, Россия, 2020 г.

3. Международная научная конференция «Кибер-физические системы: проектирование и моделирование» CYBERPHY:2020 – «Cyber-Physical Systems Design And Modelling», Россия, 2020 г.

4. The International Conference on Deep Learning, Big Data and Blockchain (Deep-BDB 2021), Италия, 2021 г.

5. The 4th International Conference on Recent Innovations in Computing (ICRIC-2021), Индия, 2021.

Публикации по теме. По теме диссертационной работы опубликовано 11 научных работ: из них 3 в научных изданиях, входящих в Перечень ВАК Минобрнауки России, 1 статья в научных изданиях, индексируемых Scopus и Web of Science, 2 статьи в научных изданиях, индексируемых Scopus.

Личный вклад автора. Все представленные в данной работе исследования и результаты проведены и получены лично соискателем. Из совместных публикаций в работу включен лишь материал, который непосредственно принадлежит соискателю. Материал, заимствованный у других авторов обозначен в работе ссылками на соответствующие публикации.

Структура диссертационной работы. Диссертация состоит из введения, четырех глав с выводами, общих выводов по диссертации и заключения, списка использованных источников. Основной текст содержит 142 страницы, 39 рисунков, 1 таблицу. Список использованных источников содержит 12 страниц и включает 127 наименований.

В первой главе проведён анализ характеристик, ограничений и возможностей Интернета вещей как распределённой информационной системы. Рассмотрены существующие подходы для реализации параллельных и распределенных вычислительных систем. Проведён анализ методов машинного обучения в качестве применимости для реализации процесса назначения вычислительных заданий в РИСИВ. В результате анализа, выделен метод машинного обучения с подкреплением как основа метода назначения заданий в РИСИВ. Описаны элементы метода обучения с подкреплением.

Во второй главе сформулирована постановка задачи, формальное описание которой основано на задаче оптимального отображения алгоритма на архитектуру параллельной вычислительной системы с использованием графовой модели, приведена модель РИСИВ и метод для назначения заданий её вычислительным узлам.

В третьей главе приводится модель вычислительного узла и алгоритм назначения заданий вычислительным узлам от распределяющего узла в РИСИВ, которые основываются на предложенных формальном описании, методе и стратегии в главе 2. Приводится также модифицированный алгоритм назначения заданий с использованием дополнительного уровня распределения заданий по группам вычислительных узлов, тем самым сократив вычислительную нагрузку по взаимодействию с распределяющим узлом и дающий возможность масштабирования разработанного метода и алгоритма назначения заданий в РИСИВ.

В четвёртой главе представлена техническая реализация РИСИВ. Представлены результаты исследования разработанного метода и алгоритмов. Рассматривается работа алгоритма распределения при различных значениях характеристик и типах поведения вычислительных узлов. Даются рекомендации по применению метода и алгоритмов для реализации и проведения распределенных вычислений на РИСИВ. Описываются проведённые эксперименты и полученные результаты: поведение алгоритма при стационарном состоянии, поведение алгоритма при нестационарном состоянии, тестирование работы алгоритма при отправке тестового задания трассировки луча.

В библиографии приведены литературные источники (монографии, статьи, интернет-ресурсы и др.), на которые есть ссылки в тексте работы.

Глава 1. Анализ концепции распределенных вычислений на основе Интернета вещей

Интернет вещей (ИВ, IoT - Internet of Things) превратился из маркетинговой концепции в развитую технологию, которая представляет собой множество объектов, содержащих микроконтроллер, которые объединены друг с другом и с устройствами инфраструктуры посредством беспроводных каналов связи [4, 18, 19]. Для обслуживания таких устройств было предложено множество инфраструктурных элементов: центры обработки данных (ЦОД), облачные технологии, кластерные технологии, системы агрегации и обработки данных и другие. Все указанные элементы в своей массе представляют платформу Интернета вещей.

По своей сути множество взаимодействующих устройств ИВ представляет параллельную и распределенную информационную вычислительную систему, которая, имея существенные отличия от других распределенных систем, может быть использована для построения распределенной информационной системы Интернета вещей (РИСИВ), позволяющую решать вычислительные задачи.

Процедура решения вычислительных задач состоит из ряда взаимозависимых этапов. В общем виде комплексная вычислительная задача преобразуется в последовательность заданий (подзадач), которые в дальнейшем обрабатываются вычислительными узлам. И одним из этапов является назначение заданий вычислительным узлам, которое может быть выполнено различными способами, для устройств Интернета вещей в качестве вычислительных узлов эта процедура усложняется за счет постоянного изменения их параметров и пространственной конфигурации.

1.1 Интернет вещей

Внедрение вычислительных и радио модулей в объекты реального мира, когда объекты могут взаимодействовать друг с другом, обмениваться параметрами, собирать информацию о своем состоянии, привел к появлению технологии ИВ. Несмотря на формирование альянсов производителей [20] и появление стандартов на ИВ [21, 22], встраиваемые модули не являются унифицированными, но реализуются на базе нескольких аппаратных платформ. Выбор аппаратной платформы модулей определяется характеристиками и особенностями объектов, к которым он применяется. С датчиков устройств ИВ поступает информация, которая может быть обработана и/или в неизменном виде концентрируется в базе данных для дальнейшего использования [3, 23]:

- мониторинг состояния и перемещения объектов ИВ;
- агрегация информации;
- сбор исторических данных объектов ИВ;
- анализ и выявление тенденций в поведении и в состоянии объектов ИВ;
- обнаружение неявных закономерностей в данных об объектах ИВ с использованием технологий интеллектуального анализа данных (Data Mining).

Учитывая все множество устройств Интернета вещей, сервисы и службы по обслуживанию поступающих данных (сбор, обработка, аналитика) можно говорить о формировании инфраструктуры Интернета вещей (или платформы Интернета вещей).

Инфраструктура Интернета вещей [18] является одной из технологий межмашинного взаимодействия (Machine to Machine, M2M), которая может быть сведена к трехуровневой модели (Рисунок 1.1):

- облачные ЦОД, предоставляющих ресурсы для выполнения аналитических приложений;
- управляющие ЦОД;
- отдельные вычислительные устройства (непосредственно устройства и объекты Интернета вещей).

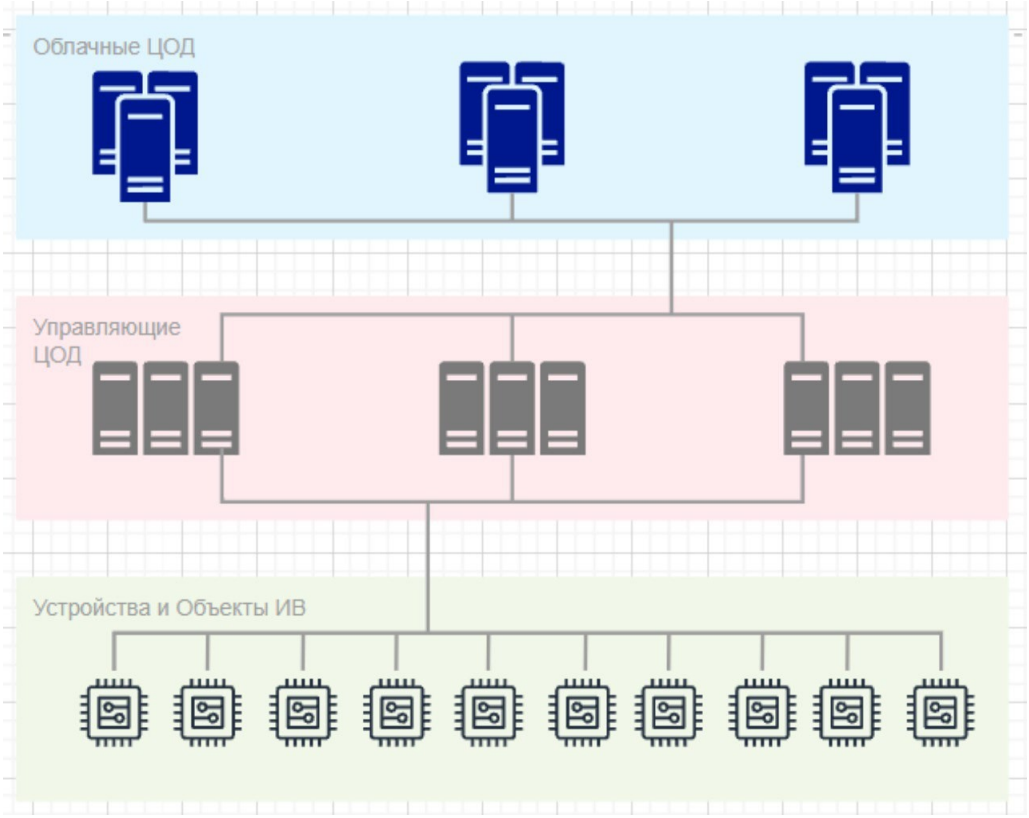

Рисунок 1.1. Трехуровневая модель инфраструктуры Интернета вещей

Классические подходы функционирования инфраструктуры ИВ предполагают использование облачных вычислений (Cloud computing), которые позволяют строить «гибкие» информационные системы с учетом информационных потоков между хранилищами данных, центрами обработки и поиска данных. Иными словами, инфраструктура ИВ имеет дело с большими данными (Big Data), которые определяются характеристиками «три V» [24]:

- volume (объем) — работа с большим количеством данных различной степени структурированности;

- velocity (скорость) — высокая скорость прироста хранимых и обрабатываемых данных;

- variety (многообразие) — большое число используемых технологий для обеспечения обработки данных различной степени структурированности.

Переход к уровню межмашинного взаимодействия позволяет уменьшить потоки данных между элементами инфраструктуры ИВ как на уровне облачных ЦОД, так и на уровне управляющих ЦОД. Благодаря тому, что в данной

вычислительной парадигме вычисления «опускаются» с уровня облачных ЦОД на более низкий уровень управляющих ЦОД, то такой подход называется туманными вычислениями (Fog computing), отличающийся от уже упомянутого подхода, называемого облачными вычислениями [25].

Туманные вычисления имеют следующие характеристики [26, 27]:

- распределение вычислительной мощности;

- географическое распределение компонентов;

- работа с большими массивами данных (в том числе в режиме реального времени);

- сложная топологическая структура;

- мобильность вычислительных узлов и беспроводной доступ;

- гетерогенность вычислительных узлов и используемых технологий (протоколов, каналов связи, типов оборудования);

- взаимодействие с аналитическими платформами (в том числе облачного типа);

- согласованность между всеми элементами вычислительной инфраструктуры.

В настоящее время наиболее развиты два верхних уровня модели инфраструктуры Интернета вещей: сервис доступа к своим платформам с облачными технологиями предоставляют компании, например, Amazon [28, 29], Google [29, 30], Microsoft [31]; для работы с устройствами Интернета вещей платформа IBM Bluemix [3], которая была интегрирована с облачной платформой более высокого уровня и в настоящее время являющаяся подсистемой платформы IBM Cloud [32].

Рассматривая инфраструктуру ИВ как единую информационную систему, можно отметить, что дальнейшее перемещение вычислений с уровня управляющих ЦОД на уровень вычислительных устройств, представляемых устройствами ИВ, позволяет повысить эффективность использования пропускной способности сети [33, 34, 35]. Данная парадигма называется граничными

вычислениями (Edge computing) [36]. Граничные вычисления позволяют также задействовать существенные вычислительные мощности растущего количества подключенных к Интернету устройств ИВ [37].

Рассматриваемая в данной работе распределенная информационная система Интернета вещей (РИСИВ) относится к системам граничных вычислений. Построение такой информационной системы заключается в использовании устройств ИВ в качестве вычислительных узлов, на которых возможна обработка и хранение не только локальной информации, но и совместное выполнение общих вычислительных задач. Вычислительные узлы такой системы обладают всеми характеристиками устройств ИВ:

- гетерогенность устройств;
- различные аппаратные характеристики;
- различные энергетические режимы;
- изменение местоположения;
- низкая стабильность и низкая пропускная способность каналов связи;
- и другое.

Учитывая вышесказанное, можно констатировать, что распределенная информационная система на основе устройств Интернета вещей — РИСИВ — относится к системам, модель которых практически невозможно построить ввиду того, что ее параметры постоянно меняются (структура, каналы связи, режимы работы), что приводит к возникновению ряда специфических проблем.

1.2 Обработка информации в системах Интернета вещей

Области использования и внедрения технологий Интернета вещей достаточно широки и разнообразны. В отчете компании IOT ANALYTICS [38] и в [39] выделены следующие:

- производство/промышленность;
- транспортировка/перемещения;
- энергетика;

- торговля;
- города;
- здравоохранение;
- цепочки поставок;
- сельское хозяйство;
- строительство;
- другие (проекты из финансовой и корпоративной сферы).

Использование технологий ИВ в промышленности позволяет даже говорить об отдельной отрасли - промышленный Интернет вещей (IIoT - Industrial Internet of Things).

Существующая на сегодняшний день схема обработки информации в информационной системе ИВ показана на Рисунке 1.2

Устройства или объекты ИВ осуществляют сбор первичной информации с датчиков, осуществляют агрегацию данных, а также, в некоторых случаях, хранение первичной информации. В схеме на рисунке устройства ИВ не являются интеллектуальными, их способность к взаимодействию друг с другом минимальная. Процессы обработки и анализа данных, а также принятия решения не являются частью непосредственно устройств ИВ, а реализованы в других элементах инфраструктуры ИВ (например, с использованием облачных технологий).

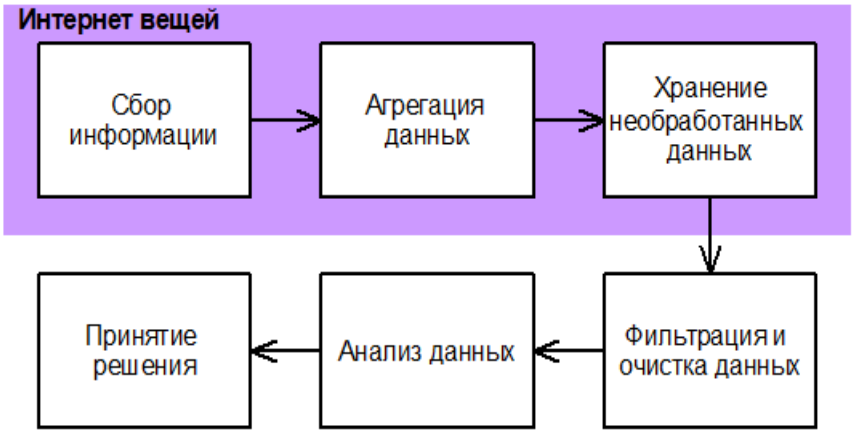

Рисунок 1.2. Схема обработки информации ИВ

Дальнейшее развитие технологий ИВ позволяет говорить не только о возможности постоянного мониторинга состояний объектов и об их постоянном подключении к сетям передачи данных, но также о возможности работать автономно [40, 41], то есть осуществлять не только сбор и хранение данных, но также их обработку (Рисунок 1.3).

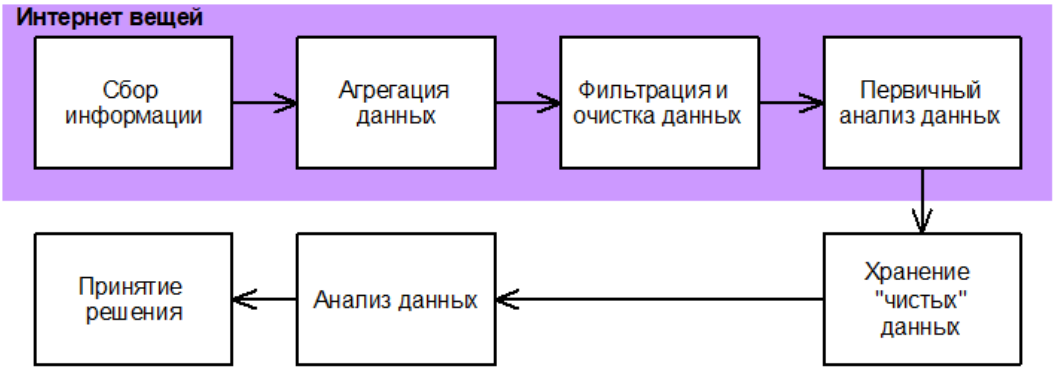

Рисунок 1.3. Схема обработки информации ИВ с элементами автономности

В работах [40, 41] указывается, что снижение стоимости обработки информации за счет усовершенствования микроконтроллеров позволяет использовать инфраструктуру ИВ для более сложной работы с поступающими данными, чем простой сбор информации (с отложенной обработкой и анализом), на основе чего становится возможным принятие при управлении технической системой. Таким образом, можно говорить, что устройства ИВ начинают выступать не только в роли источников информации, но и также сами «должны рассматриваться в качестве потребителей информации» [41], что позволит снизить различные риски, возникающие за счет «задержек передачи информации и распределенными вычислениями».

Устройства ИВ обладают определенными возможностями для обработки и хранения данных, что позволяет обеспечить агрегацию и анализ данных. Некоторые устройства ИВ имеют достаточные ресурсы для обработки данных напрямую, но некоторые могут передавать данные на другие устройства (например, шлюзы). Как отмечается в [42] что граничные вычисления для анализа данных («граничный анализ») может быть выполнен в режиме реального времени,

и позволяет избежать передачи больших объемов информации на облачное хранилище или в ЦОД для дальнейшей обработки. Это позволяет снизить требования к каналу передачи данных, а также снижает нагрузку на сеть и уменьшает требования к хранилищу. Конечно, устройства, выполняющие подобную обработку, требуют повышенных технических характеристик, чем простые устройства, выполняющие базовую обработку, такую как проверка, нормализация, фильтрация, преобразование данных.

Будучи потребителями обработанной информации, концепция ИВ расширяется до концепции Интернета автономных вещей (IoAT - Internet of Autonomous Things). Устройства ИВ начинают выступать в роли исполнительных устройств, способных функционировать как отдельные устройство, но и за счет кооперации и самоорганизации выполнять задания совместно (Рисунок 1.4).

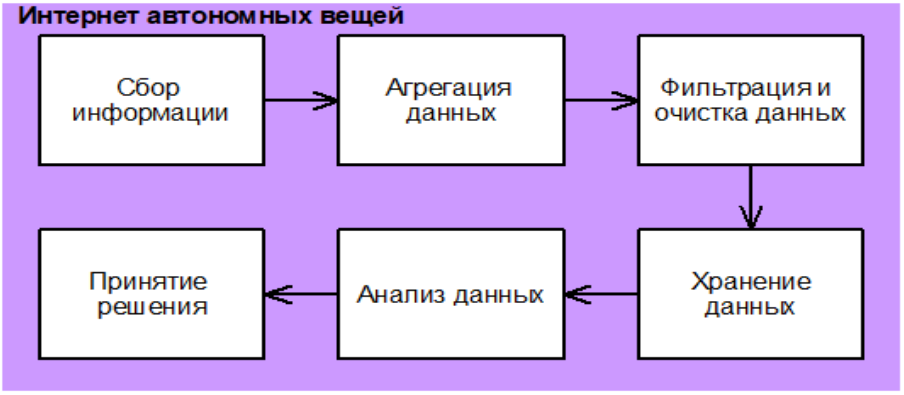

Рисунок 1.4. Схема обработки информации Интернета автономных вещей

В своих разработках контроллеров для ИВ компания Analog Devices, являющаяся одним из лидеров рынка ИВ [43], предлагает «привносить интеллект в конечные узлы сети Интернета вещей» с целью обеспечения полноты, актуальности и безопасности данных [44].

Необходимость в автономных системах, основанных на ИВ, появляется, в первую очередь, в местах, где подключение к сетям передачи данных отсутствует (либо нерегулярное), но при этом имеется возможность использовать устройства ИВ.

В качестве примера последнего можно привести автономную систему в области сельского хозяйства - сельскохозяйственный ИВ (AIoT, IoTAg - Agriculture Internet of Things), считающийся подмножеством IIoT [45]. Например, компания-производитель [46] предлагает ряд решений: устройства ИВ подключаются к крупному или мелкому рогатому скоту в виде ошейников, которые отслеживают параметры животных, их перемещения. Информация с ошейников передается на базовые станции, расположенные в различных местах пастбища так, чтобы всегда была возможность передать информацию (до 15 км) от животных на хранилище данных на базовых станциях для дальнейшей передачи в сеть фермера для анализа и обработки. Базовые станции получают питание от солнечных батарей. Поскольку пастбища могут располагаться достаточно далеко от линий связи, то в этих условиях единственным решением является накопление информации с регулярным обслуживанием базовых станций с целью переноса с них данных.

Иначе говоря, в данной автономной системе требуется иметь носители информации большого объема, а также необходимо регулярное обслуживание персоналом базовых станций. При этом, как было сказано ранее, современные устройства ИВ имеют такие характеристики, которые позволяют их использовать в роли вычислительных узлов для обработки поступающий информации. Накопление обработанных данных в рассматриваемом примере позволяет снизить требования к носителю информации (жесткий диск, флеш-накопитель) для хранения данных, а также позволит увеличить интервалы между обслуживанием автономной системы ИВ с целью «съема» данных.

Во вторую очередь, Интернет автономных вещей может быть использован с целью повышения безопасности данных, для чего нет необходимости осуществлять подключение к внешним сетям передачи данных. Например, компания EnOcean [47] предлагает необслуживаемые и полностью автономные инфраструктуры ИВ, которые могут найти свое применение в автоматизации инженерных систем зданий («умный дом», «умное здание»).

Также инфраструктура Интернета автономных вещей является полностью самодостаточной в обработке информации, но в тоже время позволит, возможно, получить синергетический эффект от использования различных аспектов информационных технологий.

Следовательно, можно сделать вывод, что использование незадействованных вычислительных возможностей устройств ИВ позволит повысить эффективность инфраструктуры ИВ (иначе говоря, перейти от схемы на Рисунке 1.2 к схеме на Рисунках 1.3 и 1.4), а разработка распределенной информационной системы на основе устройств Интернета вещей - РИСИВ - является задачей актуальной задачей [48].

1.3 Параллельные и распределенные вычислительные системы

Решение сложных технических и научных задач предполагает построение моделей, параметры которых могут быть рассчитаны аналитически, либо с использованием численных методов. Для реализации последних использование электронно-вычислительных машин (ЭВМ) позволяет получать более адекватные результаты с меньшими затратами человеческих и временных ресурсов. Часть задач может быть реализована с использованием ЭВМ с одним вычислителем, но для большинства современных задач моделирования необходимо использовать различного типа и назначения параллельные или распределенные системы [49, 50, 51]. Использование таких систем определяется тем, что достаточно часто решение задачи — это набор однотипных действий, которые имею простую структуру (например, матричные преобразования, подбор параметров, перебор значений и т. п.). В частности, в области компьютерной графики, где необходимо постоянно обсчитывать параметры графических объектов (повороты, освещение, движение), это привело к появлению специализированных графических процессоров (GPU) [52]. А последние эксперименты с такими структурами как нейронные сети позволило проводить вычисления параметров нейронных сетей

на графических процессорах [53], что существенно подтолкнуло развитие этой области знаний [54].

Другим примером является решение систем дифференциальных уравнений с использованием численных методов, которые на системе с несколькими вычислителями позволяют получать результат значительно быстрее, чем на системе с одним вычислителем.

Используя классификацию Флинна [55, 56] к параллельным и распределенным системам можно отнести:

- ОКМД (Одиночный поток команд - Множество потоков данных, SIMD - Single Instruction Multiple Data);

- МКМД (Множественный поток команд — Множество потоков данных, MIMD – Multiple Instruction, Multiple Data);

- МКОД (Множественный поток команд — Один поток данных, MISD - Multiple Instruction, Single Data).

Но в [57, 58] отмечается, что несмотря на существенное отличие параллельных систем между собой, все они относятся в большей степени к классу МКМД [59, 60, 61, 62]. На Рисунке 1.5 приведена структура класса многопроцессорных вычислительных систем согласно [58].

Параллельные системы можно подразделить на два класса, отличающиеся по принципам и подходам работы с системами:

- централизованные системы;

- распределенные системы.

В централизованных системах все вычислители находятся на относительно небольшом расстоянии друг от друга: от непосредственного расположения на одном кристалле (в случае с многоядерными процессорами или ядрами графических процессоров) до нескольких метров (в случае, если множество параллельно работающих систем расположены в одном центре хранения и обработки данных). Последние также обладают характеристиками

распределенных систем и в них могут использоваться методы, которые присущи только распределенным системам.

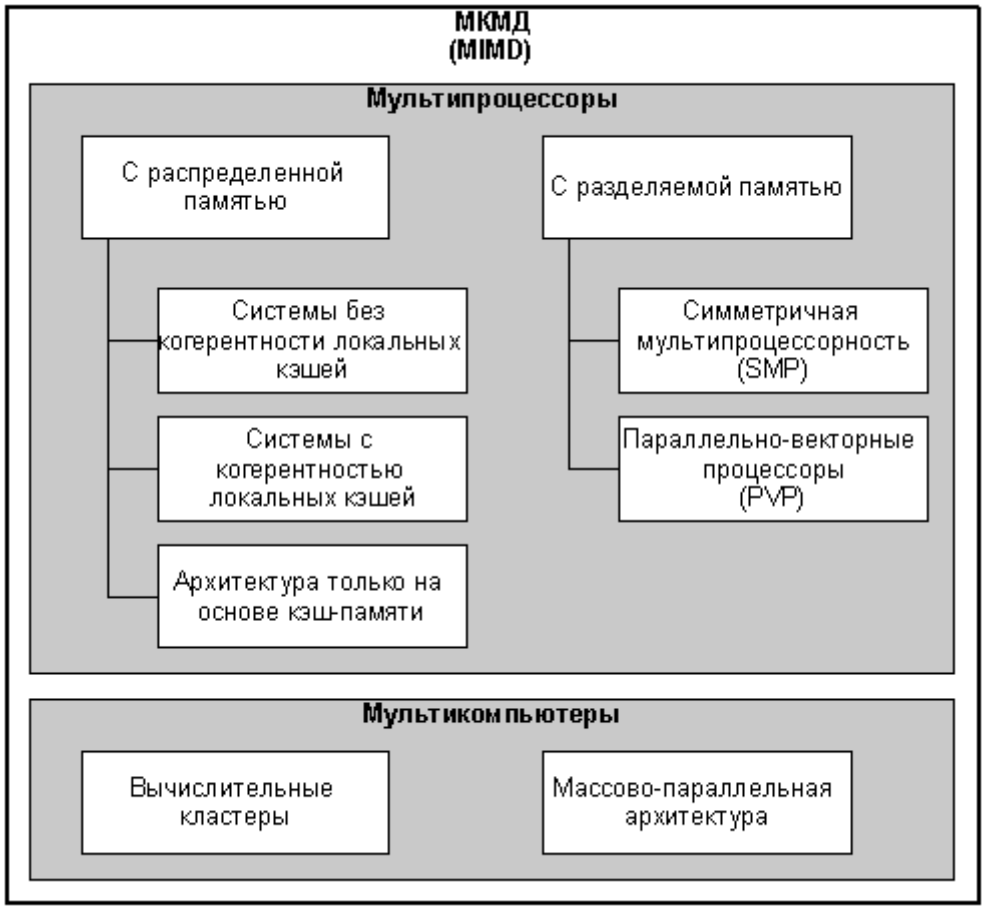

Рисунок 1.5. Структура класса многопроцессорных вычислительных систем (по классификации в [58])

Общим элементом централизованных систем является наличие постоянных (относительно постоянных) во времени характеристик системы, то есть отсутствуют существенные сложности синхронизации вычислителей, доступа к ним, потерь данных между узлами.

Распределенные вычислительные системы обладают характеристиками, которыми не обладают централизованные системы, но позволяющие выделить именно распределенные системы [63]:

- прозрачность;
- открытость;
- надежность;
- производительность;

- масштабируемость.

Распределенные вычислительные системы могут представлять собой системы, элементы которых расположены на существенном расстоянии друг от друга [64, 65, 66]. В частности, самым известным являются системы грид-вычислений [67, 68, 69] проектов @HOME, в частности, по поиску радиосигналов из космоса SETI@HOME [1] или по поиску лекарства от рака [70], в которых компьютеры участников, выступающие вычислительными узлами, распределены по всем континентом земного шара.

1.4 Кластерные вычислительное системы

Кластерные системы, как определено в [71] - это совокупность совместно функционирующих вычислительных устройств, соединенных линиями связи. При этом кластер - это слабосвязанная вычислительная система, которая отдельных вычислительных узлов и которая работает для пользователя как единая система. Также автор [71] отмечает, кто кластерные системы являются «разновидностью параллельной или распределенной системы».

Учитывая, что каждый объект ИВ содержит в себе все необходимые элементы компьютера (процессор, ОЗУ, ПЗУ и т. п.), то реализованная единая информационная система на основе устройств ИВ будет являться кластером.

Кластеры могут подразделяться на следующие виды [72]:

- кластеры высокой доступности (High-availability clusters);

- кластеры для высокопроизводительных вычислений (High-performance computing clusters);

- смешанные системы.

Также можно выделить еще два дополнительных вида кластерных систем:

- системы распределенных вычислений;

- кластеры с балансировкой нагрузки.

Стоит отметить, что два последних вида кластерных систем могут входить в первые три вида кластерных систем, либо могут их в себе содержать.

В [73] отмечается, что к кластерным вычислительным системам предъявляется следующее:

- требование высокой готовности;
- требование высокого быстродействия;
- возможность масштабирования;
- обеспечение общего доступа к ресурсам;
- удобство и простота обслуживания.

Вопрос масштабирования вычислительных систем является краеугольным камнем в вопросах повышения производительности вычислений. Масштабирование вычислительных систем может рассматриваться в двух аспектах:

- вертикальное масштабирование — предполагает увеличение производительности вычислительной системы за счет повышения производительности отдельных элементов системы: например, повышение частоты центрального процессора, повышение его разрядности, расширение шин передачи данных, увеличение скорости работы оперативной памяти и т. п.;

- горизонтальное масштабирование — предполагает, что вычислительная система может строится из отдельных вычислительных узлов, а для повышения общей производительности системы необходимо увеличивать количество таких вычислительных узлов.

Добавление в вычислительную систему дополнительных вычислителей (например, сопроцессора или графических процессоров GPU - Graphics Processing Unit), либо увеличение количества ядер процессора можно было бы также отнести к горизонтальному масштабированию. Но так как это обычно реализуется на одной печатной плате устройства, либо на общем кристалле (система на кристалле, SoC – System on Crystal), и эти дополнительные вычислители не могут функционировать как отдельные устройства, то можно считать их архитектурными модификациями единого вычислительного устройства, то есть представляет собой вертикальное масштабирование.

Также стоит отметить, что вертикальное масштабирование, как правило, имеет ряд ограничений. Во-первых, это физические ограничения: например, в настоящее время невозможно повышать частоту и производительность процессора в несколько раз только за счет уменьшения размера транзистора. Во-вторых, это архитектурные ограничения: невозможно установить 64-разрядный процессор в систему, разработанную для работы с 32-разрядными процессорами. Несмотря на указанные ограничения, вертикальное масштабирование может быть использовано при горизонтальном масштабировании, когда отдельные вычислительные узлы могут заменяться на более производительные с улучшенными аппаратными характеристиками.

Кластеры высокой доступности

Кластерные системы высокой доступности используются в областях, где время простоя является критически важной характеристикой, такие как банковские системы, системы электронной коммерции, системы финансовых расчетов, системы управления предприятием и т. п.

К основным проблемам кластерных систем высокой доступности можно отнести следующее:

- обеспечение отказоустойчивости;
- обеспечение минимального времени простоя;
- возможность замены оборудования без остановки системы;
- минимизация времени восстановления в случае сбоев.

Решение данных проблем осуществляется путем дублирования всех элементов системы, а также за счет использования специального оборудования, которое имеет высокую степень надежности. В такого рода системах необходимо предусматривать простоту обслуживания, что также должно снижать время простоя при проведении технических работ.

Кластеры высокопроизводительных вычислений

Кластеры высокопроизводительных вычислений используются в областях, где необходимо наличие существенных вычислительных мощностей (причем ограничение на высокую доступность не накладывается), такие как обработка изображений (распознавание образов, формирование двухмерных изображений), моделирование, обработка и анализ научных данных, обучение нейронных сетей.

Смешанные кластерные системы

Смешанные кластерные системы в зависимости от предъявляемых бизнес-требований могут обладать в разной мере характеристиками как кластеров высокопроизводительных вычислений, так и кластеров высокой доступности.

По степени связанности элементов в кластере можно выделить [73, 74]:

- тесносвязанные системы;
- умеренносвязанные системы;
- слабосвязанные системы.

Тесно связанная система показана на Рисунке 1.6.

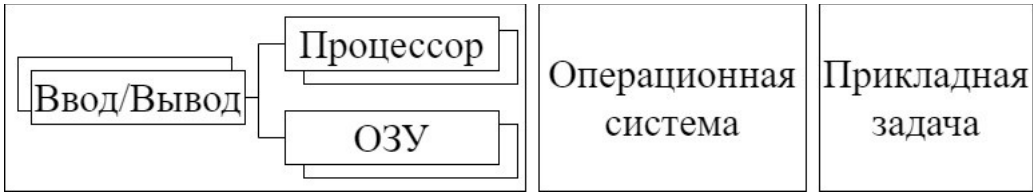

Рисунок 1.6. Тесносвязанная система

В тесносвязанных системах все процессоры расположены в одном блоке (возможно, на одной шине), при этом они разделяют общие устройства ввода/вывода, а также могут разделять оперативную память. По большей части, такие системы можно назвать классическими системами с параллельной обработкой информации. Размещение большого количества процессоров на одной печатной плате приводит к увеличению стоимости вычислительной системы (в связи с увеличением количества узлов печатного монтажа), поэтому системы имеют низкую степень масштабирования: возможно только вертикальное масштабирование. Также такие системы имею высокую стоимость, как за счет

использования специальных архитектурных решений, так и за счет более сложного контроля качества компонентов.

Структурная схема умеренно связанной (распределенной) системы показана на Рисунке 1.7. При таком подходе подразумевается, что вычислительная задача будет решаться на вычислительной системе как на тесно связанной, но при этом вычислительные узлы будут представлять собой отдельные аппаратные устройства, содержащие компоненты, такие как процессоры, память, подсистему ввода/вывода. Операционная система на каждом вычислительном узле будет своя, но она также может быть частью более крупной распределенной системы, такой как Amoeba [75].

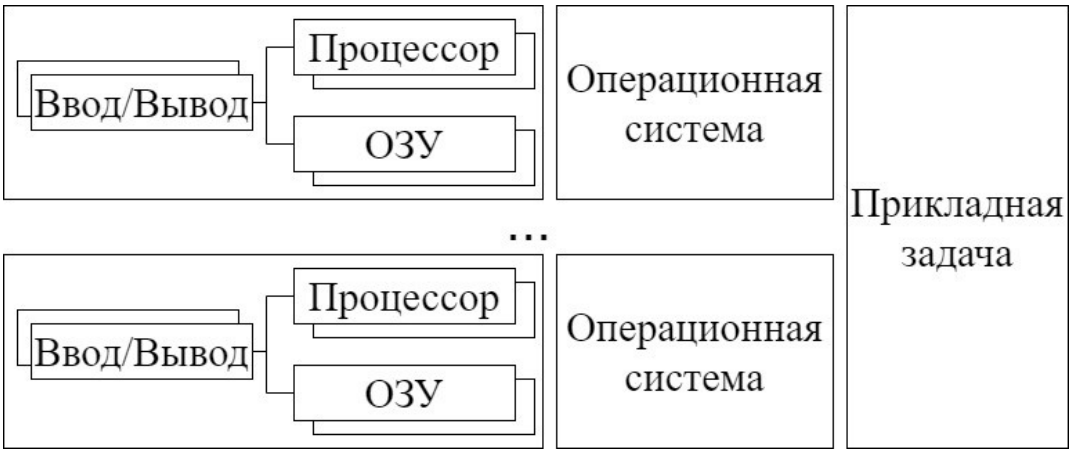

Рисунок 1.7. Умеренносвязанная система

Умеренносвязанная система имеет высокую степень масштабирования: как горизонтальное в сторону увеличения вычислительных узлов, так и вертикальное. Возможным ограничением такой системы может стать требование к вычислительным узлам, что они должны быть построены на единой платформе. Использование гетерогенных узлов для мультипроцессорной системы такого типа приводит к ее усложнению (в плане согласования, синхронизации и арбитража) и переходу системы к системе со слабой связностью.

Слабосвязанная (распределенная) система показана на Рисунке 1.8. Слабосвязанная (распределенная) система имеет в качестве вычислительных узлов независимые компьютеры (вычислительные устройства), которые могут быть реализованы на различных аппаратных платформах.

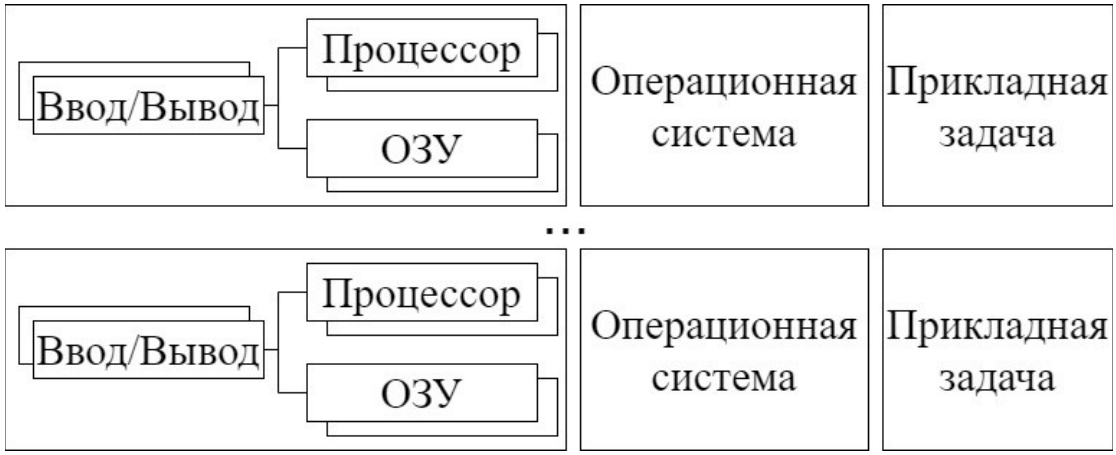

Рисунок 1.8. Слабосвязанная система

Операционная система на узлах слабосвязанных систем, как правило, является полноценной, а распределенная обработка может заключаться в том, что исходная вычислительная задача разделяется на подмножество отдельных, относительно независимых, подзадач, каждая из которых выполняется на вычислительном узле. Реализация вычислительного процесса в данном случае осуществляется на прикладном уровне модели OSI, а вопросы именования, адресации и синхронизации решаются стандартными инструментами стека сетевых протоколов.

Поскольку узлы независимы друг от друга, то в роли таких узлов в слабо связанной мультипроцессорной системе могут выступать устройства ИВ, которые могут быть архитектурно реализованы на основе различных аппаратных устройств: ARM, i386, i386_x64 и т. п. Но в данном случае необходимо, чтобы на этих платформах было реализовано специальное программное обеспечение, которое позволяет реализовать распределенную обработку данных.

1.5 Разработка программ для распределенных вычислительных систем

Независимо от типа используемой параллельной системы перед разработчиками алгоритмов возникает ряд задач по разделению исходной проблемы на множество частей, которые могут быть обработаны отдельными элементами, или узлами, вычислительной системы. В [57] приведены способы

организации вычислений, применимых к параллельным системам, и которые можно с небольшой модификацией перенести на распределенные системы:

- провидение декомпозиции комплексной вычислительной задачи, чтобы получить последовательность составляющих ее подзадач (заданий), которые могут быть распределены на выполнение по вычислительным узлах;
- для полученных подзадач (заданий) необходимо выделить информационные взаимодействия, которые возникают при решении комплексной вычислительной задачи;
- сопоставить подзадачи вычислительным узлам, а затем выполнить отправку подзадач на вычислительные узлы для их исполнения.

В тесносвязанных вычислительных системах предполагается, что каждый вычислительный узел (процессор) имеет одинаковые вычислительные способности, поэтому вопрос балансировки в данном случае можно считать несущественным, чего не скажешь про распределенные системы, где каждый вычислительный узел может иметь существенно отличные характеристики от других узлов.

Общая схема разработки параллельных алгоритмов показана на Рисунке 1.9 [57].

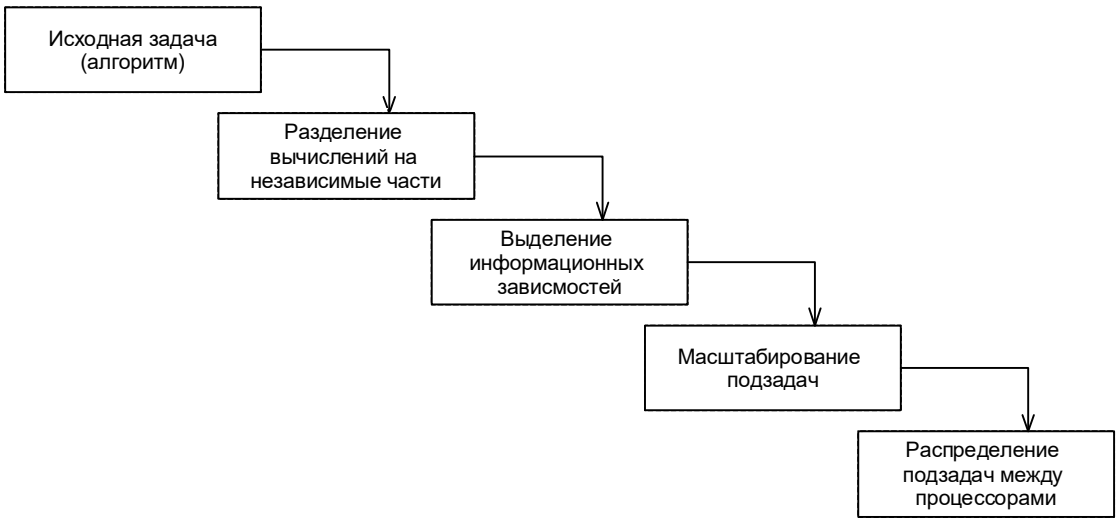

Рисунок 1.9. Общая схема разработки параллельных алгоритмов

В зависимости от характеристик вычислительного узла возможно выполнение масштабирования вычислительной задачи за счет агрегации или

детализации: соответственно, объединение подзадач в более крупные элементы, либо разделены на еще более мелкие подзадачи [76].

В результате выполнения указанных этапов необходимо сформировать программу, которая будет выполнять необходимое разбиение подзадачи и выполнять все необходимые действия для получения решения задачи. Такая программа носит название *метапрограммы*.

Таким образом, метапрограмма получает на вход набор подзадач и связей между ними, а также параметры вычислительной системы (количество и характеристики вычислительных узлов), в результате чего на распределенной системе можно получить решение исходной задачи.

Для оценки эффективности разработанного алгоритма решения задачи можно ввести показатели качества, на основе которых можно сравнивать результаты работы распределенной системы с другими типами систем. К таким показателям можно отнести [57]:

- ускорение;
- эффективность;
- масштабируемость.

Анализ этих показателей позволит определить необходимость корректировки любого из этапов работы с задачей для ее решения на распределенной системе.

Для выполнения задачи на РИСИВ необходимо запустить метапрограмму на выполнение, что может быть реализовано в виде процессов или потоков. Информационное взаимодействие между выполняемыми частями программы осуществляется посредством каналов передачи сообщений, которые основаны на стеке протоколов, реализованных для канала связи между устройствами ИВ.

Каждый вычислительный узел РИСИВ выделяется для решения одной задания (подзадачи), но в том случае, когда количество заданий будет велико, то возможно их выполнять на одних и тех же узлах, что определяется

характеристиками узла и алгоритмом сопоставления задания конкретному вычислительному узлу (назначение заданий вычислительным узлам).

Согласно [77] в программе можно выделить следующие типы действий:

- преобразователи — осуществляют преобразование и изменение информации;

- распознаватели — определяют последовательность выполнения преобразователей.

Действия связаны между собой связями (отношениями) двух типов:

- операционная (по управлению);

- информационная.

Таким образом, любая программа представляется в виде множества действий и множества связей между ними, то есть в виде графа, где вершинами обозначаются действия, а связи между ними — дугами. Такой граф именуется управляющим графом.

Представляя ситуацию, когда программа выполняется на вычислителе последовательного типа, то можно каждое срабатывание оператора представлять в виде вершины нового графа, который отличается от графа управления, но позволяет точно определить порядок выполнения каждого оператора. Такой граф называется операционно-логической историей [77, 78], в котором количество вершин зависит от входных данных. Действительно, до начала выполнения, например, программы, содержащей циклические структуры, невозможно определить количество реальных действий, которые будут выполнены на вычислителе.

Проведя преобразования над графом операционно-логической истории (объединяя вершины, соответствующие одному оператору, удаляя при этом кратные дуги) можно получить новый граф, который является подграфом управляющего графа [78].

Также возможно построение еще одного типа графа, который также зависит от входных данных, — граф истории реализации программы, когда каждое

срабатывание оператора является вершиной, которые связаны дугами передачи информации.

Либо возможно использование графа алгоритма [78], который является еще одной структурой, позволяющей описать программу. Стоит отметить [79], что все виды указанных графовых моделей могут быть преобразованы друг в друга.

Таким образом, для выполнения задачи на распределенной системе её необходимо представить в виде графа какого-либо из указанных видов, который покажет элементы задачи — подзадачи, или задания, которые будут связаны дугами, которые могут определять не только последовательности, но и также зависимости между подзадачами.

В результате последовательность заданий (подзадач) поступает на распределительный узел, в котором на основании выбранного метода планирования заданий осуществляется назначение (сопоставление) заданий вычислительным узлам, а затем выполняется отправка назначенных заданий на устройства ИВ на исполнение (Рисунок 1.10).

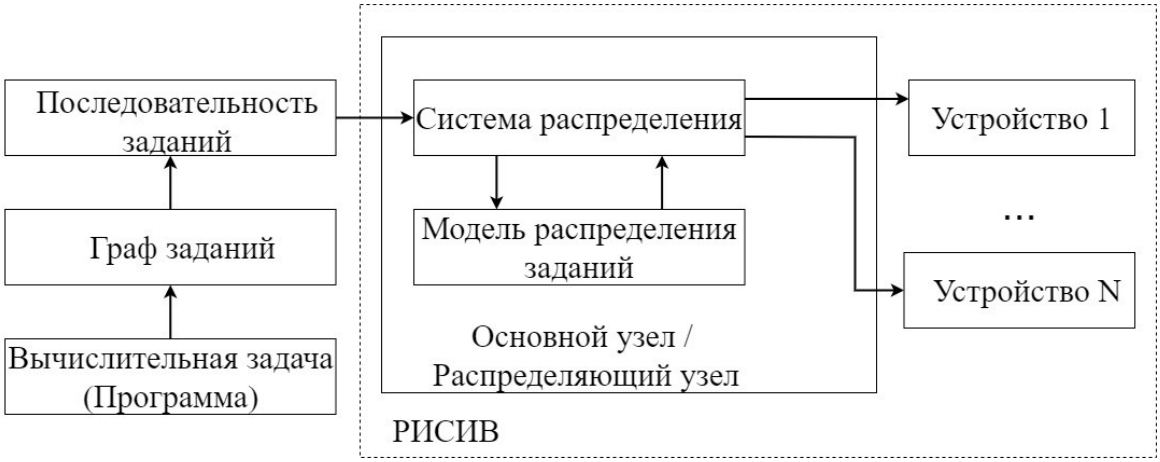

Рисунок 1.10. Схема обработки последовательности подзадач

Распределяющие узлы могут в своей основе работать с моделями распределенной системы разного вида:

- модели, в которых определена вся структура сети распределенной системы;

- модели, в которых определена часть структуры сети распределенной системы;

- модели, в которых структура распределенной системы не определена (или ее невозможно определить).

В зависимости от модели распределенной системы можно выделить различные подходы, которые могут быть использованы для отправки заданий узлам распределенной системы.

1.6 Планирование заданий в распределенной информационной системе Интернета вещей

Устройства ИВ содержат в своей основе один или несколько вычислительных модулей. Технологически такие модули выполнены в виде одного изделия типа «система на кристалле» (SoC). Как уже указывалось ранее, модули могут иметь различную архитектуру, иметь разный набор периферии, различные характеристики, тем не менее общим для них всех является следующие особенности:

- устройства ИВ имеют встроенный модуль беспроводной связи (например, WiFi, 4G, LoraWAN);

- наличие аккумулятора;

- наличие режима низкого энергопотребления, в том числе может быть несколько режимов работы для обеспечения экономичности;

- относительно невысокие скорости передачи данных;

- возможная работа в обстановке с высоким уровнем радиопомех;

- устройства ИВ могут быть мобильными и постоянно изменять свое местоположение.

Указанные характеристики отсутствуют у классических распределенных систем, где подразумевается достаточно высокие скорости передачи данных по каналам связи, неизменность структурной организации системы, достаточно высокая стабильность и надежность [80]. Например, распределенная файловая система GFS (Google File System) [81, 82] в основе своей концепции

подразумевает наличие высокой пропускной способности канала, хотя также в ней предполагается, что узлы будут иметь низкую надежность.

Ограничения устройств ИВ также ограничивает круг задач, которые могут быть решены на распределенной системе с вычислительными узлами, имеющими непостоянные характеристики. Таким образом, в данном случае распределенная система должна иметь планировщик заданий на распределяющем узле, который имеет соответствующие методы и алгоритмы назначения заданий узлам с учетом ограничений ИВ. Иными словами, распределяющий узел будет работать с моделью, в которой структура распределенной системы не определена, либо определена лишь часть структуры.

В общем виде взаимодействие планировщика заданий с вычислительными узлами реализуется по архитектуре «клиент-сервер» [83, 84], где вычислительный узел выполняет роль сервера, а распределяющий узел — клиент (показано на Рисунке 1.11).

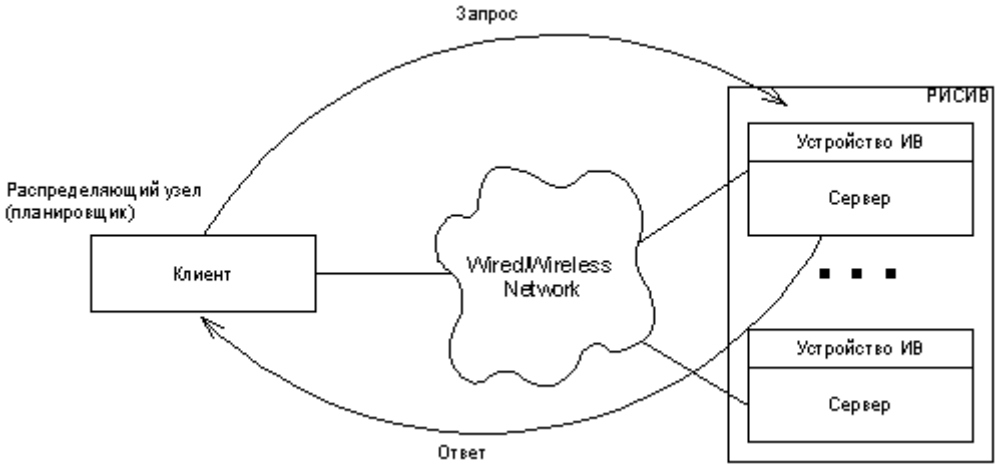

Рисунок 1.11. Клиент-серверная архитектура РИСИВ

В литературе часто предполагается (например, в [85]) в указанной архитектурной модели наличие достаточно большого количества клиентов и небольшого количества (или одного) серверов. В таком случае возникает проблема большого количества запросов к серверу, что решается использованием технологии балансировки нагрузки [83, 84].

В клиент-серверной архитектуре РИСИВ данной проблемы не существует, так как клиент (распределяющий узел) не будет направлять запрос (задание)

серверу (вычислительному) узлу до тех пор, пока сервер занят обработкой текущего запроса.

Необходимо отметить терминологический казус, что в распределенных системах термины «балансировка нагрузки» (load balancing) и «распределение нагрузки» (network load balancing) являются синонимами, в то время как «распределение заданий» (task assignment) имеет существенно отличное значение. Поэтому в данной работе для устранения амбивалентности используется термин «назначение заданий», синонимичный термину «распределение заданий».

На сегодняшний день наиболее частым случаем является подход, когда передача всего информационного потока, генерируемого устройствами ИВ, осуществляется в облачные хранилища без предварительной обработки. Авторы [23] замечают, что обработка информации сенсорных сетей на вычислительных узлах, основанных на встроенных технологиях, дает потенциально большую энергоэффективность, поскольку отпадает необходимость передавать по каналам связи необработанную информацию: «затраты энергии на сетевую передачу больших объемов данных превосходит затраты на их локальную обработку» [23]. В [86] указывается, что использование вычислительных мощностей встраиваемых устройств дает преимущество в два раза в плане энергоэффективности при передаче обработанных данных, чем при передаче необработанных данных.

Устройства ИВ могут определяться как «автономные мобильные интеллектуальные датчики», хотя имеют все характеристики вычислительных систем, но обладающими своими особенностями. Ограничения устройств ИВ не дают возможность в настоящее время использовать свою распределенную инфраструктуру для выполнения вычислительных задач, поскольку имеющиеся методы назначения заданий, используемые в современных параллельных и распределенных вычислительных системах, не могут использоваться в системах с изменяющимися параметрами вычислителей.

1.7 Машинное обучение

Любая система, которая обладает интеллектуальностью, должна также уметь изменяться за счет способности к обучению. Сложность создания обучающихся систем связана с тем, что необходимо рассматривать множество различных областей знаний.

Машинное обучение относится к одному из направлений искусственного интеллекта, позволяющему решать широкий спектр задач. Отличительной особенностью машинного обучения является способность работы с исходной информацией, представлением данных и стратегиями обучения. В [87] машинное обучение делится на три большие категории:

1) машинное обучение, в основе которого лежит символьное представление информации;

2) машинное обучение на основе коннективистского подхода;

3) машинное обучение на основе социальных и эмерджентных принципов.

Методы, относящиеся к первой категории, предполагает использование наборов символов, которые представляют сущности и взаимодействие между ними. Такие алгоритмы обучения позволяют строить обобщения, которые также выражаются в символьном виде.

Обучение, основанное на символьном подходе, осуществляет формирование обобщений с учетом имеющегося опыта, когда повторное решение одной и той же задачи, а также похожих задач, приводит к повышению эффективности системы. Поскольку все возможные варианты задачи невозможно представить обучающей системе, то в указанной группе методов используется индуктивный подход. Пространство поиска в задачах машинного обучения достаточно велико, более того, необходимо также выбрать наилучшее решение при неполноте входных примеров.

Машинное обучение на основе символьного представления подразделяется на следующие группы [87]:

- обучение на основе подобия;

- обучение на основе объяснения;
- обучение с учителем;
- обучение без учителя;
- обучение с подкреплением.

Классификация методов машинного обучения приведена в Таблице 1.

Таблица 1.

Методы машинного обучения

Критерии сравнения	Методы машинного обучения							
	Основанные на символьном представлении					На основе связей	На основе социальных принципов	
	Обучение на основе подобия	Обучение на основе объяснения	Обучение с учителем	Обучение без учителя	Обучение с подкреплением	Нейронные сети	Генетические алгоритмы	Эволюционные алгоритмы
Не требует классифицированных данных	-	+	-	-	+	-	X	X
Возможность реакции на окружающую среду	-	-	-	-	+	+	-	-
Возможность работы при существенном изменении окружающей. среды	-	-	-	+	+	-	-	-
Работа при отсутствии инвариантных свойств	+	+	+	+	+	-	+	+
Не требует наличия целевой функции	+	+	+	+	+	+	-	-

Обучение на основе подобия, обучение на основе объяснения и обучение с учителем предполагают, что данные для обучения классифицированы. В общем случае все три первые группы машинного обучения первой категории можно назвать *обучение с учителем (supervised learning)*. Обучаемой системе

представляются данные и информация о том, к какому классу они относятся: которые решают задачу или которые не решают эту задачу. В любом случае требуется наличие априорной информации, которая может быть изначально заложена в систему, либо представлена «*извне»* в форме учителя.

В задаче *обучения без учителя* (*unsupervised learning* или *learning without a teacher*) предполагается, что ценная информация из данных извлекается без наличия исходных классифицированных данных, что приводит к проблеме того, как можно определить, какие категории «*хорошие»*, а какие «*плохие»*. Обучение с подкреплением (ОП, RL - reinforcement learning) интеллектуальный агент помещается в изначально неизвестную среду, и только по ответной реакции на свои действия от среды он определяет, насколько его результативно.

Вторая категория машинного обучения представляется группой алгоритмов, в которых осуществляется изменение и модификация связей (коннективистский подход), соединяющими небольшие вычислительные элементы, а процесс обучения представляет собой «записью» примеров действий в виде указанной связующей сети. Методы, основанные на связях, позволяют обнаруживать неизменные качества и фрагменты в данных и записывать их в структуру сети.

Основными методами в машинном обучении на основе кннективистского подхода являются искусственные нейронные сети (artificial neural networks), представляющие собой множество взаимосвязанных пороговых элементов (называемых искусственными нейронами), а знания в неявном виде записываются в межэлементных связях. Нейронные сети «*не строят явную модель мира, а сами принимают его форму»* [87]. Иными словами, нейронные сети не запоминают новую информацию об окружающем мире, а обучаются за счет модификации своей структуры в ответ на воздействующие на них сигналы из окружающей среды.

Нейронные сети в наибольшей степени подходят для решения следующих групп задач [87, 88, 89]:

- классификация/кластеризация образцов;

- распознавание образов;
- ассоциация образов / реализация памяти;
- прогнозирование;
- оптимизация;
- фильтрация.

Все эти задачи объединяет то, что во входных данных выявляются инвариантные свойства (например, полезный сигнал в случае фильтрации и т. п.).

Третья категория методов машинного обучения строится на принципе эмерджентности, то, когда система рассматривается как множество отдельных компонентов, но при этом общая система не является суммой качеств этих компонентов. К таким системам относятся, в частности, генетические и другие эволюционные алгоритмы, когда решение задачи представляет в виде целой популяции решений-кандидатов, оцениваемых по определенному принципу (фитнес-функция), позволяющих отобрать лучшие решения-кандидаты для следующей итерации алгоритма. После комбинации и изменений решений-кандидатов снова осуществляется их оценка. В итоге такого подхода получается последовательность решений, которые дают все более точные решения.

Машинное обучение второй и третьей категорий требуют наличия целевой функции, на основании которой будет приниматься решение о том, насколько полученный результат соответствует ожиданиям, но в случае неопределенной окружающей среды такой подход будет неопределенным. Если рассматривать машинное обучение первой категории, то у первых трех групп методов требуется наличия примеров («правильного» и/или «неправильного» класса), на основе которых принимается решение, насколько полученный результат соотносится с исходными классами. Обучение без учителя не требует наличия предварительно классифицированных примеров для агента, но тем не менее осуществляется разделение поступающих примеров на группы. Но в случае, когда осуществляется взаимодействие со средой, и в принципе любое действие дает какой-то результат, больший или меньший, то осуществить оценку такого результата невозможно, так

как агенту неизвестны другие возможные ситуации. Поэтому требуется метод, который бы позволял учитывать указанную ситуацию.

Подобным методом является машинное обучение с подкреплением [90].

Обучение с подкреплением не требует наличия примеров требуемого поведения, которые позволяли бы определять, какое действие правильное, а какое - нет. Агент в данном случае должен обучаться только за счет своего взаимодействия с окружающей средой, то есть только на основе собственного опыта.

С одной стороны, существующие подходы машинного обучения на основе нейронных сетей, генетических алгоритмов позволяют учесть изменчивость окружающей среды. Но, с другой стороны, используемые для обучения примеры не покрывают все множество взаимодействий агента со средой. Нет гарантии, что при существенном изменении среды не приведет к тому, что заранее обученный агент не столкнется со снижением качества своей работы на достаточно длительном временном интервале. Решением проблемы поиска компромисса между использованием имеющихся данных об окружающей среде и поиском новых видов взаимодействия стало машинное обучение с подкреплением, или просто обучение с подкреплением (ОП, RL - Reinforcement Learning). Таким образом, ОП позволяет решать дилемму исследования и применения (exploration / exploitation).

Обучение с подкреплением рассматривает целостную картину проблемы целенаправленного взаимодействия агента с неопределенной окружающей средой. Агент должен выполнять действия вне зависимости от имеющегося у него количества информации о среде. Поскольку агент не имеет возможности предугадывать влияние своих действий на окружающую среду, то он обязан контролировать состояние среды и в соответствие с этим принимать решение о своих последующих действиях.

Само ОП возникло как разновидность кибернетического эксперимента и многое было заимствовано из области стохастической аппроксимации, оптимального управления и марковского процесса принятия решений.

1.8 Обучение с подкреплением

Основными элементами модели обучения с подкреплением являются агент, который получая сигналы и формируя сигналы воздействия, взаимодействует с окружающей средой (Рисунок 1.12).

Рисунок 1.12. Схема взаимодействия в машинном обучении с подкреплением

Также модель обучения с подкреплением содержит следующие элементы:

1) стратегия поведения агента;

2) функция поощрения;

3) функция ценности;

4) модель окружающей среды.

Все элементы являются обязательными, кроме модели окружающей среды. Последняя может как присутствовать, так и отсутствовать, в зависимости от решаемой задачи.

Стратегия поведения агента

Агент воспринимает состояние окружающей среды, а затем выполняет некоторое действие над окружающей средой. Стратегия поведения агента - это отображения множества состояний окружающей среды в множество действий. Стратегия определяет характеристику поведения агента в каждый конкретный момент времени, причем агент может иметь множество стратегий, в том числе и совершенно случайную.

Функция поощрения

Для оценки степени «успешности» каждого действия агента на окружающую среду используется функция поощрения, которая каждому состоянию среды ставится в соответствие числовая характеристика - вознаграждение (reward). Иными словами, функция поощрения определяет цель агента в задаче ОП, которая заключается в увеличении (максимизации) общего вознаграждения на длительном промежутке времени.

Функция поощрения не зависит от действий агента, но результат функции может послужить для агента поводом для изменения используемой стратегии, либо ее выбора из множества возможных стратегий. Величина сигнала вознаграждения определяет текущую эффективность выполняемого действия в зависимости от состояния и реакции окружающей среды. В общем случае функция поощрения может быть случайной.

Функция ценности

Функция ценности определяет эффективность поведения взаимодействующего с окружающей средой агента на протяжении всего времени взаимодействия, то есть степень желательность каждого состояния при учете наиболее вероятной последовательности следующих состояний и их вознаграждений. Ценность действия для текущего состояния является интегральной характеристикой всех возможных значений вознаграждений,

получаемых в будущем, если было выбрана некоторая последовательность действий в том случае, когда для агента текущее состояние будет начальным.

Необходимо отметить, что в долгосрочной перспективе текущее полученное агентом вознаграждение не обязательно должно быть максимальным, но соответствующее этому вознаграждению действие может приводить к действиям с б*о*льшими суммами вознаграждений, чем при максимальном текущем вознаграждении. Определить такую ситуация достаточно непросто, так как значение ценности должно вычисляться на протяжении всего времени взаимодействий агента со средой (также и учитывая возможное изменение стратегий поведения агента). Именно поэтому точное вычисление функции ценности не является строго необходимым в задачах ОП.

В случае невозможности расчета функции ценности можно провести хотя бы приблизительную ее оценку с учетом стратегии за счет использования методов аппарата поисковой оптимизации. Такими методами могут быть основаны на эволюционных, или вдохновленных природой, алгоритмов [91, 92], такие как генетические алгоритмы, алгоритмы роя частиц и другие. Подобные подходы могут быть эффективными с точки зрения поиска в пространстве стратегий, но только при небольшом их количестве. В [90] отмечается, что использование эволюционных алгоритмов оптимизации может давать преимущество при невозможности восприятия агентом сигналов от окружающей среды с достаточной точностью. В других же ситуациях эволюционные подходы пренебрегают полезными качествами, лежащими в основе ОП: обучение через взаимодействие со средой.

Модель окружающей среды

Агент может иметь некое представление об окружающей среде (заложенное в него заранее, либо полученное на основании опыта) - модель окружающей среды. Модель предназначена для имитации возможной реакции среды на возможную последовательность действий агента, то есть, если известны

состояние среды и действие агента, то можно рассчитать следующее состояние среды, а следовательно, и возможное значение вознаграждения. Модель не является обязательным элементом ОП, но позволяет проводить планирование поведения агента. В общем случае реализации ОП агент реализует метод перебора вариантов, или метод проб и ошибок.

1.9 Выводы по первой главе

Рассмотрение концепции ИВ позволяет сделать вывод, что устройства ИВ могут выступать в роли вычислительных узлов для распределенной информационной системы. В настоящее время имеется тенденция на перенос обработки данных с централизованных систем (облачные вычисления) на уровень ЦОД распределения (туманные вычисления) и далее на уровень конечных устройств (граничные вычисления). Таким образом, появляется возможность задействовать существенные вычислительные мощности, которые ранее не использовались, либо использовались крайне ограничено.

Распределение вычислений по множеству вычислительных узлов позволяет говорить о параллельных вычислениях, которые в ситуации использования независимых вычислительных устройств являются распределенными системами. В зависимости от организации взаимосвязи между вычислительными узлами также говорят о кластерных системах: использование ИВ в качестве распределенной информационной системы позволяет построить слабосвязанную кластерную систему.

Устройства ИВ имеют свою специфику и ряд ограничений, которые для организации процесса решения вычислительных задач не позволяют их использовать подходы, которые используются в классических параллельных и распределенных вычислительных системах.

Исходная вычислительная задача для дальнейшей обработки должна быть представлена в виде вычислительного графа, который с учетом связи его вершин преобразуется в последовательность заданий, которые могут быть выполнены на

вычислительных узлах. Распределительный узел в соответствие с алгоритмом планировщика заданий должен назначать (сопоставлять) задания вычислительным узлам, состояние которых изначально неизвестно и постоянно изменяется. Для построения метода назначения заданий вычислительным узлам рассмотрены методы машинного обучения, которые позволяют в условиях постоянного изменения характеристик вычислительных узлов организовать процесс назначения заданий. В результате анализа методов машинного обучения был выбран метод обучения с подкреплением, который в наибольшей степени подходит для организации процесса назначения заданий вычислительным узлам в РИСИВ.

Глава 2. Разработка модели системы назначения заданий на основе машинного обучения с подкреплением

В основе системы назначения заданий по вычислительным узлам распределенной информационной системы лежит модель машинного обучения - обучение с подкреплением (ОП). Исходная модель обучения с подкреплением предложена отечественным ученым-математиком М.Л. Цетлиным в 1961 году [93], развитие модель получила в работе Р. Саттона, опубликованной в 1988 году, в которой, в частности, рассматривается разновидность, называемая проблемой многорукого бандита (ПМБ, n-armed or multi-armed bandit problem - MABP). Особенностью ПМБ является представление окружающей среды в виде игровых слот-машин, так называемых «одноруких бандитов», у которых имеется не один рычаг, а множество.

Согласно [90] обучение с подкреплением (RL, reinforcement learning) — это такое обучение, при котором определяется то, что необходимо делать, «как отображать ситуацию в действие», чтобы получить максимальное значение сигнала поощрения.

В общем случае, задача *назначения заданий* - это процедура, которая позволяет программу, предназначенную для выполнения на параллельной или распределенной системе и представленную в виде последовательности элементарных заданий, выполнить на распределенной информационной системе, которая основана на инфраструктуре ИВ. Вычислительные узлы, которые представлены устройствами ИВ, выступают в роли вычислителей, которые получают задания и должны их выполнить. Центральный распределительный узел выполняет работу по разделению исходного алгоритма на задания, по отправке заданий на вычислительные узлы и по сбору выполненных заданий, а также отслеживает состояние каналов связи и состояние вычислительных узлов.

2.1 Постановка задачи назначения заданий

После начала выполнения программы на распределенной системе осуществляется ее преобразование к последовательности заданий, которые будут выполняться вычислительными узлами. Как указывалось ранее, построенная на базе инфраструктуры ИВ, распределенная информационная система представляет собой параллельную распределенную систему, с той разницей, что в классических параллельных системах вычислительные узлы являются гомогенными и коммутационная сеть не меняет своих характеристик в плане задержки и пропускной способности.

Приведем формальное определение задачи назначения заданий на вычислительные узлы ИВ, основанное на задаче оптимального отображения алгоритма на архитектуру параллельной вычислительной системы с использованием графовой модели [91, 77].

Пусть вычислительный алгоритм представляется в виде ациклического графа,

$$\langle A,\ C^A \rangle, \tag{2.1}$$

в котором $A=(a_1, a_2, \ldots, a_{|A|})$ - это вершины графа, которые соответствуют составным частям алгоритма (операторам), или *заданиям;*

а $C^A=(c^A_{i,j},\ i, j\in \mathbb{N}_{|A|},\ j\neq i)$ - ребра графа, соответствующие информационным связям между заданиями, так что $c^A_{i,j}$ - это объем информации в байтах, который передается от задания a_i к заданию a_j, $\mathbb{N}_{|A|}$ - упорядоченное множество натуральных чисел от 1 до $|A|$.

С точки зрения программиста-разработчика задание может являться достаточно крупным фрагментом исходного кода программы, предназначенной для выполнения на параллельной системе: подпрограмма, функция, процедура, модуль и т. п.

Пусть распределенная вычислительная система представлена в виде графа

$$\langle P, C^P \rangle, \tag{2.2}$$

в котором $P=(p_1, p_2, \ldots, p_{|P|})$ - это вершины графа, соответствующие вычислительным узлам;

а $C^P=(c^P_{i,j},\ i, j\in\mathbb{N}_{|P|},\ j\neq i)$ - ребра графа, соответствующие каналам связи между узлами, где $c^P_{i,j}$ - пропускная способность линии связи между вычислительными узлами p_i и p_j без учета задержек, $\mathbb{N}_{|P|}$ - упорядоченное множество натуральных чисел от 1 до $|P|$.

Задачу назначения (сопоставления) заданий можно определить как отображение графа $\langle A,\ C^A\rangle$ на граф $\langle P, C^P\rangle$.

Рассматривая $|A|\times|P|$ вводится *отображающая матрица:*

$$Z=(z_{i,j},\ i\in[1,\mathbb{N}_{|A|}],\ j\in[1,\mathbb{N}_{|P|}]). \qquad (2.3)$$

Отображающая матрица является бинарной, то есть её компоненты могут принимать только два значения:

- если задание a_i назначено узлу p_j, то $z_{i,j}=1$;

- если задание a_i не назначено узлу p_j, то $z_{i,j}=0$.

Решение задачи назначения заданий вычислительным узлам будет определяться критерием оптимальности отображения μ над матрицей Z, а решением задачи является матрица Z^*, такая что

$$\min_{Z\in D_Z}\mu(Z)=\mu(Z^*), \qquad (2.4)$$

где D_Z - множество допустимых отображающих матриц.

Данная задача является NP-полной, то есть ее решение может быть получено на основе полного перебора. А с учетом того, что характеристики вычислительных узлов распределенной информационной системы на основе ИВ постоянно меняются, то приходится решать эту задачу в каждый момент времени (возможно наличия нескольких решений в каждый конкретный момент времени, если одному вычислительному узлу могут быть назначены несколько заданий), что приведет к тому, что распределенная вычислительная система будет находится в постоянном решении задачи оптимизации, но никогда не перейдет к решению заданной вычислительной задачи. Использование метода, основанного на

использовании машинного обучения с подкреплением, позволяет решить задачу распределения (близко к оптимальному в случае, когда время $t \to \infty$).

2.2 Общая структура модели машинного обучения с подкреплением

В центре фокусирования машинного обучения с подкреплением находится реализация способности самостоятельного обучения и принятия решений объектом посредством его взаимодействия с окружающей средой. Соответственно, ключевыми элементами модели машинного обучения с подкреплением являются:

- *агент* (agent). Описывает некий объект, который для принятия решений и формирования стратегии поведения, взаимодействует со средой.

- *окружающая среда* (environment). Является внешней по отношению к агенту и описывает всё то, с чем взаимодействует агент.

- *действия* (actions). Описывают совокупность действий, которые выполняет агент при взаимодействии с окружающей средой. А именно, отправляемые агентом действия в окружающую среду.

- *состояния* (states). Данный элемент направлен на учёт состояний окружающей среды. Каждое действие агента переводит окружающую среду в некое состояние.

- *сигналы вознаграждения* (rewards). Описывают реакции окружающей среды на действия агента, формируют его опыт. Могут быть отрицательными или положительными.

Агент взаимодействует с окружающей средой и получает от неё реакции непрерывно. Агент оказывает на среду различные действия, в то время как на каждое из действий окружающая среда реагирует. Агент, в общем случае, не имеет никакой информации об окружающей среде и ему не сообщается, какое действие следует предпринимать на каждом шаге.

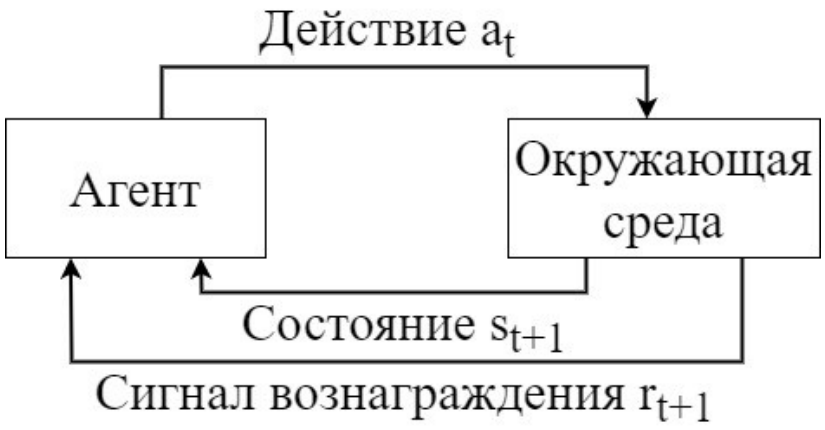

Рисунок 2.1. Модель машинного обучения с подкреплением

Под воздействием агента окружающая среда может изменяться, поэтому каждый раз агент на одно и тоже действие может получать разное значение вознаграждения. Если среда не меняется, то одно и тоже действие агента будет давать всегда одно и тоже вознаграждение.

Согласно Рисунку 2.1 модели машинного обучения с подкреплением агент отправляет в каждый дискретный момент времени - t окружающей среде своё действие a_t , в ответ на которое, окружающая среда оказывается в новом состоянии s_{t+1} и формирует ответную реакцию r_{t+1}, которая является новой для агента и выражена в виде некоторого числового значения - $r_{t+1} \in \mathbb{R}$.

При этом непрерывное взаимодействие агента с окружающей средой описывается в каждый t из последовательности, как $t = 0, 1, 2, 3, \ldots$, последующий временной шаг описывается как $t+1$.

Каждое состояние окружающей среды в каждый t представлено s_t и принадлежит множеству возможных состояний - $s_t \in S$. Агент на основе всех возможных состояний окружающей среды в конкретный t выбирает некое действие a_t из множества всех действий $a_t \in A(s_t)$, которые возможны в данном s_t в этот момент времени, после чего окружающая среда оказывается в новом состоянии, а агент получает новые данные об окружающей среде. Тем самым, агент обучается и накапливает опыт.

Опыт позволяет агенту принимать дальнейшие решения в отношении окружающей среды — выбирать дальнейшие действия, что означает формирование стратегии поведения агента в зависимости от имеющегося опыта.

Стратегия агента описывается как:

$$\pi_t(a|s)\,. \tag{2.5}$$

Указанное π_t является вероятностью того, что $a_t=a$, если $s_t=s$. Иными словами, каждый временной шаг агент осуществляет отображение из множества состояний на множество вероятностей выбора каждого действия.

Методы обучения с подкреплением позволяют определять, как агент может менять свою стратегию в зависимости от имеющегося опыта. Стратегии агентов направлены на максимизацию суммарного вознаграждения на длительном промежутке времени.

Согласно модели, агент должен иметь:

- цель (или множество целей), которые связаны с состоянием среды;

- возможность выполнять некоторые действия, которые могут влиять на состояние окружающей среды;

- возможность воспринимать состояние окружающей среды.

Показанная на Рисунке 2.1 схема несущественно отличается от схемы «агент-окружающая среда», представленной для архитектуры нейронных сетей в [94], но основное отличие лежит в наличии *оценочной обратной связи.* То есть агент использует обучающую информацию, которая только оценивает его действия, но никак не указывает на то, правильное это действие или нет, то есть отсутствует инструктивная обратная связь, характерная для обучения с учителем — обучение по предъявляемым некоторой третьей (информированной) стороной образцам. Обучение с учителем в общем случае не позволяет реализовать такое же обучение, как в модели обучения с подкреплением, так как невозможно получить набор образцов поведения, которые были бы корректными и покрывали все возможные ситуации, в которых может оказаться агент.

В модели машинного обучения с подкреплением используется дискретное время, то есть процесс обучения агента можно представить как множество дискретных состояний, между которыми осуществляются переходы. Окружающая среда также является дискретной системой, имеющей состояния и переходы между ними (среда не обязана быть дискретной системой, а может быть и непрерывной, но с учетом того, что агент получает от нее ответы только в дискретные моменты времени, то тогда окружающая среда с точки зрения агента всегда дискретная).

Стоит отметить, что в идеальном случае переход агента и окружающей среды из состояния в другое состояние определяется только сигналами действия и вознаграждения, но стоит уточнить, что окружающая среда не управляется агентом. В общем случае, окружающая среда может иметь и другие воздействия (помимо агента), на которые реагирует и о которых агент не имеет никакого представления.

В результате взаимодействия с окружающей средой агент может строить свою модель этой среды, на основе которой может работать алгоритм предикции для более эффективной работы агента. Также, если имеются какие-либо особенности или закономерности поведения среды, то существует целый ряд разновидностей моделей обучения с подкреплением (например, Q-learning [95], TD-learning [90, 96] и т. п.), которые могут повысить качество поведения агента.

Применительно к РИСИВ, как было сказано ранее в главе 1, в роли окружающей среды выступают вычислительные узлы, изменение характеристик которых в общем случае не имеет какой-либо закономерности. Вычислительные узлы являются независимыми друг от друга, поэтому выполнение заданий на одном узле не влияет на другие узлы. Также после выполнения одного задания вычислительный узел ,по сути, поступает в пул свободных узлов и выполненное задание не оказывает влияние на дальнейшее функционирование этого узла. Поэтому, учитывая указанные условия, наилучшей моделью обучения с подкреплением является модель с ε-жадной (ε-greedy) стратегией (формально:

вычислительные узлы представляют собой матроид, а жадные алгоритмы на матроиде дают оптимальное решение).

Следуя из вышесказанного, стратегией поведения агента при обучении с подкреплением является активное воздействие на среду путем тестирования различных действий и получения ответной реакции. Данный подход подобен эвристическим методам оптимизации функций, когда, например, в эволюционных алгоритмах осуществляется постоянная оценка поведения функции практически методом «проб и ошибок» [91].

Модель машинного обучения с подкреплением сталкивается с дилеммой «изучения/применения» (исследования/эксплуатации: агент применяет только те действия, о которых он знает и которые дают максимальное вознаграждения, но также он должен изучать новые действия (которые, возможно, не дают большого вознаграждения), которые могли бы в будущем сделать лучший выбор и получить большее вознаграждение.

Несмотря на то, что агент не имеет общей информации о среде, он должен продолжать действовать. Обучение с подкреплением позволяет получить полную картину поведения целенаправленного агента, который взаимодействует с неопределенной окружающей средой.

2.3 Цель агента и выгода

Основной целью агента является максимизация суммарного вознаграждения, получаемого от окружающей среды. Действия агента могут оказывать влияние не только на текущее значение вознаграждения, но также на возникающую ситуацию, а следовательно, через нее на все последующие значения вознаграждений.

Цель агента определяется значением вознаграждения (сигнала вознаграждения) — специального сигнала, получаемого как ответную реакцию

окружающей среды. В каждый момент времени t вознаграждение определяется некоторым числом $r_t \in \mathbb{R}$.

В силу того, что агент не знает, какое вознаграждение может быть получено в каждый момент времени, то его целью будет являться максимизация не текущего вознаграждения, а именно суммарного значения вознаграждения, получаемого в течении длительного интервала времени.

Сигнал вознаграждения должен указывать то, чего агенту необходимо достичь, но при этом он не должен сообщать агенту априорную информацию о том, как достичь желаемого. Например, если агент осуществляет поиск максимума некоторой функции, то от окружающей среды (описанной указанной функцией) получает только значение этой функции, но не информацию о направлении её роста, которую он мог бы применить для выполнения следующих шагов. Иначе это будет получением вознаграждения за достижение промежуточных целей, то есть агент научится достигать промежуточную цель, при этом никогда не достигнет главной цели. В примере с поиском максимума функции агент может обнаружить локальный максимум и на этом остановиться, никогда не найдя глобальный максимум.

Также обязательном условием является полная независимость окружающей среды от агента. Агент не должен иметь цель, которую он мог бы контролировать (даже частично), в противном случае он сможет самостоятельно устанавливать получаемое значение сигнала вознаграждения (аналогично тому, как он произвольным образом может менять свои действия).

Выполняемая агентом последовательность взаимодействий с окружающей средой называется *заданием* — полное описание характеристик окружающей среды, или один конкретный вариант задачи обучения с подкреплением. В зависимости от того, возможно ли разбить задания на повторяющиеся (одинаковые) последовательности, выделяют:

- задания, состоящие из эпизодов;

- непрерывные задания.

Задания, состоящие из эпизодов

Обозначив через $r_{t+1}, r_{t+2}, r_{t+3}, \ldots$ последовательность значений сигналов вознаграждения, полученных после некоторого временного шага t, необходимо определить, какие элементы последовательности необходимо максимизировать.

Поскольку цель агента заключается в максимизации суммарного значения всех вознаграждений, то для общего случая необходимо найти максимум ожидаемой выгоды R_t, определяемой как функция на последовательности вознаграждений:

$$R_t = r_{t+1} + r_{t+2} + r_{t+3} + \ldots + r_T, \quad (2.6)$$

где T - завершающий временной шаг.

Формула (2.6) будет иметь значение в ситуациях, где существует возможность естественным образом определить завершающий временной шаг. Иначе говоря, когда взаимодействие агента с окружающей средой можно представить в виде конечной последовательности. Такая последовательность называется *эпизодом*, а каждый эпизод заканчивается *терминальным состоянием*. После достижения терминального состояния система «агент — окружающая среда» возвращается к начальному состоянию (или выборке из стандартного распределения начальных состояний).

Последовательность эпизодов составляют задания, называемое *заданием, состоящим из эпизодов*. В таких заданиях необходимо различать множество всех состояний S от множества всех состояний, включая терминальное состояние S^+.

Непрерывные задания

В том случае, если взаимодействие агента с окружающей средой невозможно представить в виде последовательности эпизодов, когда невозможно осуществить фиксацию завершающего момента времени (то есть отсутствует терминальное состояние), то такие задания называются *непрерывными заданиями*.

Поскольку для непрерывных заданий завершающий временной шаг $T=\infty$, то, соответственно, получаемая агентом выгода будет бесконечной. Для такого случая формула (2.6) становится неприменимой, поэтому вводится понятие *дисконтирования*, когда текущее вознаграждение имеет большую ценность для агента, чем вознаграждение в будущем. Для этого вводится коэффициент приведения (коэффициент дисконтирования, discount rate) γ, который определяет текущую ценность будущих вознаграждений. Коэффициент приведения находится в диапазоне $[0, 1]$. Ценность будущего вознаграждения, полученного после k временных шагов будет отличаться от текущего в γ^{k-1} раз.

Таким образом, вместо максимизации суммы всех вознаграждений (2.6) для непрерывных заданий, агент максимизирует сумму приведенных вознаграждений, которые он получит в будущем, то есть агент выбирает действия a_t таким образом, чтобы максимизировать ожидаемую приведенную выгоду:

$$R_t=r_{t+1}+\gamma r_{t+2}+\gamma^2 r_{t+3}+\ldots=\sum_{k=0}^{\infty}\gamma^k r_{t+k+1}, \qquad (2.7)$$

где γ - коэффициент приведения ($0\leqslant\gamma\leqslant 1$).

При $\gamma=1$ из (2.7) получается формула (2.6) с суммой бесконечного ряда, которая также бесконечна.

Когда $\gamma<1$, то сумма бесконечного ряда будет иметь конечное значение до тех пор, пока последовательность $\{r_k\}$ будет ограниченной.

При $\gamma=0$ формула (2.7) упрощается только до ближайшего вознаграждения:

$$R_t=r_{t+1}, \qquad (2.8)$$

то есть агента интересует только максимизация ближайшего вознаграждения, в этом случае цель агента — выбирать действия a_t так, чтобы максимизировать r_{t+1} (в данном случае говорят, что агент «близорукий»).

Если стратегией агента было бы влияние только на ближайшее вознаграждение и не влияло бы на будущие вознаграждения, то такой агент

максимизируя отдельно каждое ближайшее вознаграждение не сможет максимизировать всю приведенную выгоду.

Стремясь максимизировать только ближайшие вознаграждения, агент ограничивает свое воздействие на будущие вознаграждения, что приводит к снижению общей выгоды. По мере увеличения коэффициента приведения γ (при приближении к 1), агент становится более дальновидным и значимость будущих вознаграждений увеличивается.

2.4 Определение ценности действия

Ценность конкретного действия определяется как общая сумма всех вознаграждений, которые агент может получить в будущем, начиная от этого данного действия. Таким образом, если фактическую ценность действия a определить как $Q^*(a)$, а предполагаемое значение этого действия на временном шаге t обозначить $Q_t(a)$, то в случае выбора действия a к моменту t ровно k_a раз получим последовательность вознаграждений $r_1,\ r_2,\ \ldots,\ r_{k_a}$.

Ценность действия a можно оценить следующим образом:

$$\begin{cases} Q_t(a)=0, \ \textit{если} \ \ k_a=0; \\ Q_t(a)=\dfrac{1}{k_a}\cdot\sum\limits_{i=1}^{k_a} r_i, \ \textit{если} \ \ k_a\neq 0. \end{cases} \tag{2.9}$$

В случае бесконечного количества временных шагов и бесконечного количества выпадений действия a, то есть при $k_a\to\infty$, то ценность действия будет стремиться к фактической ценности действия

$$Q_t(a)\to Q^*(a). \tag{2.10}$$

Как отмечалось в пп. 2.3 в обучении с подкреплением выделяется два вида заданий: задания, состоящие из эпизодов, и непрерывные задания.

Несмотря на существенное отличие обоих видов заданий, возможно свести их к одной схеме обозначений. Вместо одной непрерывной последовательности временных шагов, можно рассматривать их как серии эпизодов, каждый из

которых состоит из конечного множества временных шагов (в каждом эпизоде нумерация шагов начинается с нуля). В данном случае можно применять следующие обозначения, в которых указывается одновременно не только временной шаг, но и также номер эпизода: например, состояние в момент времени t будет иметь обозначение S_t, в то время как для состояния в момент t в некотором i-ом эпизоде — обозначение $S_{t,i}$ (и так для всех обозначений). Но поскольку описания всех эпизодов являются идентичными, то в данном случае индекс i можно опускать.

Для каждого вида заданий возможно отдельно определить значение выгоды. Но если после последнего состояния у конечного эпизода добавить бесконечно количество состояний, который будут давать нулевое вознаграждение, либо (как предложено в [90]) ввести поглощающее состояние, переход из которого будет только в само себя и которое порождает только нулевые вознаграждения (Рисунок 2.2).

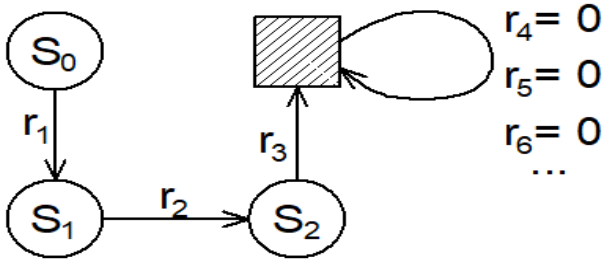

Рисунок 2.2. Поглощающее состояние

Поглощающее состояние позволяет получать ту же самую выгоду, что и при суммировании бесконечной последовательности. Таким образом, в общем случае выгода описывается уравнением

$$R_t = \sum_{k=0}^{T} \gamma^k r_{t+k+1}, \tag{2.11}$$

где могут быть либо $T = \infty$, либо $\gamma = 1$ (но не одновременно).

Стационарная задача

В стационарной среде реакция среды на действия агента не изменяется со временем, поэтому оценка вознаграждения каждого действия со временем не меняется.

При увеличении количества временных шагов растет также и требования к памяти вычислительной системы, так придется накапливать все значения вознаграждений за все время работы.

Определим для действия a среднее значение его k вознаграждений как Q_k . Тогда после получения следующего $(k+1)$ -го вознаграждения будем иметь:

$$Q_{k+1}=\frac{1}{k+1}\sum_{i=1}^{k+1} r_i=\frac{1}{k+1}(r_{k+1}+\sum_{i=1}^{k} r_i)\,. \tag{2.12}$$

Так как $Q_k=\frac{1}{k}\sum_{i=1}^{k} r_i$, то получим сумму $\sum_{i=1}^{k} r_i=k\,Q_k$ и подставим ее в формулу (2.12), получаем

$$Q_{k+1}=\frac{1}{k+1}(r_{k+1}+k\,Q_k)=\frac{1}{k+1}(r_{k+1}+[k\,Q_k+Q_k]-Q_k)=$$
$$=\frac{1}{k+1}(r_{k+1}+Q_k[k+1]-Q_k)=Q_k+\frac{1}{k+1}[r_{k+1}-Q_k].$$

Выполнив замену

$$\frac{1}{k+1}=\frac{1}{k_a+1}=\alpha_{k+1}(a)=\alpha\,, \tag{2.13}$$

Получаем

$$Q_{k+1}=Q_k+\alpha\,[r_{k+1}-Q_k]\,, \tag{2.14}$$

где α - значение длины шага.

Данная рекуррентная формула может быть записана в виде правила корректировки:

Новая оценка ← *Прежняя оценка* + *Длина шага* × *Ошибка*,

где *Ошибка* ← [*Цель* − *Старая оценка*].

Ошибка уменьшается с каждым шагом приближения к *Цели*.

Параметр *Длина шага* меняется при каждом временном шаге.

Нестационарная задача

Нестационарная задача встречается чаще, чем стационарные. Также можно выделить класс задач, называемые существенно нестационарные, когда среда изменяется очень сильно. В таких задачах отклик, получаемый от среды, в некий момент времени лучше отражает текущую ситуацию, чем отклик от среды, полученный в некоторый более ранний период времени.

Из формулы (2.13) видно, что значение шага α зависит (обратно пропорционально) от количества раз, когда было выбрано некоторое действие. Если же задать постоянное значение шага α ($0<\alpha\leqslant 1$), то становится возможным учесть нестационарность окружающей среды:

$$Q_{k+1}=Q_k+\alpha\left[r_{k+1}-Q_k\right]. \qquad (2.15)$$

В результате получается значение Q_k, которое для начальной оценки Q_0 и для предыдущих значений будет средневзвешенным:

$$\begin{aligned}
&Q_k=Q_{k-1}+\alpha\left[r_k-Q_{k-1}\right]=Q_{k-1}+\alpha r_k-\alpha Q_{k-1}=\\
&=\alpha r_k+(1-\alpha)Q_{k-1}=\alpha r_k+(1-\alpha)\alpha r_{k-1}+(1-\alpha)^2 Q_{k-2}=\\
&=\alpha r_k+(1-\alpha)\alpha r_{k-1}+(1-\alpha)^2 Q_{k-2}+\ldots+(1-\alpha)^{k-1} r_1+(1-\alpha)^k Q_0=\\
&=(1-\alpha^k)Q_0+\sum_{i=1}^{k}\alpha(1-\alpha)^{k-i} r_i.
\end{aligned}$$

Полученное значение является средневзвешенным, так как для суммы весов выполняется следующее:

$$(1-\alpha)^k+\sum_{i=1}^{k}\alpha(1-\alpha)^{k-i}=1, \qquad (2.16)$$

откуда видно, что значение веса $\alpha(1-\alpha)^{k-i}$ для вознаграждения r_i зависит от величины $k-i$, то есть от того, сколько шагов назад было получено это вознаграждение. Так как величина $(1-\alpha)<1$, то вес вознаграждения r_i уменьшается экспоненциально с ростом числа полученных вознаграждений (поэтому такое среднее называется экспоненциальным средним).

В некоторых ситуациях имеет смысл изменять значение шага: пусть $\alpha_k(a)$ - величина шага, используемая для расчета полученного после k-го выбора действия a вознаграждения.

Если $\alpha_k(a)=\frac{1}{k}$, то будет получен метод среднего выборочного, гарантирующего сходимость к истинному значению ценности действия (согласно закону больших чисел).

Однако сходимость последовательности $\{\alpha_k(a)\}$ не гарантирована для всех возможных вариантов. Для обеспечения сходимости последовательности с вероятностью 1 должны быть выполнены условия (получаемые из результатов теории стохастической аппроксимации):

$$\sum_{k=1}^{\infty}\alpha_k(a)=\infty, \tag{2.17}$$

$$\sum_{k=1}^{\infty}\alpha_k^2(a)<\infty. \tag{2.18}$$

Условие (2.17) гарантирует, что шаги имеют достаточной большую величину, чтобы на процессе обучения не сказывались никакие начальные условия или случайные флуктуации. Условие (2.18) гарантирует, что шаги достаточно малы, чтобы обеспечивалась сходимость.

Например, оба условия выполняются для $\alpha_k(a)=\frac{1}{k}$. Для случая постоянного шага $\alpha_k(a)=\alpha$ не выполняется условие (2.18), что показывает, что процесс не сходится, но значения продолжают изменяться в ответ на последние полученные вознаграждения.

Нестационарные задачи имеют также существенное отличие от стационарных, в них на результаты работы не оказывают влияние начальные оценки ценности действия.

2.5 Выбор следующего действия

Агент осуществляет выбор следующего действия на основании оценки ценности действия, которая подразумевает следующие стратегии поведения:

- жадная стратегия;
- ε - жадная стратегия;
- случайный выбор.

Жадная стратегия является наиболее простым вариантом поведения агента: агент каждый раз выбирает то действие, которое имеет наибольшее значение оценки ценности действия. То есть, на некотором временном шаге t следует выбирать такое действие a^*, что

$$Q_t(a^*) = \max_a Q_t(a). \tag{2.19}$$

В данном случае агент никогда не пытается исследовать среду с целью обнаружения некоторого действия, которое дало бы большее вознаграждение. При постоянной оценке ценности действий, когда в любой момент времени существует хотя бы одно действие, имеющее наибольшую предполагаемую ценность действия, то такое действие будет жадным действием. В стационарной среде, когда все значения вознаграждений известны, такое поведение является наиболее правильным, но для нестационарной среды такое поведение не всегда даст лучший результат.

Противоположностью жадной стратегии является стратегия, основанная на случайном выборе, когда агент на каждом шаге выбирает случайное значение. В данном случае, независимо от полученного результата, агент всегда исследует окружающую среду, но никогда не использует полученную информацию для планирования своего дальнейшего поведения.

Случайное поведение может давать результат в нестационарной среде, когда параметры среды меняются хаотично, но следует отметить, что другие стратегии также могут давать аналогичный результат. Жадная стратегия хороша, когда среда неизменна.

Для нестационарной среды наиболее приемлемым вариантом является ε-жадная стратегия, где вводится вариативный коэффициент ε, задающий вероятность выбора действия. При такой стратегии агент попеременно выбирает то жадное действие, то нежадное. ε-жадная стратегия позволяет разрешить дилемму между исследованием и эксплуатацией. При выборе жадного действия используется текущее значение о ценности действия (режим эксплуатации, exploitation), что позволяет максимизировать ожидаемое вознаграждение при выполнении текущего действия. При выборе нежадного действия агент имеет возможность оценить более точно ценность нежадного действия (режим исследования, или изучения, exploration). Такой режим не позволяет получить максимальное вознаграждение при выполнении текущего действия, но может привести к увеличению общего вознаграждения в течении длительного времени, то есть привести к максимизации суммы вознаграждений — своей цели.

Проблема многорукого бандита

Реализация ε-жадной стратегии представлена в проблеме (алгоритме) многорукого бандита (МРБ, Multi-armed bandit, n-armed bandit), где окружающая среда представляется в виде игрового автомата — слот-машины (также называемым «одноруким бандитом»), которая имеет не один, а множество рычагов, выбирая каждый из них игрок (агент) получает различное вознаграждение.

В алгоритме МРБ агент учится оптимизировать фиксированную, но неизвестную функцию через ряд последовательных взаимодействий с окружающей средой.

В алгоритме агент выполняет некоторое конкретное действие a из множества всех действий, доступных агенту - A. На каждое действие агента $a \in A$, имеется сигнал вознаграждения r, который принадлежит множеству сигналов вознаграждения R, то есть $r \in R$.

Исходя их этого, сопоставление реакции каждому произвольному действию в конкретный момент времени t описывается соотношением:

$$q_*(a)=E[R_t|A_t=a], \tag{2.20}$$

где $q_*(a)$ - значение ценности действия при ожидаемом вознаграждении, получаемом при выборе произвольного действия в некий момент времени.

$R^a(t)=P(r|a)$ - описывает неизвестное распределение вероятности награды при условии выбора действия a.

Для поиска оптимального (субоптимального) решения в алгоритме необходимо обеспечить баланс между процессом исследования и процессом эксплуатации, алгоритм МРБ использует жадную и ε-жадную стратегию.

При жадной стратегии алгоритмом не рассматриваются действия, которое не максимизируют значение вознаграждения, то есть выбирается только то, которое является наибольшим из известных q. В общем случае, с точки зрения обучения с подкреплением, это является разумной стратегией, так как даёт максимальное вознаграждение на текущий момент времени, но при этом не учитывает того, что возможные вознаграждения для других действий могут увеличиться со временем. Такое поведение алгоритма напоминает исследованное в университете Stanford поведение детей в эксперименте с зефиром, когда ребенок предпочитал выбрать небольшое вознаграждение «прямо сейчас», чем большее вознаграждение чуть позднее [97].

При ε-жадной стратегии, коэффициент ε находится в диапазоне $\varepsilon\in[0;1]$ и регулирует стратегию поведения агента — действовать полностью «жадно» и постоянно выбирать действие с максимальным значением вознаграждения или периодически выбирать иные действия. Регулирование стратегии поведения по выбору каждого действия задаётся соотношением:

$$a_t=\begin{cases} a_t^*, & \text{с вероятностью } 1-\varepsilon; \\ \text{случайное действие}, & \text{с вероятностью } \varepsilon. \end{cases} \tag{2.21}$$

При $\varepsilon=0$ стратегия становится жадной, то есть алгоритм работает только в режиме эксплуатации и агент никогда не пытается исследовать новое действие;

при $\varepsilon = 1$ алгоритм заставляет агента каждый раз менять свое текущее действие на окружающую среду случайным образом.

С учетом вышеприведенного указанную модель и алгоритм МРБ можно использовать для назначения заданий по вычислительным узлам в распределенной информационной системе, где в роли агента выступает распределяющий узел, а в роли окружающей среды – вычислительные узлы, которые выполняют полученные задания и возвращают результат агенту.

2.6 Выводы по второй главе

Проведя формальную постановку задачи назначения заданий вычислительным узлам вычислительной системы, была рассмотрена общая структура модели машинного обучения с подкреплением.

Машинное обучение с подкреплением позволяет представить распределенную вычислительную систему в виде агента и окружающей среды. Агент, пытаясь достичь своей цели, воздействует на среду, от которой получает обратную связь в виде сигнала вознаграждения. Поскольку структура окружающей среды агенту неизвестна, то для этого используются различные стратегии поведения, которые с одной стороны должны позволить ему исследовать окружающую среду, а с драгой стороны — получить максимальное вознаграждение на длительном промежутке времени. Подробное рассмотрение цели агента, выгоды, определение ценности действия и стратегии выбора агентом следующего действия позволяют получить уравнения, которые могут использоваться при разработке метода назначения заданий вычислительным узлам в РИСИВ.

В следующей главе будет рассмотрен метод и алгоритмы, которые позволяют реализовать функциональность модели машинного обучения с подкреплением на распределяющем узле РИСИВ.

Глава 3. Метод назначения заданий в распределенной информационной системе Интернета вещей

Основным элементом задачи распределения заданий является определение вычислительного узла, на котором должно оказаться задание. Главной особенностью системы распределения на основе обучения с подкреплением является то, что изначально распределяющий узел не имеет никакой информации о структуре вычислительной системы и предварительно не выделяет потенциально предпочтительный узел [98]. Лишь через какое-то время распределяющий узел получает больше информации, но она по-прежнему не может считаться абсолютно надежной, так как структура РИСИВ может меняться.

Распределяющий узел получает данную информацию непрерывно в процессе функционирования РИСИВ и непрерывно обновляет её с учётом изменения структуры РИСИВ, тем самым, актуализируя и выделяя предпочтительный узел в зависимости от изменения структуры РИСИВ.

Используя формальную постановку задачи распределения и математическое описание модели обучения с подкреплением, в этой главе предлагается преобразовать модель ОП в алгоритм, который сможет решать задачу распределения заданий, и должен быть реализован программно и быть размещённым на распределяющем узле в виде программы.

Также в данной главе предлагается добавить модификацию к алгоритму с целью его масштабирования путём предоставления возможности самому устройству ИВ исполнять роль распределяющего узла в РИСИВ.

Поскольку особенностью модели ОП в качестве основы метода распределения заданий является именно возможность отправки задания на удаленный вычислительный узел и получения от него результата, то в данном случае будет излишним рассматривать вопросы безопасности, аутентификации, целостности и потери сообщений.

3.1 Этапы преобразования программы

Процесс выполнения вычислительного задания на РИСИВ состоит из следующих этапов:

- формальное определение задачи;
- разработка алгоритма решения вычислительной задачи;
- анализ алгоритма с целью выявления участков для параллельной обработки;
- выполнение этапа «исследование» алгоритма распределения заданий;
- выполнение этапа «эксплуатации» алгоритма распределения заданий;
- получение ответов от вычислительных узлов;
- формирование итогового результата решения вычислительной задачи.

Этапы выполнения вычислительной задачи не обязательно должны быть четко выражены. Поскольку деление на этапы является условностью, то возможно как объединение этапов, так и разделение части их функциональности с соседними этапами.

Формальное определение задачи позволяет сформировать требования к входным и выходным данным, а также закономерности (формулы) преобразования данных.

Разработка алгоритма задачи позволяет определить шаги (дискретные), которые необходимо выполнить для решения поставленной задачи, определить спецификацию будущего программного обеспечения.

Этапы формального определения задачи и разработки алгоритма свойственны императивной парадигме программирования. При использовании других парадигм программирования (декларативной, функциональной и т. п.) начальные этапы могут отличаться, но так как любые парадигмы можно свести к императивной [99], то можно считать первые два этапа универсальными для любого подхода. Этапы определяют в данной работе задания.

Анализ алгоритма с целью выявления участков для параллельной обработки позволяют выделить фрагменты, которые могут быть выполнены на вычислительных узлах. Стоит отметить, что в части, касаемо выявления программных участков с параллельной структурой, в распределенных вычислительных системах (в том числе с вычислительными узлами на основе устройств ИВ) нет отличий от классических алгоритмов [77]. Для данного этапа для РИСИВ характерны следующие моменты:

а) найти участки алгоритма, которые можно исполнять параллельно;

б) параллельные участки должны быть оформлены в виде вычислительных примитивов (подпрограмм: функций, процедур);

в) разработать структуры данных, которые возможно передавать по вычислительной сети (сериализация/десериализация, маршалинг/демаршалинг);

г) определить формат сообщения сигнала вознаграждения, а также определить элементы, входящие в эту интегральную характеристику.

Выполнение этапа «исследование» алгоритма назначения заданий позволяет выполнить начальную настройку РИСИВ путем опроса всех доступных вычислительных узлов и сбора статистики об их состоянии (сигнал вознаграждения).

На этапе выполнения этапа «эксплуатация» алгоритма назначения заданий осуществляется непосредственные назначения и отправка заданий на вычислительные узлы, а затем получение обработанных данных. В случае возникновения ситуации, когда вычислительный узел недоступен, задание помечается как невыполненное и снова ставится в очередь на отправку к вычислительным узлам.

На этапе получения ответов от вычислительных узлов осуществляется сбор обработанных данных, а также возможно получение с ними значений сигналов вознаграждений от вычислительных узлов.

Затем на последнем этапе - этапе формирования итогового результата - происходит обобщение полученных от вычислительных узлов результатов

выполнения заданий, то есть получение решения исходной вычислительной задачи.

3.2 Формирование вычислительных заданий

Создание программных средств для решения вычислительной задачи предполагает разделение исходной задачи на последовательность более мелких задач - заданий. Задание с точки зрения алгоритма вычислительной задачи и вычислительной системы является не просто частью исходной задачи, а является более сложной абстракцией, которая имеет большое количество свойство, не имеющих непосредственного отношения к исходной задаче, но служащих для функционирования алгоритма на распределенной или параллельной системе (Рисунок 3.1).

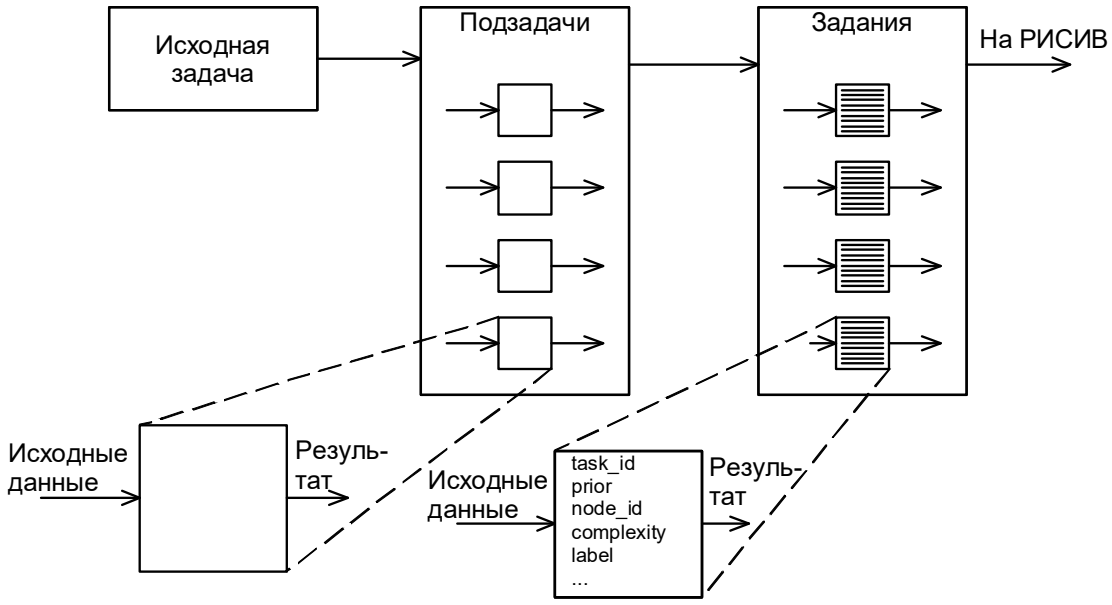

Рисунок 3.1. Формирование вычислительного задания

Таким образом, к свойствам заданий, имеющих отношение к исходной вычислительной задаче, можно отнести следующие [100, 101]:

- исходные данные (они, как правило, известны);

- результат (данные, которые необходимо найти или получить).

Указанные свойства определяют интерфейсы вычислительных примитивов. Результат также может содержать дополнительные данные, например, значение сигнала вознаграждения.

К свойствам заданий, которые необходимы для выполнения на распределенной вычислительной системе, можно отнести следующие:

- идентификатор задания (уникальный номер);
- приоритет задания;
- идентификатор узла;
- сложность задания;
- тело задания (наименование вычислительного примитива).

В РИСИВ передача вычислительным узлам тела задания существенно повысит нагрузку на вычислительную сеть, поэтому задание должно содержать лишь указание вычислительного примитива, который будет выполнен на удаленном вычислительном узле. То есть все вычислительные примитивы уже должны быть реализованы на вычислительных узлах, а по сети должны передаваться только данные для проведения расчетов и их результаты.

3.3 Модель вычислительного узла

Устройства ИВ представлены многообразием технологий и компонент, их составляющих. Тем не менее, для устройств ИВ может быть составлено общее описание, которое обеспечивает абстрагирование от особенностей реализации и снимает возможные ограничения на взаимодействие в РИСИВ.

Общее описание основывается на выделении общих характеристик [100, 101], присущих устройствам ИВ, а именно:

- данных о состоянии устройства ИВ: вычислительная мощность, объем ОЗУ, частота процессора, время отклика, время обслуживания, количество потребляемой энергии [101] и т. д.;

- данных о сети передачи данных среды, где функционирует устройство ИВ: пропускная способность канала, показатели потерь, скорость передачи данных и т. д.;

- данных о местоположении устройства ИВ и его перемещении: позиционирование в пространстве, расстояние до остальных вычислительных узлов в РИСИВ и т.д.

Исходя из выделенных характеристик, может быть задано соотношение вычислительного узла на способность назначения ему задания в РИСИВ с учётом модели машинного обучения с подкреплением в виде:

$$Reward = (State, Location, Network),$$

где $State$ - данные состояния, $Network$ - данные сети, $Location$ - данные местоположений и перемещений.

Поскольку вычислительные узлы РИСИВ выполняют роль окружающей среды, то согласно модели машинного обучения с подкреплением окружающая среда должна формировать реакцию в виде сигнала вознаграждения - *reward* . Соответственно, каждый вычислительный узел в РИСИВ формирует параметр $Reward$. Данный параметр вбирает в себя все характеристики, зависящие и независящие от вычислительного узла, и рассчитывается вычислительным узлом как итоговая интегральная характеристика от них. Вследствие, данный параметр отражает способность и готовность к назначению задания на вычислительный узел в РИСИВ.

Однако поскольку типы заданий могут варьироваться, следует учитывать также тип задания, решаемого вычислительным узлом. Тогда модель вычислительного узла задаётся в виде:

$$D = (ID, Label, Reward),$$

где D – устройство ИВ,

ID - идентификатор узла (уникальный номер);

$Label$ - тип решаемого задания (вычислительного примитива);

$Reward$ – значение сигнала вознаграждения.

Данная модель вычислительного узла РИСИВ описывает готовность конкретного вычислительного узла принять на исполнение очередное задание от вычислительного узла.

3.4 Функция вознаграждения

Во второй главе определено значение r_k - вознаграждение на k-ом временн*о*м шаге. Рассматривая отдельный вычислительный узел, как элемент окружающей среды в машинном обучении с подкреплением, можно утверждать, что значение вознаграждения определяется способностью узла успешно выполнить полученное задание [102].

Успех выполнения задания зависит от множества параметров как самого узла, так и параметрами других элементов РИСИВ. Параметры, зависящие от вычислительного узла:

- частота и режимы работы процессора;
- количество и скорость ОЗУ;
- скорость ПЗУ (если используется);
- архитектурные решения устройства ИВ (частоты шин, задержки сигналов, вид и тип кодирования и т. п.);
- установленное программное обеспечение устройства ИВ (операционная система, наличие/отсутствие дополнительного программного обеспечения и т. п.);
- количество датчиков, способы и последовательность их опроса;
- наличие дополнительного периферийного оборудования и его параметров.

Параметры, которые не зависят от вычислительного узла:

- ширина канала связи;
- пропускная способность канала связи;
- наличие помех в канале связи;
- использование сетевые протоколы;
- параметры распределяющего узла;

- удаленность устройства ИВ от элементов РИСИВ;

- нахождение устройства ИВ в статическом или динамическом состоянии.

Вышеуказанные списки параметров достаточно полны, но не исчерпывающие. Таким образом, вознаграждение является интегральной характеристикой узла, которая должна учитывать все указанные (а также неуказанные) параметры. В общем виде эти характеристики были обобщены в (reward). Тогда параметр Reward рассчитывается согласно (reward), как :

$$r=\sum_{i=1}^{n} p_i \lambda_i , \tag{3.1}$$

где p_i - значение i-го параметра;

λ_i - нормирующий (или весовой) коэффициент;

n - количество параметров, влияющих на значение сигнала вознаграждения.

Поскольку определить количество параметров и степень их влияния на общее значение вознаграждения зачастую не представляется возможным (также необходимо учесть, что параметры меняются в каждый момент времени), то в качестве функции вознаграждения в частном случае предлагается использовать функцию от времени оборота задания — по аналогии с временем оборота в сетях передачи данных (RTT - round-trip time). То есть можно определить время оборота задания как время, затраченное на отправку задания на вычислительный узел, время на выполнение задания на вычислительном узле и время получения ответа от вычислительного узла. Таким образом, время оборота можно вычислить по формуле:

$$t_{RTT}=t_{пол}-t_{отп} , \tag{3.2}$$

где $t_{пол}$ - время получения ответа от вычислительного узла;

$t_{отп}$ - время отправки задания распределяющим узлом.

Тогда вознаграждение будет определяться как функция:

$$r=r(t_{RTT}) , \tag{3.3}$$

что позволяет учитывать все параметры РИСИВ, которые могут влиять на успешность выполнения задания, так как время оборота также зависит от всех

этих параметров. При этом также вычислительный узел может отправлять свое значение вознаграждение, которое может быть использовано распределяющим узлом для расчета общего значения вознаграждения на основе времени оборота. Последнее может быть необходимо, например, если устройство ИВ оценивает количество заряда в своем аккумуляторе и уже «предполагает», что следующее задание может быть выполнено в режиме энергосбережения, либо вообще не будет выполнено.

3.5 Алгоритм распределения заданий по вычислительным узлам

Анализ модели машинного обучения с подкреплением позволяет перейти от работы агента с окружающей средой к составлению алгоритма, который реализует поведение модели в контексте распределения заданий по вычислительным узлам.

Работа модели машинного обучения с подкреплением может быть реализована в виде мультиагентной системы [95], где элементы РИСИВ представляются как множество взаимодействующих друг с другом сущностей — агентов. Такой подход предполагает множество дополнительных вычислительных расходов, связанных с передачей и хранением большого количества данных, а также наличия у каждого узла РИСИВ информации о полном состоянии модели в каждый момент времени. Поскольку РИСИВ реализуется на устройствах с ограниченными вычислительными возможностями и, возможно, с каналами связи низкого качества и низкой пропускной способности, то именно поэтому предлагается реализовать на распределяющем узле алгоритм распределения заданий, реализующий функциональность машинного обучения с подкреплением [102, 103].

Также стоит отметить, что на этапе эксплуатации распределяющий узел получает от вычислительного узла ответ, который также содержит значение вознаграждения вместе с результатом. Такой подход позволяет уменьшить количество передаваемых по сети данных за счет устранения дополнительного опроса вычислительных узлов.

Схема алгоритма назначения заданий показана на Рисунке 3.2.

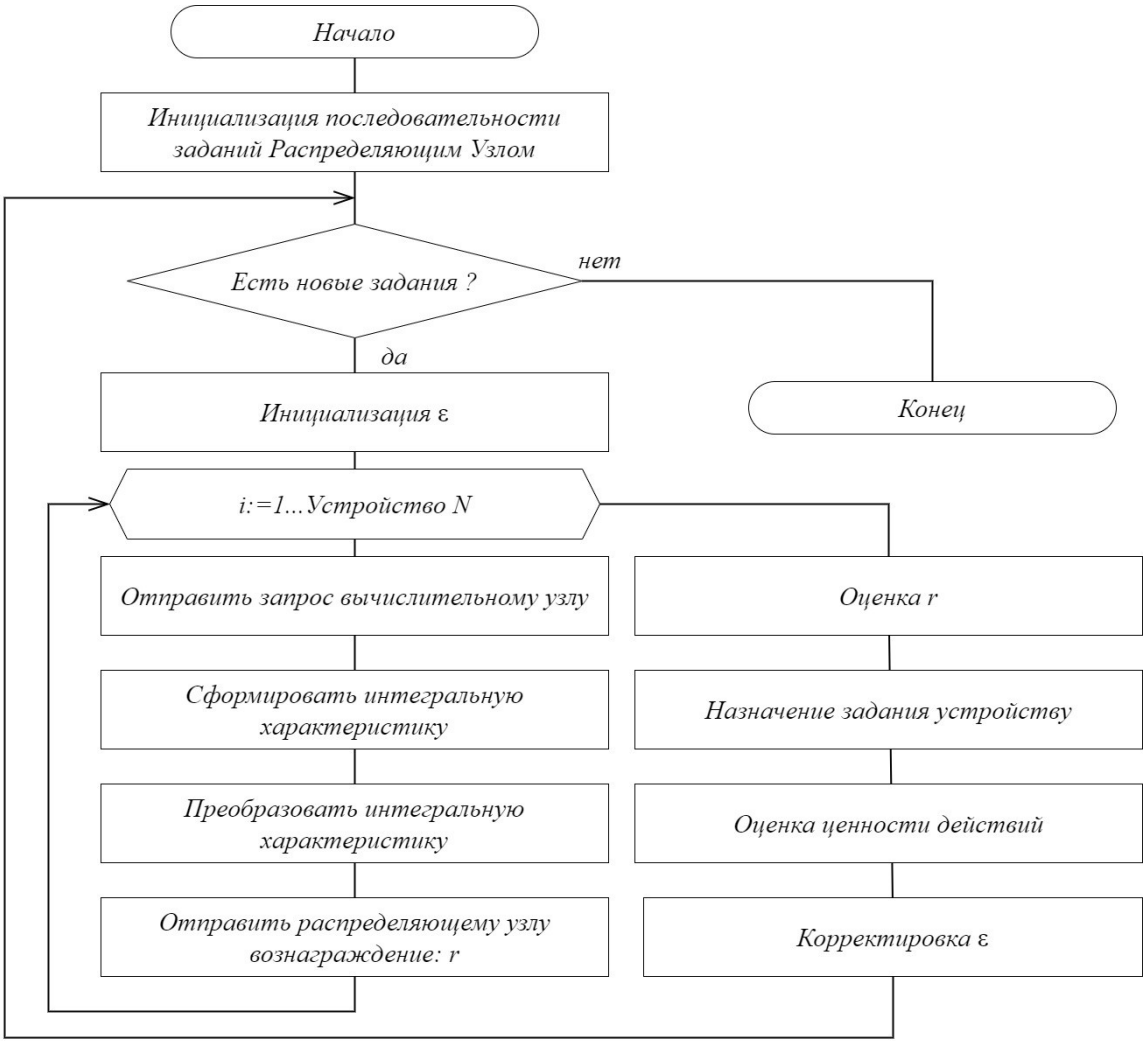

Рисунок 3.2. Схема алгоритма назначения заданий

Алгоритм назначения заданий[102, 103]:

Этап инициализации.

Шаг 0. Инициализация последовательности заданий распределяющим узлом в РИСИВ.

Этап исследования.

Шаг 1. Всем вычислительным узлам отправляется запрос от распределяющего узла на получение от них интегральных характеристик *Reward* , каждая из которых является значением сигнала вознаграждения каждого вычислительного узла и строится на основе его характеристик, в том числе, времени обработки пакетов, его доступности и т.д.

Шаг 2. Распределяющим узлом нормализуются и преобразуется полученные интегральные характеристики *Reward* в значения вероятностей. Данные значения вероятностей используются распределяющим узлом для выбора вычислительного узла для назначения и отправки задания.

Этап эксплуатации.

Шаг 3. Распределяющий узел отправляет задания на вычислительные узлы, при этом старается получить максимальное значение сигнала вознаграждения и, в общем, максимизировать общее вознаграждение на длительном промежутке времени.

Шаг 4. При получении выполненных заданий, распределяющий узел пересчитывает значения сигналов вознаграждения, поскольку с выполненными заданиями передается также информация о текущем состоянии РИСИВ и вычислительных узлов.

Шаг 5. Если в последовательности заданий нет невыполненных заданий, то выполняется переход на Шаг 6, иначе переход на Шаг 3 (для стационарного состояния) или переход на Шаг 1 (для нестационарного состояния).

Шаг 6. Конец алгоритма.

Общая схема взаимодействия, согласно алгоритму, имеет вид, как показано на Рисунке 3.3.

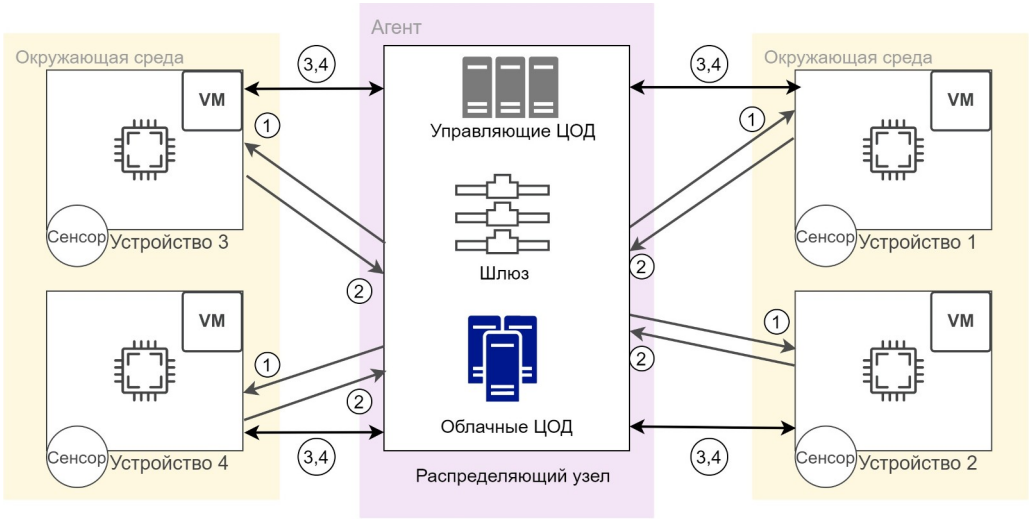

Рисунок 3.3. Общая схема взаимодействия алгоритма в РИСИВ

3.6 Модифицированный алгоритм распределения заданий по вычислительным узлам

В данном исследовании выполнялась проверка на применимость дальнейшего масштабирования разработанного алгоритма на основе машинного обучения с подкреплением, а именно: проверка способности назначения заданий вычислительных узлов друг другу с минимальным участием распределяющего узла [101].

Предлагается следующая последовательность:

- определить кластера устройств со схожими характеристиками;
- выполнить алгоритм назначения заданий.

Методы кластеризации можно разделить на два основных типа: жесткие и мягкие [104]. Первый тип методов имеет чёткие границы, в отличие от второго.

Среди существующих алгоритмов для выполнения кластеризации выделяется алгоритм нечёткой кластеризации С-средних (Fuzzy C-Means), который основан на данных о сходстве множеств и является алгоритмом мягкой кластеризации.

Принцип мягкой кластеризации

Принцип мягкой кластеризации предполагает, что рассматриваемый объект, вычислительный узел в РИСИВ, представленный устройством ИВ, может принадлежать одному или нескольким кластерам одновременно. Применение данного принципа к вычислительным узлам РИСИВ является оправданным, так как характеристики устройств ИВ изменчивы. И в зависимости от задаваемых условий, порогов, приоритетов рассмотрения характеристик, устройства ИВ могут быть соотнесены по нескольким кластерам одновременно. Кроме того, характеристики устройств ИВ могут не только варьироваться по значению, но и по наличию. Так, например, возможны ситуации, когда некоторые характеристики в принципе отсутствуют или их невозможно получить.

Следуя принципу мягкой кластеризации, одно устройство может попадать в несколько кластеров одновременно на основе своих меняющихся характеристик, что расширяет в принципе сам кластер и повышает вероятность назначения задания без постоянной необходимости пересчёта и перераспределения кластеров, что тем самым уменьшает вычислительную сложность и общую вычислительную нагрузку [105] в РИСИВ при назначении заданий вычислительных узлов друг другу.

Описанная модель устройства и разработанный способ получения интегральной характеристики - *Reward* позволяют учитывать изменчивость характеристик и выполнять назначение заданий.

Кластеризация вычислительных узлов, устройств ИВ, выполнялась с использованием Fuzzy C-Means (FCM) алгоритма и с перечнем условий:

- количество кластеров конечно и постоянно;

- центроид каждого кластера рассчитывается как [106]:

$$p_{jk}=\frac{1}{\sum_{i=1}^{c}\left(\frac{d_{jk}}{d_{ik}}\right)^{2/(m-1)}}, \tag{3.4}$$

$$q_k=\frac{\sum_{j=1}^{n} p_{jk}^{m} w_j}{\sum_{j=1}^{n} p_{jk}^{m}}, \tag{3.5}$$

где i - номер каждого w устройства ИВ, k - номер кластера, d_{jk} - расстояние до центроида от устройства ИВ по Евклидовой метрике , m - индекс нечеткости, который задаёт нечеткую область. Алгоритм позволяет определить степень принадлежности каждого устройства ИВ к каждому кластеру.

В данном случае в качестве агента выступает тот вычислительный узел, который имеет задание на отправку, и с помощью разработанного алгоритма на основе машинного обучения с подкреплением он ищет оптимальный узел для назначения задания. Все остальные вычислительные узлы по отношению к нему являются средой согласно модели машинного обучения с подкреплением. В

данном случае нет фокусировки на конкретные причины, по которым узел может нуждаться в поиске другого узла на назначение ему задания. Принимается, что узел в общем имеет такую потребность, и на период назначения задания сам становится распределяющим узлом, тогда в остальном принцип взаимодействия подобен описанному ранее.

Однако перед тем, как распределяющему узлу непосредственно назначить задание другому вычислительному узлу с помощью разработанного алгоритма, были выполнены дополнительные модификации к разработанному алгоритму, выполняющиеся перед ним.

По сути, при решении задачи назначения заданий с использованием FCM назначение может быть выполнено узлу и из другого кластера. То есть задание будет выполнено в любом случае, наиболее благоприятным считается выполнение задания узлов в рамках его наибольшей степени принадлежности к кластеру, однако если это невозможно или нет специальных предпочтений для назначения узлу, то оно может быть назначено на выполнение узлу с наименьшей степень принадлежности по отношению к рассматриваемому кластеру. То есть задание на выполнение будет назначено в любом случае.

Модифицированная версия разработанного алгоритма состоит из последовательности шагов [101]:

Этап инициализации.

Шаг 0. Инициализация последовательности заданий и начальных значений в РИСИВ.

Этап исследования.

Шаг 1. Распределяющий узел опрашивает все вычислительные узлы в РИСИВ, и получает от каждого соответствующий *Reward*, сформированный вычислительным узлом согласно модели вычислительного узла. Полученные *Reward* нормализуются и преобразуются распределяющим узлом.

Шаг 2. На основе вычисленного ранее *Reward* каждый вычислительный узел сопоставляется распределяющим узлом кластеру или нескольким кластерам с

использованием алгоритма Fuzzy C-Means одновременно, поскольку алгоритм относится к мягкому типу кластеризации. Распределяющий узел хранит запись разделения всех вычислительных узлов по кластерам. В случае изменений, производится пересчёт принадлежности к кластерам, и запись обновляется на распределяющем узле. Однако всем вычислительным узлам, находящимся в пределах кластера, сообщается данные о нахождении в этом кластере остальных узлов.

Шаг 3. Для получения данных о результате разбиения по кластерам каждый вычислительный узел отправляет запрос распределяющему узлу. В ответ он получает запись своего или своих кластеров и находящихся узлов. Если одна их характеристик узла меняется, осуществляется переход на Шаг 2.

Этап эксплуатации.

Шаг 4. При наличии вычислительного задания на назначение вычислительный узел перенимает на себя роль распределяющего узла. И производит выбор другого вычислительного узла в пределах кластера на назначение. В случае принадлежности узла, имеющего задание, нескольким кластерам одновременно узла, предпочтение на рассмотрение отдаётся узлам кластера, которые имеют наибольшую степень принадлежности этому кластеру. Таким образом, вычислительный узел на время становится в роли распределяющего узла, который определяет вычислительный узел на назначение задания. Дальнейшая последовательность шагов выполняется согласно разработанному методу и алгоритму назначения заданий от распределяющего узла к вычислительным узлам.

Шаг 5. Отправка значений вознаграждений вычислительными узлами распределяющему узлу внутри кластера.

Шаг 6. Расчет оптимального узла для назначения задания. В случае отсутствия возможности назначения задания происходит переход на Шаг 2.

Шаг 7. Назначение задания узлу.

Шаг 8. Конец алгоритма.

Схема модифицированного алгоритма приведена на Рисунке 3.4.

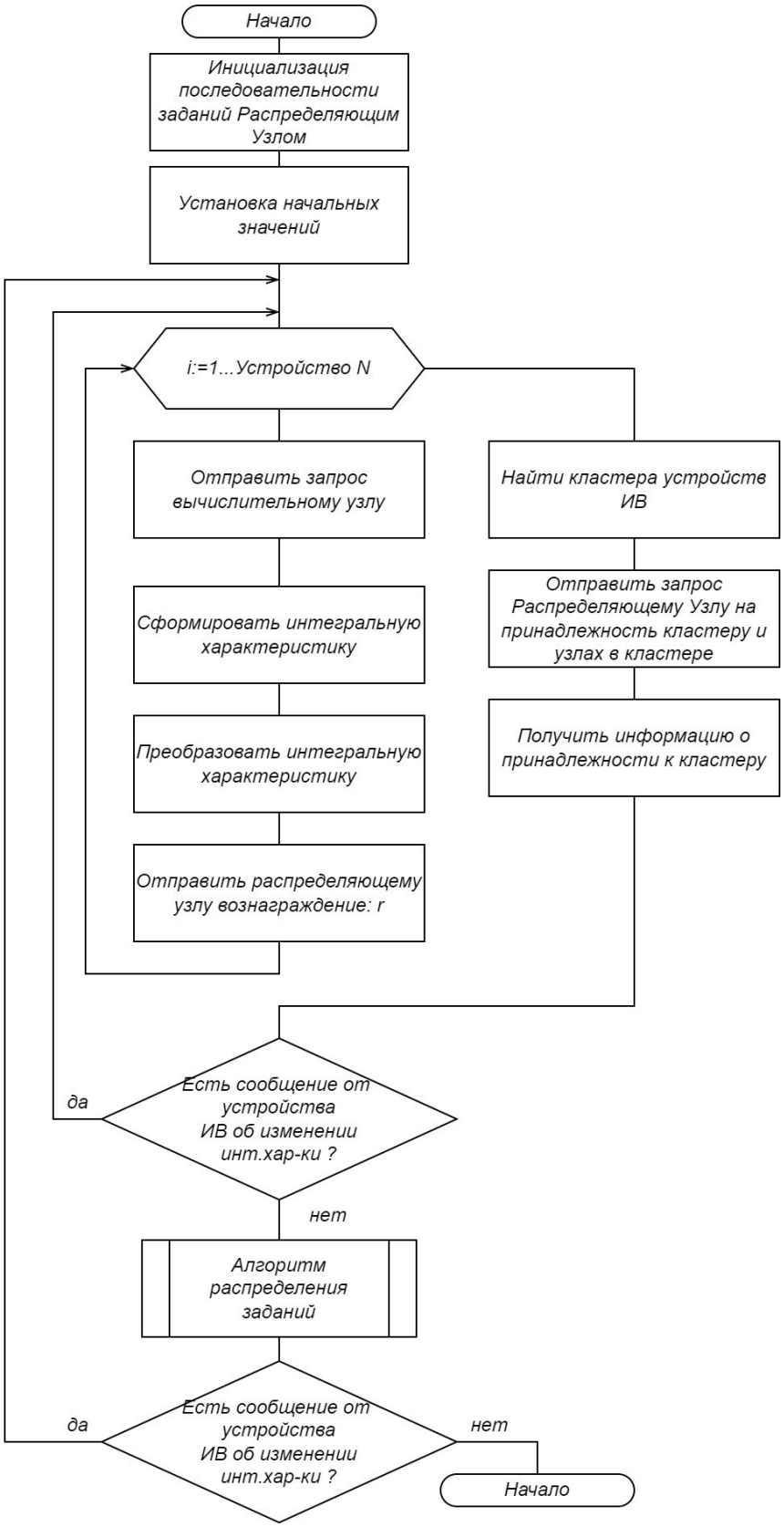

Рисунок 3.4. Схема модифицированного алгоритма назначения заданий

Архитектура выделения кластеров приведена на Рисунке 3.5.

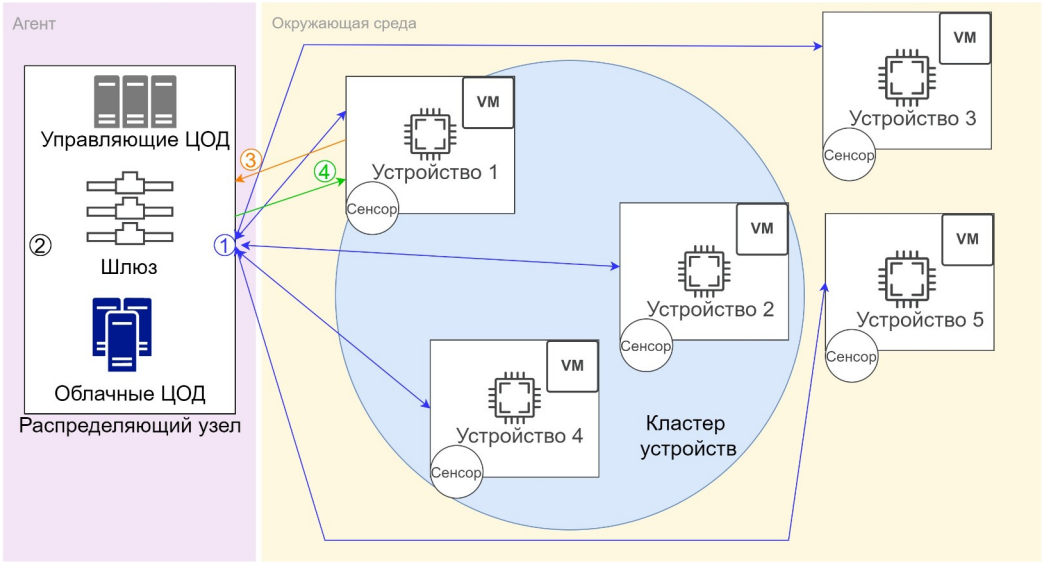

Рисунок 3.5. Архитектура выделения кластеров

Архитектура взаимодействия при модифицированном алгоритме приведена на Рисунке 3.6.

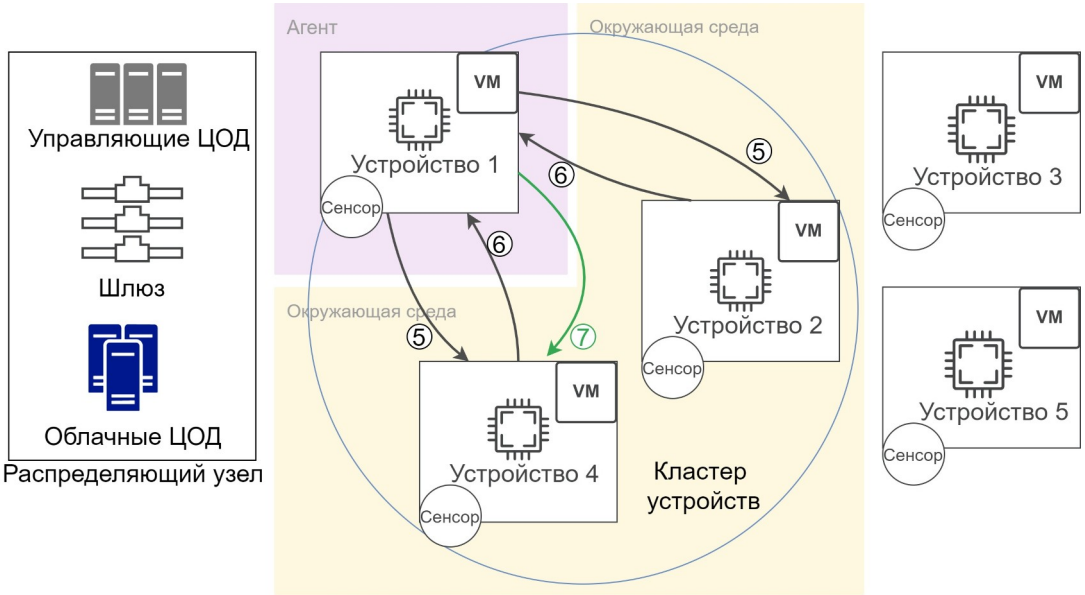

Рисунок 3.6. Архитектура взаимодействия при модифицированном алгоритме

3.7 Особенности работы алгоритма назначения заданий

Вне зависимости от произведённых модификаций над алгоритмом, особенности работы алгоритма назначения заданий остаются такими же, как и до произведённых модификаций. Поскольку эти модификации не затрагивают целевую модель назначения заданий, а лишь позволяют переопределить участников в процессе назначения заданий, а именно: расширяют возможности по тому, что может выполнять роль распределяющего узла в РИСИВ.

Таким образом, состояние вычислительных узлов РИСИВ может быть статическим, то есть неизменяемым со временем, и динамическим, то есть изменяемым со временем.

Преимуществом алгоритма распределения заданий на основе машинного обучения с подкреплением является его универсальность, так как один и тот же алгоритм может работать как со статической средой, так и со статической средой, так и с динамической (соответственно, работа в статическом и динамическом режимах).

Используя коэффициент ε можно изменять поведение алгоритма и лучше его настраивать на вычислительную среду, с которой работает распределяющий узел.

В том случае, если состояние вычислительных узлов не меняется, а параметры сети передачи данных также находятся в стабильном состоянии, то проводить дополнительные проверки на изменчивость среды нет никакого смысла, тогда значение параметра ε должно приближаться к 0, но не достигать его. Поскольку вероятность выбора каждого вычислительного узла не равна нулю, то не возникнет ситуации, что алгоритм распределяет задания только по узлам, которые имеют наибольшее значение сигнала вознаграждения (хотя в системе также присутствуют узлы с близким значением сигнала вознаграждения, но тем не менее отличным от максимального).

В случае существенной изменчивости вычислительной среды имеет смысл делать значение параметра ε как можно ближе к 1, что позволяет постоянно

«исследовать» новые вычислительные узлы, так как за небольшое время у них могли улучшится параметры (а следовательно, и вырастет значение вознаграждения).

Влияние параметра ε на процесс распределения заданий подробно рассматривается в экспериментальной части данной работы (глава 4). Однако необходимо отметить, что задание параметра ε близким к 1 в случае со статической средой не очень существенно влияет на процесс распределения заданий, в то время как задание параметра ε близким к 0 для динамической среды может иметь катастрофические последствия вычислительного процесса, вплоть до полной невозможности его осуществления.

Возможна ситуация, когда состояние сети передачи данных и вычислительных узлов неизвестно, то возможно подбирать параметр ε во время работы РИСИВ: в начале работы предположить, что система динамическая с существенными изменениями параметров, то ε будет близок 1, а затем по мере выполнения работы распределительный узел может накапливать статистику по изменениям сигнала вознаграждения каждого вычислительного узла и в соответствие с этим осуществлять уменьшение значения параметра ε до тех пор, пока производительность системы не достигнет максимального значения.

Стоит отметить, что, за счет решения дилеммы «исследование/эксплуатация» в обучении с подкреплением, при постоянном изменении параметра ε алгоритм все равно при $t \to \infty$ всегда будет давать близкое к оптимальному решение, даже если среда будет переходить из статического состояния в динамическое, и наоборот.

3.8 Выводы по третьей главе

Рассмотрение этапов выполнения вычислительной задачи позволило определить возможность использования методов из области классических параллельных и распределенных систем по преобразованию исходной задачи в виде последовательности заданий для ее решения на РИСИВ.

Во-первых, определен способ формирования заданий, определены параметры заданий, который зависят непосредственно от исходной задачи, а также параметры, которые необходимы для возможности выполнения заданий на вычислительных узлах.

Во-вторых, определена модель вычислительного узла, которая позволила задавать отдельный вычислительный узел, учитывать его особенности расположения и функционирования. Особое внимание уделено значению сигнала вознаграждения, которые вычислительные узлы будут отправлять распределяющему узлу для реализации алгоритма распределения заданий на основе машинного обучения с подкреплением. Поскольку сигнал вознаграждения является интегральной характеристикой вычислительного узла и может зависеть от множества параметров РИСИВ, то предлагается для расчета вознаграждения использовать время от отправки задания распределяющим узлом до получения его результатов от вычислительного узла (поскольку время также является величиной, зависящей от тех же самых параметров).

Рассмотренные особенности функционирования модели машинного обучения с подкреплением, реализованных метода и алгоритма демонстрируют поведение алгоритма при различных значениях параметра ε, который позволяет гибко настраивать вычислительную систему и позволяет решать дилемму «исследование/эксплуатация» для получения оптимальной производительности РИСИВ.

Глава 4. Экспериментальная оценка и результаты исследования эффективности метода назначения заданий на основе машинного обучения с подкреплением

В результате анализа модели обучения с подкреплением был получен алгоритм, который позволяет решать сложную вычислительную задачу за счет разделения на последовательность заданий, которые могут быть распределены на выполнение между вычислительными узлами РИСИВ.

Проведение экспериментальной оценки метода назначения заданий вычислительным узлам РИСИВ позволяет удостовериться в эффективности подхода, предложенного в данной работе. Изменение входных данных и параметров работы модели назначения заданий дает возможность выработки практических рекомендаций, которые позволят использовать метод в условиях решения реальных комплексных вычислительных задачах по обработке и анализу данных в РИСИВ.

Экспериментальные исследования разделены на две части: исследование поведения самой модели назначения заданий на основе обучения с подкреплением при различных параметрах, так и в реальной задаче формирования двухмерных изображений за счет распараллеливания алгоритма распространения лучей по вычислительным узлам, основанных на устройствах Интернета вещей.

Экспериментальная оценка метода позволяет выявить слабые стороны и недостатки предложенного в работе метода и дать рекомендации по их устранению.

4.1 Разработка программного обеспечения моделирования назначения заданий на основе машинного обучения с подкреплением

Для снижения сложности проектирования и производства устройств и объектов ИВ используются предложенные крупнейшими производителями

стандартизированные подходы и цифровые платформы, с применением микропроцессоров на базе ARM, Intel, ATmega. Такие решения позволяют использовать универсальные компоненты, такие как микропроцессоры, модули оперативной памяти, постоянные запоминающие устройства, периферийные устройства. С учетом большого количества производителей цифровых компонентов, возникает некоторый разброс в их параметрах, наличие архитектурных особенностей, что приводит к усложнению разработки программного обеспечения, а также снижает переносимость программного кода. Использования низкоуровневого языка ассемблера позволяет учитывать все особенности каждой платформы, но существенно ограничивает программистов в расширении разработки на другие платформы. Для повышения универсальности могут использоваться более высокоуровневые языки (например, язык Си, хотя его использование также может быть ограничено платформой). Использование операционной системы Linux позволяет снизить зависимость от оборудования и разработать универсальные инструменты работы с устройствами ИВ.

В 1995 году компанией Sun Microsystems был разработан язык программирования Java, который изначально был предназначен для создания единой универсальной платформы, которая могла бы функционировать в качестве встроенных систем для любой аппаратуры: микроволновки, домашние медиа-центры, телевизоры, компоненты «умного дома» и т. д. [107]. Платформа Java содержит множество программных средств, которые позволяют реализовывать приложения от встраиваемых до корпоративных. Универсальность Java дает возможность ликвидировать возможные отличия электронных компонентов и цифровых платформ разных производителей и архитектур. В основе платформы Java лежит виртуальная машина (JVM - Java Virtual Machine), которая устанавливается на все устройства информационной системы ИВ, что дает возможность реализовывать единый код, обладающий максимальной переносимостью, и который преобразуется в байт-код, выполняемый любой виртуальной машиной Java.

Таким образом, для реализации распределенной информационной системы на основе устройств Интернета вещей возможно использование языка Java, а для взаимодействия узлов между собой – технологии платформы Java [108].

Клиент-серверная архитектура

Взаимодействие компонентов распределенной информационной системы на базе устройств ИВ можно реализовать на основе архитектуры клиент-сервер. Возможно выделение двух видов взаимодействия при такой структуре [109]:

- распределительный узел отправляет задание (сообщение) на вычислительный узел;
- вычислительный узел отправляет результат (сообщение) на узел распределения задания.

Таким образом, выполняемые узлами роли клиента и сервера будут, соответственно, следующие:

- вычислительный узел является сервером, а узел распределения заданий является клиентом;
- узел распределения заданий является сервером, а вычислительный узел является клиентом.

Рассмотрим ситуация, когда вычислительный узел является сервером, а узел распределения заданий – клиентом (Рисунок 4.1). Будучи сервером вычислительный узел предоставляет клиенту вычислительные примитивы (service calculating primitive), в то время как узел распределения заданий запрашивает у клиента выполненные небольших вычислительных задач в соответствии с декомпозицией исходной вычислительной задачи. Также узел распределения заданий может выдавать команды на управление вычислительными узлами: например, переход в различные режимы работы процессора. В соответствии с полученными заданиями сервер реагирует: производит вычисления, переключает режимы и т. п.

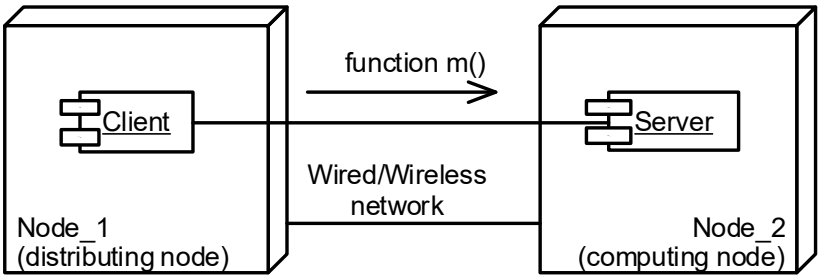

Рисунок 4.1. Взаимодействие вычислительного узла
и распределяющего узла

Рассматривая взаимодействие узлов в терминах клиент-серверной архитектуры, то приложение на Node_1 запрашивает у сервера Node_2 выполнение определенной на нем функции (вычислительного примитива), представленного в виде функции m(). С учетом того, что клиент и сервер размещаются на различных вычислительных узлах, взаимодействующих через компьютерную сеть, то возникает проблема. Таким образом, получается, что указанная функция расположена на одном узле (сервере), а вызывать ее должен другой узел (клиент), что невозможно без использования программных средств сетевого взаимодействия.

Платформа Java имеет три различные технологии, позволяющие удаленным приложениям взаимодействовать между собой через компьютерные сети [109], которые будут рассмотрены в далее разделе 4.2:

- сокеты (серверные сокеты и клиентские сокеты);
- удаленный вызов процедур RMI;
- технология CORBA.

4.2 Выбор технологии взаимодействия между узлами РИСИВ

Сокеты

Передача данных между распределяющим узлом (клиентом) и вычислительным узлом (сервером) может быть реализовано на основе сокетов, программных интерфейсов, обеспечивающих обмен данными между двумя приложениями (Рисунок 4.2).

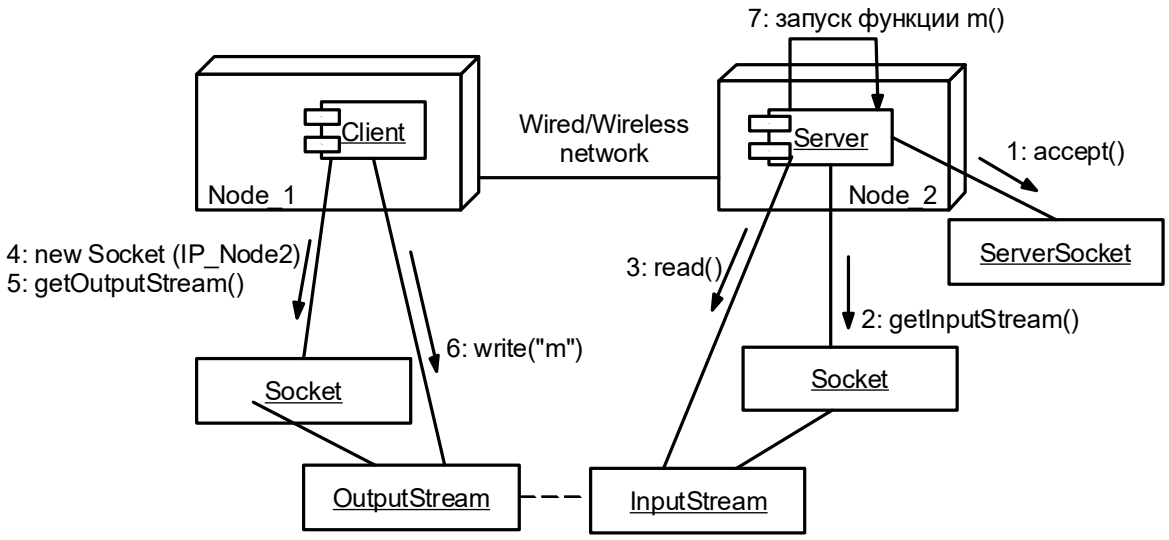

Рисунок 4.2. Взаимодействие на основе сокетов

Как указывалось ранее, клиентское приложение не имеет возможности вызвать функции сервера, поэтому у серверного приложения создается метод-заглушка (stub-method), выполняемого в случае поступления сообщения от клиента, а затем передавать управление функции, реализующий вычислительный примитив.

Листинг клиентской части приложения:

```
1         . . .
2         Socket socket = new Socket(“192.168.0.1”, 9001);
3         BufferedWriter out = new BufferedWriter(new
OutputStreamWriter(
   socket.getOutputStream()));
4         String message = “m”;
5         out.write(message, 0, message.length());
6         out.newLine();
7         out.flush();
8         socket.close();
9         out.close();
10        . . .
```

Работа алгоритма основывается на том, что клиент создает сокет и открывает его на соединение с сервером, а затем отправляет в него сообщение

(message = “m”), которое соответствует вычислительному примитиву и передаваемым ему параметрам на сервере, который необходимо выполнить.

Листинг серверной части приложения:

```
11        private void m() { . . . } // calculation primitive
12        . . .
13        ServerSocket serverSocket = new ServerSocket(9001);
14        Socket socket = serverSocket.accept();
15        BufferedReader in = new BufferedReader(new
InputStreamReader(
  socket.getInputStream()));
16        String message = in.readLine();
17        if(message.equals(“m”)) {
18            m()
19        } else {
20            System.out.println(“No such action”);
21        }
22        socket.close();
23        serverSocket.close();
24        . . .
```

Серверная часть приложения создает серверный сокет, который ожидает подключения по определенному порту и, в случае успешного подключения со стороны клиента, серверный сокет возвращает обычный сокет. С полученным сокетом связывается входной поток. И используя буферизированное чтение серверное приложение считывает присланное сообщение. Затем сервер анализирует сообщение и ставит ему в соответствие вычислительный примитив и передает ему параметры.

Для реализации обратного взаимодействия, когда сообщения отправляются от вычислительного узла к узлу распределения заданий, клиентская и серверная части меняются местами.

Удаленный вызов процедур

Удаленный вызов процедур в Java реализован в виде технологии RMI (удаленный вызов метода - Remote Method Invocation). Эта технология позволяет осуществлять взаимодействие между объектами, расположенных на разных виртуальных машинах. Схема взаимодействия приложений с использованием RMI показана на Рисунке 4.3.

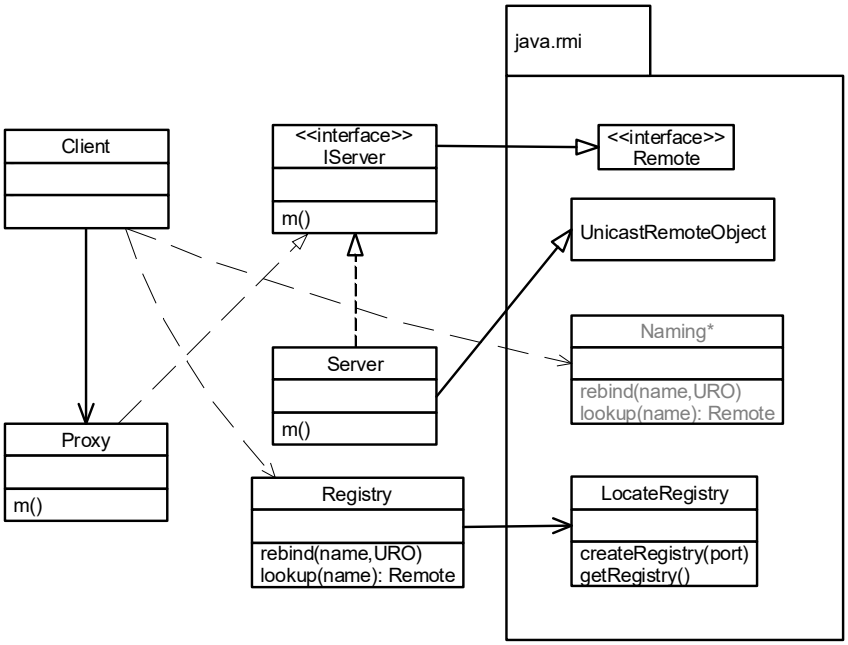

Рисунок 4.3. Взаимодействие приложений за счет удаленного вызова методов RMI

Для организации взаимодействия через RMI в РИСИВ необходимо создать интерфейс IServer, в котором будут описаны прототипы всех необходимых вычислительных примитивов, которые могут вызываться клиентом и которые будут реализованы на сервере.

Листинг интерфейса IServer:

```
25        public interface IServer extends Remote {
26          public void m() // calculation primitive
27             throws RemoteException;
28        }
```

Интерфейс IServer должен быть доступен и клиенту и серверу, так как на стороне сервера он будет реализован (implemented) в виде класса Server, а со стороны клиента он будет представлен в виде Proxy.

Листинг серверной части приложения:

```
29        . . .
30        public class Server extends UnicastRemoteObject
31          inmplenets IServer {
32          public Server () throws RemoteException {}
33          public void m() throws RemoteException {
34          . . . //calculation primitive function realization
35          }
36        public static void main() {
37        . . .
38          Server s = new Server();
39          Registry re = LocateRegistry.createRegistry(1099);
40          re.rebind(“server”, s);
41        . . .
42        }
```

Класс Server реализует всю функциональность серверной части вычислительной системы, все функции, соответствующие вычислительным примитивам. Экземпляр класса Server обеспечивает удаленный доступ. Служба именования Naming обеспечивает универсальное именование устройств ИВ и поиск удаленных объектов на шине RMI, для чего используется регистрация в реестре (Registry), куда передается имя, заданное разработчиком, и непосредственно сам экземпляр класса Server.

Листинг клиентской части приложения:

```
43        . . .
44        Registry re = LocateRegistry.getRegistry();
45        (IServer) s = (IServer) re.lookup(“server”);
46        s.m(); //remote function – calculating primitive
47        //or for Naming service
```

```
48        //IServer s = (IServer)
Naming.lookup(“rmi://localhost/server”)
49
```

В клиентском приложении создается объект, описанный интерфейсом IServer, и удаленный объект для доступа запрашивается в службе Naming по его имени, заданному на этапе регистрации на серверном приложении.

Технология CORBA

Технология CORBA (Common Object Request Broker Architecture) позволяет осуществить взаимодействие программного кода, написанного на разных языках с помощью платформонезависимого языка описания IDL (Interface Description Language, язык описания интерфейса). Унификация технологии CORBA осуществляется в соответствие со стандартом OMG (Object Management Group) и предназначена для реализации распределенных приложений. По аналогии с RMI технология CORBA также имеет «шину», на которой осуществляется регистрация и поиск удаленных объектов.

Остальные особенности технологии CORBA аналогичны RMI, но позволяет использовать несколько языков программирования при реализации распределенного приложения, что повышает сложность ее использования и развертывания.

Анализ подходов к организации взаимодействия между узлами

Каждая из описанных технологий имеет свои достоинства и недостатки, которые необходимо учитывать при разработке РИСИВ. К достоинствам сокетов можно отнести следующее:

- простота использования и реализации;

- за счет отсутствия привязки к платформам, сокеты позволяют взаимодействовать приложениям, написанных на разных языках программирования;

- технология сокетов реализована на практических всех платформах (в том числе в минимальную встраиваемую платформу для разработки мобильных приложений Java ME).

Недостатки технологии сокетов следующие:

- в приложении должны быть подпрограммы, обеспечивающие обработку входящих сообщений;

- отсутствие системы именования и регистрации объектов, что усложняет реализацию работы с множеством устройств ИВ;

- для серверного приложения необходимо настраивать несколько портов (входной и выходной) для обеспечения двустороннего взаимодействия, а также такой наор портов необходим для каждого удаленного устройства ИВ.

К достоинствам технологии RMI можно отнести следующее:

- простота реализации на платформе Java;

- программная шина для регистрации и поиска удаленных объектов (возможна регистрация самих серверных и клиентских приложений в качестве удаленных объектов);

- не требуется использование дополнительных языков описания удаленных объектов.

К отрицательным сторонам технологии RMI можно отнести следующее:

- технология является частью платформы Java, поэтому взаимодействие с приложениями, написанными на других языках и реализованных на других платформах, усложнено;

- не входит по умолчанию в платформу разработки встраиваемых приложений Java ME (требуется дополнительная установка библиотек).

Применительно к разработке РИСИВ технология CORBA имеет следующие достоинства:

- возможность разработки приложений с использованием разных языков программирования и программных платформ;

- технология является открытой и стандартизирована OMG;

- наличие программной шины для регистрации и поиска удаленных компонентов.

К недостаткам технологии CORBA можно отнести:

- сложность реализации и развертывания распределенных приложений;

- для каждой платформы должно быть разработано отдельное приложение для реализации программной шины;

- необходимость описания удаленных компонентов на отдельном языке IDL, не являющемся частью какого-то другого языка программирования.

Выбор технологии взаимодействия между узлами

Поскольку для реализации РИСИВ предполагается использовать платформу Java и соответствующий язык программирования, то в данном случае возможно использование всех технологий, обеспечивающих взаимодействие приложений, поскольку они имеются в стандартных библиотека Java. Использование других, более сложных технологий, таких как Spring или Enterprise Java Beans (EJB), для взаимодействия компонентов РИСИВ не имеет смысла ввиду того, что они созданы для разработки корпоративных приложений и ни в каком виде не могут быть использованы при работе на маломощных устройствах ИВ (например, сами библиотеки программных компонентов требуют сотни мегабайт на постоянной памяти).

Подход, основанный на использовании сокетов, требует разработки дополнительного программного кода: для обработки входящих сообщений, для обеспечения адресации и именования удаленных объектов. Таким образом, возникает ситуация, что все, что имеется по умолчанию в технологиях RMI или CORBA, разработчику придется реализовывать самостоятельно, это определяет отказ от использования сокетов при реализации РИСИВ.

В технологии CORBA имеется множество возможностей, что обеспечивает ее универсальность, но при этом также увеличивается и сложность разработки программного обеспечения для РИСИВ. Мультиплатформенность CORBA может

быть существенным преимуществом для реализации на гетерогенных вычислительных узлах, но так как предполагается наличие виртуальной машины Java на всех устройствах, то указанное достоинство теряет свой смысл.

RMI обладает всеми достоинствами CORBA, но является частью платформы Java, на которой предполагается реализация РИСИВ. В данном случае, технология RMI имеет более простую реализацию, чем конкурирующие технологии, а также она развивается вместе с платформой Java, становясь все более простой для разработчиков. Таким образом, универсальность языка Java позволяет разрабатывать приложения для широко класса устройств, а наличие виртуальной машины JVM для операционной системы GNU/Linux позволяет реализовывать программный код для элементов РИСИВ, в основе которого будет лежать технология RMI.

В результате, реализация распределенных приложений для обеспечения функционирования частей РИСИВ основана на использовании «нативной» технологии платформы Java – RMI.

4.3 Разработка методики и программы экспериментальных исследований

Алгоритм распределения заданий, основанный на обучении с подкреплением, невозможно сравнить с другими аналогичными алгоритмами в классических параллельных и распределенных системах. В параллельных системах не предполагается, что вычислительные элементы могут менять свои характеристики, а в распределенных системах узлы также не могут менять свои характеристики и также не могут во время работы выходить из состава системы и снова подключаться.

На сегодняшний день существует целый ряд алгоритмов распределения заданий по узлам распределенной вычислительной системы [110, 111]. Существенная часть алгоритмов предполагает, что структура вычислительной системы не будет меняться во время работы, поэтому возможны методы статического оптимального (субоптимального) распределения заданий, где

возможно использовать в том числе эвристические подходы [112, 113]. Другой тип алгоритмов предполагает наличие гомогенной вычислительной среды (например, [114]). Наиболее близкие к предложенному в данной работе методы — на основе адаптивного подхода [115] – разработаны только для реализации MapReduce, и в методе также не предполагается постоянное изменение структуры вычислительной среды. Тем не менее, поведение метода назначения заданий вычислительным узлам на основе машинного обучения с подкреплением для РИСИВ можно сравнить с результатами, приведенными в [110], причем именно с результатами планирования в режиме реального времени (online scheduling).

Исследование и эксперименты в данной работе состоят из четырех частей:

1) исследование поведения модели назначения заданий:

- исследование поведения алгоритма при стационарном состоянии;

- исследование поведения алгоритма при нестационарном состоянии;

2) исследование поведения алгоритма при его модификации и модифицированном методе в стационарном и нестационарном состояниях;

3) тестирование работы алгоритма при отправке тестового задания трассировки луча.

Одинаковым для всех исследований является:

- для передачи заданий и получения значений вознаграждений используется технология Java RMI;

- помимо интегральных характеристик вычислительных узлов РИСИВ иных характеристик и параметров не вводится для расчёта значений вознаграждений вычислительных узлов в РИСИВ;

- распределяющий узел принимает решение о назначении задания на основе значений вознаграждений, значения вознаграждения в РИСИВ формируют только вычислительные узлы;

- распределяющему узлу в РИСИВ в качестве действий доступны только вычислительные узлы в РИСИВ, дополнительные действия в РИСИВ не вводятся;

- значение вознаграждения изменяется в диапазоне [0; 1].

Показателями оценки работы алгоритма для всех исследований является, то есть того, что алгоритм большую часть времени выбирал оптимальное действие (вычислительный узел) основаны на его среднем поведении, которыми являются:

- средние значения производительности (mean rate);

- функции вознаграждения (mean reward);

- функция сожаления (mean regret), обратная функция к функции вознаграждения.

Данные показатели описывают в принципе оптимальность выбранной стратегии агентом.

Для третьего исследования показатели дополняются: временем выполнения задания в зависимости от числа узлов и размера обкатываемого изображения.

В качестве решаемой задачи выбирается задача, которая имеет высокий уровень распараллеливания, при этом необходимо учесть, что задания (вычислительные примитивы) должны быть одинаковыми по вычислительной сложности и количеству передаваемых данных. Также на вычислительных узлах должна быть одинаковая реализация вычислительного модуля, с целю уменьшения влияния характеристик аппаратной платформы на итоговую оценку работы алгоритма.

В результате проведения экспериментов необходимо обнаружить возможные особенности работы метода назначения заданий вычислительным узлам РИСИВ на основе машинного обучения с подкреплением и предложить рекомендации для работы с РИСИВ.

4.4 Исследование поведения модели назначения заданий

Исследование поведения алгоритма при стационарном состоянии

В данном исследовании принимается:

- распределяющий узел в РИСИВ один;

- в процессе работы алгоритма на всех итерациях распределяющий узел не меняется;

- количество вычислительных узлов в РИСИВ постоянно и не меняется со временем;

- интегральные характеристики вычислительных узлов в РИСИВ постоянны в своем значении, то есть, значения вознаграждений постоянны;

- параметр ε для каждого эксперимента принимает единственное значение из $\varepsilon = [0; 0{,}001; 0{,}05; 0{,}25; 0{,}1; 0{,}2; 0{,}3; 0{,}4; 0{,}5; 1]$ и сохраняет это значение на всём протяжении работы одной итерации алгоритма; под итерацией принимается назначение всех заданий одной задачи вычислительным узлам РИСИВ.

В статическом режиме структура и поведение вычислительных узлов не меняется, поэтому во время взаимодействия распределяющего узла с вычислительными узлами интегрированная характеристика возвращают постоянное значение.

Для оценки процедуры назначения заданий в РИСИВ используется вариант алгоритма многорукого бандита, у которого параметры вероятностей выбора действия остаются неизменными. Параметр ε определяет в РИСИВ баланс между режимами исследования и эксплуатации. В данном случае не важно, какая задача будет решаться и какие параметры у вычислительных узлов. Основное условие, что в каждом эксперименте эти параметры были одинаковыми.

На диаграммах (Рисунок 4.4 и Рисунок 4.5) показаны значения вознаграждения, получаемого агентом, при выборе 5 и 20 действий (соответственно) при выбранных значения параметра ε: переключения модели назначения заданий от стадии исследования к стадии использования и выбору действия с максимальной выгодой.

Результаты, показанные на Рисунке 4.4 и Рисунке 4.5, демонстрируют, что значение среднего вознаграждения со временем достигает некоторого стабильного значения, то есть процедура назначения будет в итоге давать стабильное качество распределения заданий.

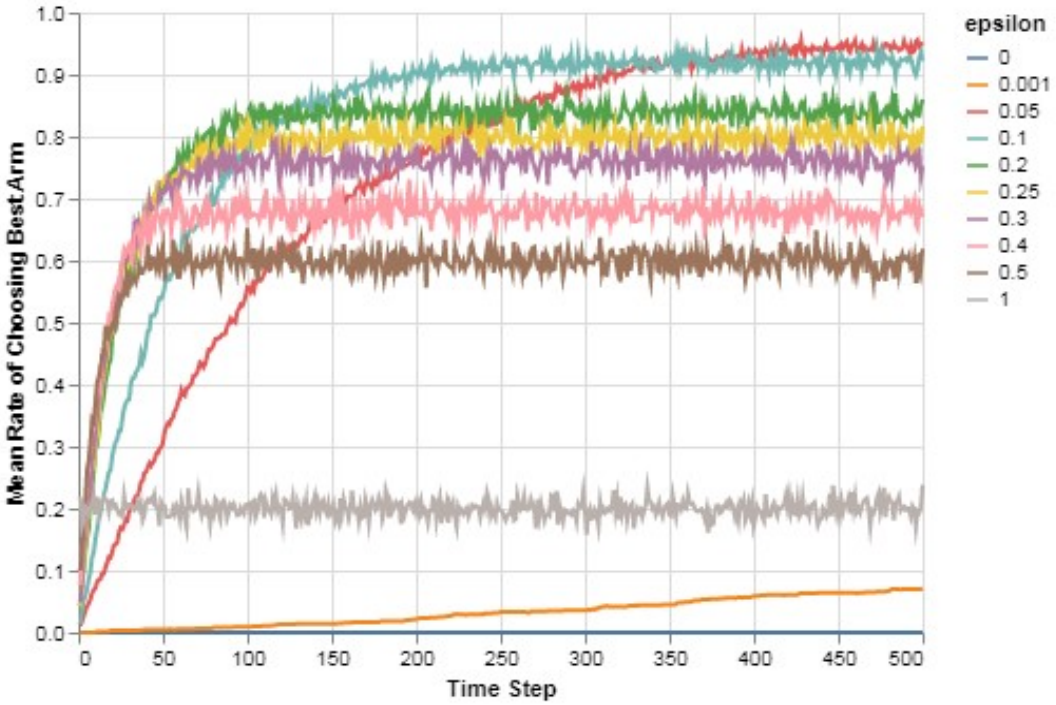

Рисунок 4.4. Средняя скорость достижения оптимальной стратегии выбора наилучшего действия (из 5 действий) при 1000 симуляциях

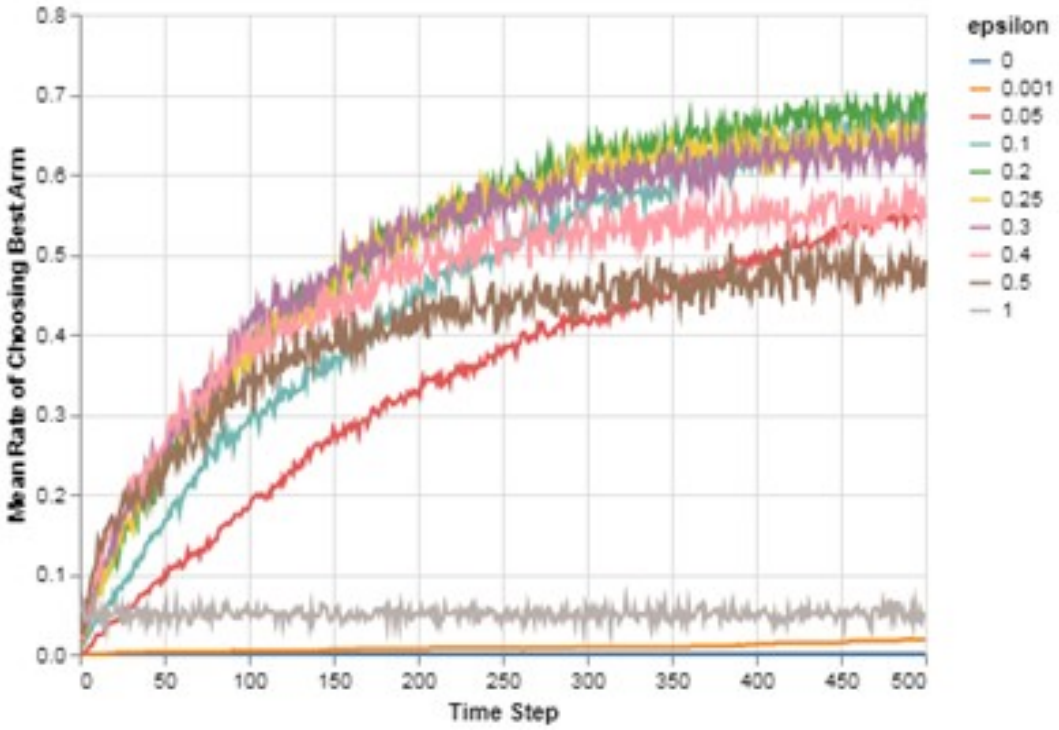

Рисунок 4.5. Средняя скорость достижения оптимальной стратегии выбора наилучшего действия (из 20 действий) при 1000 симуляциях

Увеличение параметра ε приводит к тому, что алгоритм делает больше попыток исследования, и меньше попыток эксплуатации. То есть, агент старается исследовать окружающею среду больше, чем получить от нее полезный результат: распределяющий узел постоянно осуществляет отправку заданий на новые узлы, вместо того чтобы использовать вычислительные узлы, характеристики которых ему известны. Такое поведение соответствует ранее проведенным исследованиям, например, в [102, 97].

Таким образом, увеличение параметра ε повышает вероятность случайного выбора вычислительного узла, вместо выбора узла с наилучшими показателями.

Диаграмма зависимости суммарного значения вознаграждения показано на Рисунке 4.6 и Рисунке 4.7.

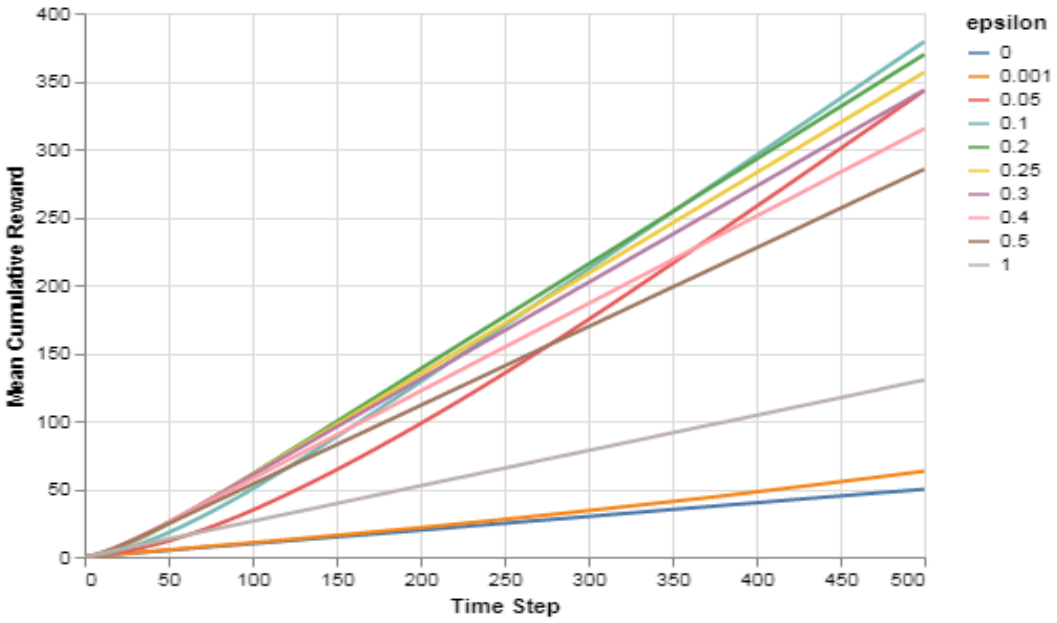

Рисунок 4.6. Среднее суммарное значение вознаграждения (5 действий) при 1000 симуляциях

Можно отметить, что с увеличением количества попыток происходит увеличение общего суммарного значения вознаграждения, как показано на диаграммах на Рисунке 4.6 и Рисунке 4.7 для 5 и 20 действий (соответственно). Изменение суммарного значения вознаграждения позволяет определить общее поведение системы на длительном промежутке времени, в то время как на

начальных этапах у системы больше ошибок, то в дальнейшем во время работы значение вознаграждения увеличивается.

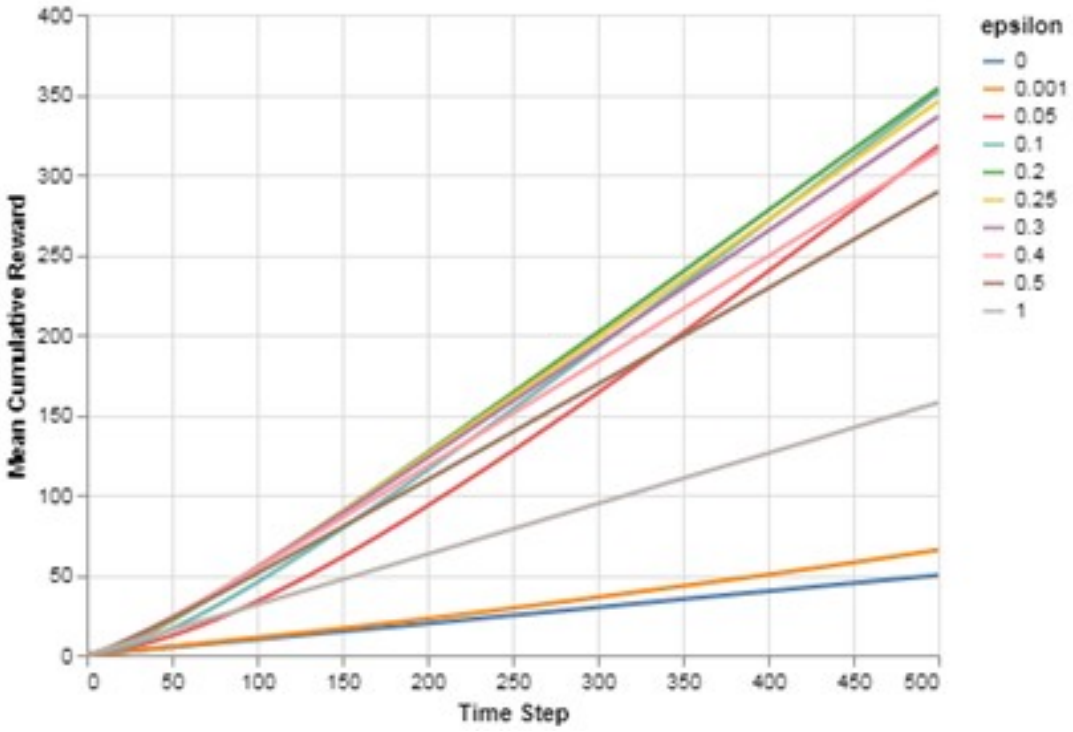

Рисунок 4.7. Среднее суммарное значение вознаграждения (20 действий) при 1000 симуляциях

Для оценки оптимальной стратегии алгоритма вводится функция сожаления (regret function), согласно [98, 103] имеющая логарифмический вид. Рисунок 4.8 показывает зависимость среднего значения функции сожаления от параметра ε. Функция сожаления является монотонно возрастающей при любом значении параметра ε, но для $\varepsilon \in [0{,}2\,;0{,}5]$ скорость роста функции наименьшая.

С ростом числа вычислительных узлов (действий), включаемых в РИСИВ, увеличивается и время, при котором алгоритму требуется перейти в режим эксплуатации из режима исследования. Данное утверждение доказывается проведенными исследованиями. В данном случае рассматривалось число действий (узлов) – 20. Эффективность ε также сохраняется прежней с увеличением числа узлов.

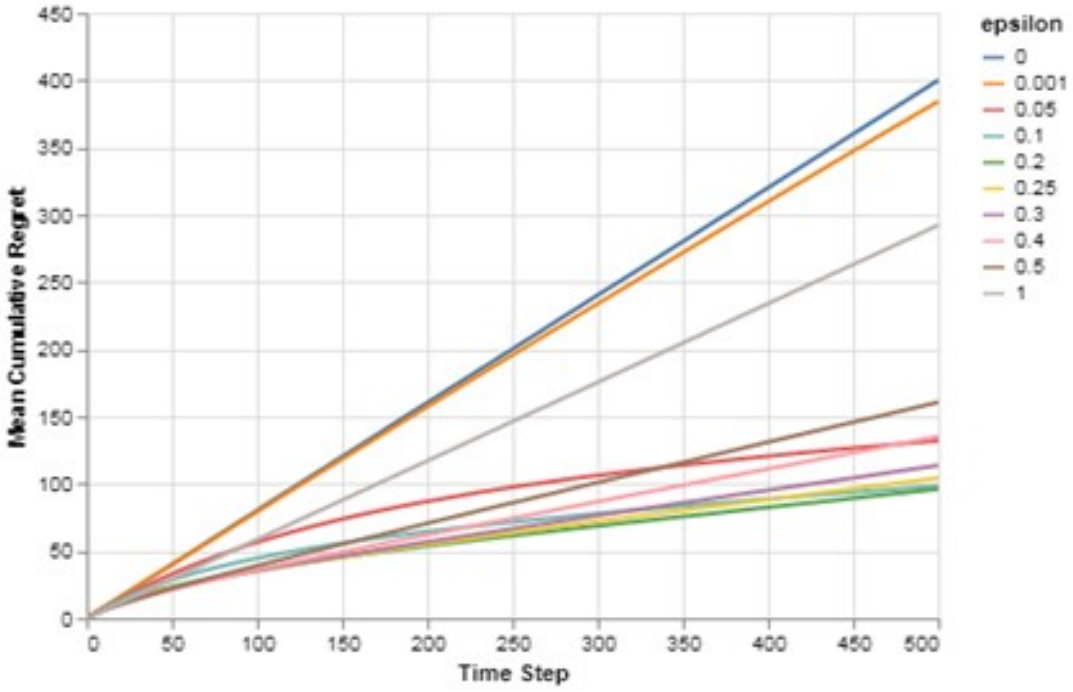

Рисунок 4.8. Среднее значение функции сожаления (20 действий) при 1000 симуляций

Исследование поведения алгоритма при нестационарном состоянии

В данном исследовании принимается:

- распределяющий узел в РИСИВ один;

- распределяющему узлу в РИСИВ в качестве действий доступны только вычислительные узлы в РИСИВ, дополнительные действия в РИСИВ не вводятся;

- в процессе работы алгоритма на всех итерациях распределяющий узел не меняется;

- количество вычислительных узлов в РИСИВ постоянно и не меняется со временем;

- интегральные характеристики (значения сигнала вознаграждения) вычислительных узлов в РИСИВ меняются со временем;

- параметр ε в процессе работы алгоритма стремится к полностью к жадному поведению, то есть из режима исследования $\varepsilon = 1$ стремится к режиму эксплуатации $\varepsilon = 0$.

В реализованном алгоритме с динамически изменяющейся вероятностной характеристикой и значениями вознаграждений, которое называется поведением в нестационарном режиме, или в динамическом режиме, функция сожаления имеет вид, показанный на Рисунке 4.9.

Функция среднего суммарного вознаграждения агента в нестационарном режиме показано на Рисунке 4.10.

Как видно из рисунков 4.9 и 4.10, в полученном динамическом случае оптимальным значением также является 0.1, однако в данном случае в алгоритм сам оценивает оптимальное решение и обеспечивает сходимость до него. Также поскольку алгоритм реализует динамическую/адаптивную концепцию, то функция сожалений намного меньше [102].

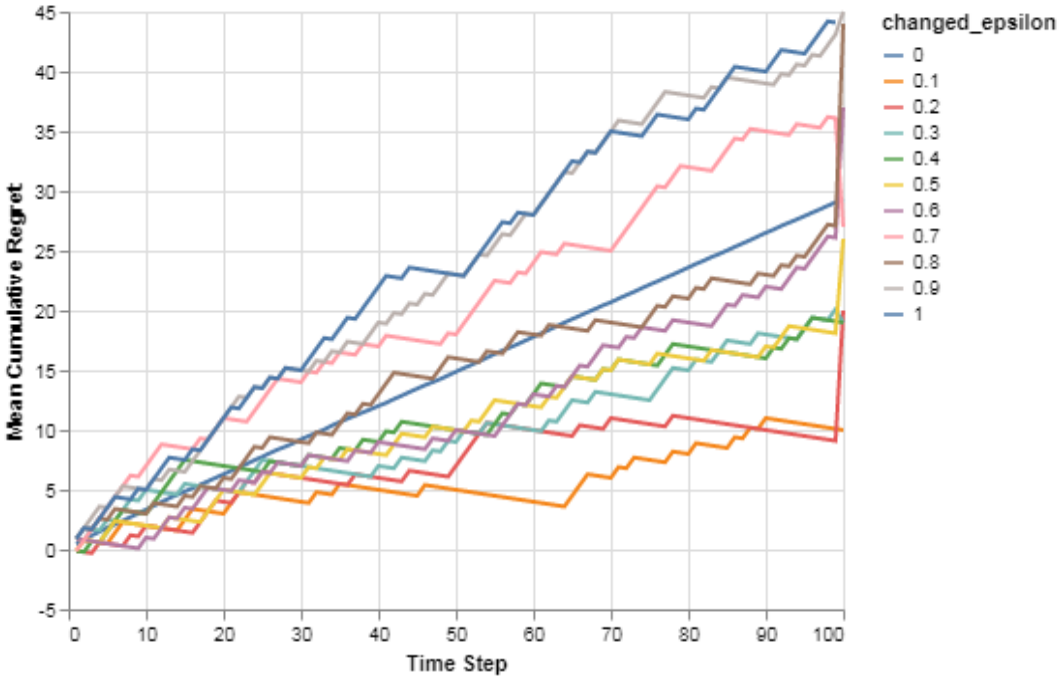

Рисунок 4.9. Функция сожаления в динамическом режиме (5 действий, 100 симуляций)

Наиболее информативным и показательным свидетельством работоспособности алгоритма в нестационарном состоянии является оценка посредством среднего суммарного сожаления и среднего суммарного вознаграждения, поскольку, находясь в нестационарном состоянии, средняя скорость достижения оптимальной стратегии может не сохраняться долгое время

на одном и том же уровне вследствие происходящих изменений. Средние суммарные значения отражают способность РИСИВ в нестационарном состоянии назначать задания с приемлемым уровнем значений.

Поведение алгоритма при его модификации и модифицированном методе в стационарном и нестационарном состояниях

В данном исследовании принимается:

- распределяющий узел для кластеров один, однако внутри каждого кластера выделяются собственный распределяющий узел, который меняется со временем; распределяющим узлом внутри кластера становится тот, который имеет задание на распределение;

- количество кластеров постоянно;

- количество вычислительных узлов в кластерах изменчиво;

- количество вычислительных узлов в РИСИВ изменчиво;

- параметр $\varepsilon = 0{,}1$.

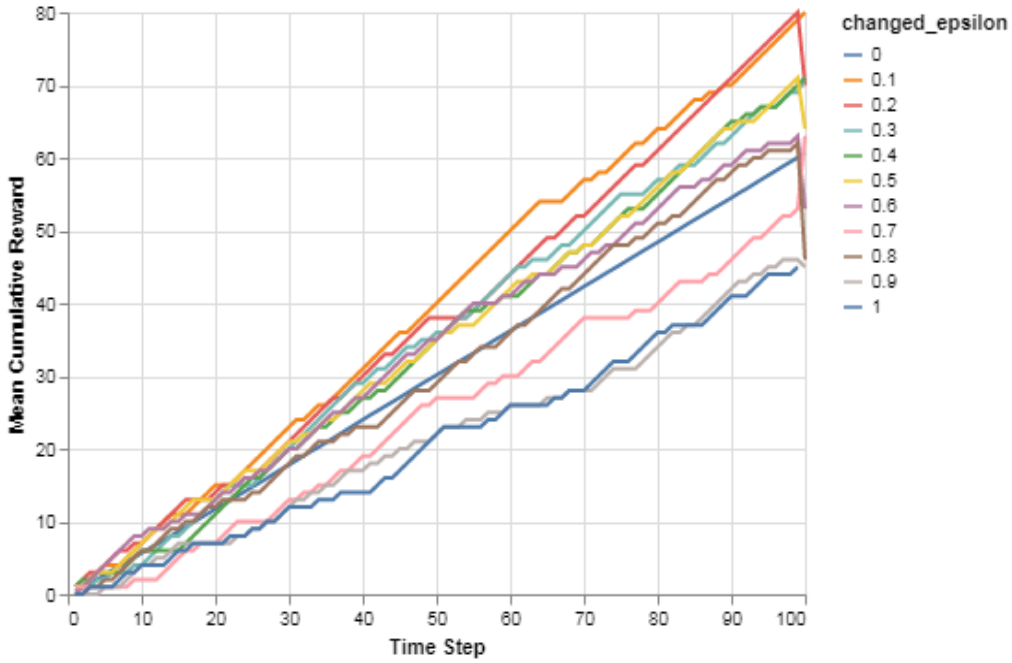

Рисунок 4.10. Среднее суммарное вознаграждения в нестационарном режиме (5 действий, 100 симуляций)

Индекс нечёткости в модифицированном алгоритме для FCM принимается $m=2$, данное значения индекса в большинстве случаев является оптимальным [116]. Тем не менее, в каждом случае выделения кластеров следует учитывать потенциально возможные размеры кластеров, и в зависимости от этого выбирать и устанавливать определённое значение индекса [117]. В данном случае выбрано значение индекса нечеткости $m=4$.В данной работе значение количества кластеров фиксировано и равно 5. На Рисунке 4.11 в левой части указаны устройства ИВ до процесса кластеризации, в правой части после кластеризации.

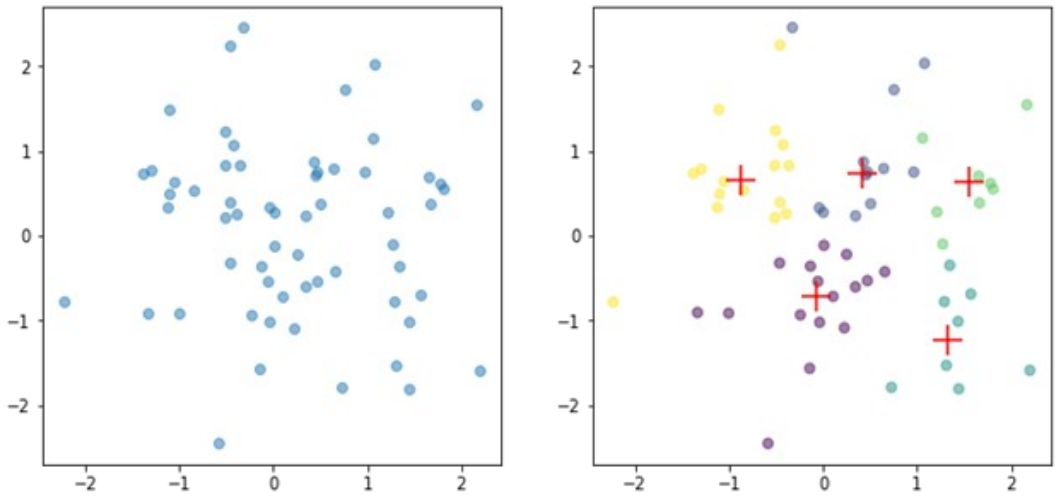

Рисунок 4.11. Кластеры устройств ИВ

Рисунок 4.11 (слева) демонстрирует данные распределения точек параметров устройств вычислительных узлов: каждое устройство ИВ имеет свои параметры, которые соответствуют модели устройства. Каждая точка данных, описывающих параметры каждого устройства, сопоставляется с каждым центроидом кластера с учетом близости по евклидову расстоянию. После добавления точки в кластер значения центроидов пересчитываются. Рисунок 4.11 (справа) показывает группы точек, сгруппированных вокруг центроидов и имеющих близкие значения параметров.

После того, как были выявлены кластеры, вычислительные узлы внутри своих кластеров перед непосредственной отправкой задания проходят этап назначения с помощью разработанного алгоритма на основе машинного обучения

с подкреплением, имеющего два характерных для обучения с подкреплением этапа:

- исследования: распределительные узлы кластера формируют запросы другим вычислительным узлам кластера. В случае активности устройств в кластере, то ими осуществляется подготовка и отправка ответного сообщения (сигнала вознаграждения);

- эксплуатации: распределительный узел кластера, сравнивая полученные вознаграждения, осуществляет выбор в кластере вычислительного узла для отправки задания на исполнение.

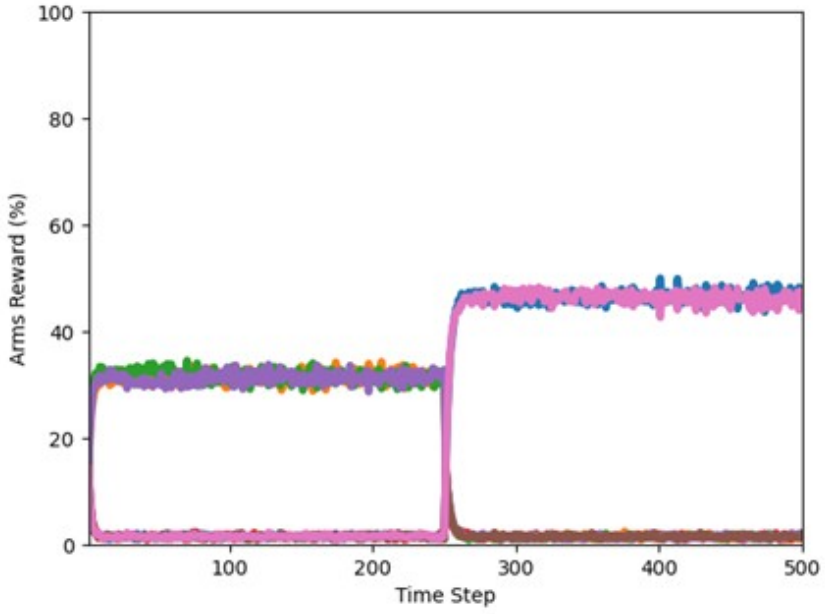

Рисунок 4.12. Выбор оптимального устройства (от 5 до 7 устройств ИВ, 2000 симуляций)

В левой части Рисунка 4.12 до временного порога в 250 число устройств в кластере составляло пять штук, после были включены в кластер два новых устройства, которые исходя из своих характеристик генерировали интегральную характеристику и выдали её в виде сигнала вознаграждений. При этом значения вознаграждений остальных узлов в кластере остались без изменений.

Рисунок 4.12 демонстрирует зависимость изменения значений вознаграждений внутри одного кластера от количества вычислительных узлов в

нем. Значение вознаграждение сохраняется за устройством до тех пор, пока не будет назначено новое задание.

В течении некоторого количества временных шагов распределяющее устройство сможет осуществить опрос всех вычислительных узлов в кластере и затем обновит значения сигналов вознаграждений. Распределяющий узел оценивает значения сигналов полученных вознаграждений и передаст задание на выполнение тому вычислительному узлу, который сможет его выполнить наиболее эффективно.

Рисунок 4.13 демонстрирует среднее значение сигнала вознаграждения для пяти и семи устройств ИВ (соответственно, левая и правая часть рисунка). Значение вознаграждения растет быстрее и быстрее достигает максимума для меньшего количества устройств, чем для большего. Кроме того, среднее значение вознаграждения будет больше в том случае, если сами значения сигналов вознаграждений будет больше, что определено особенностями выполнения алгоритма [118].

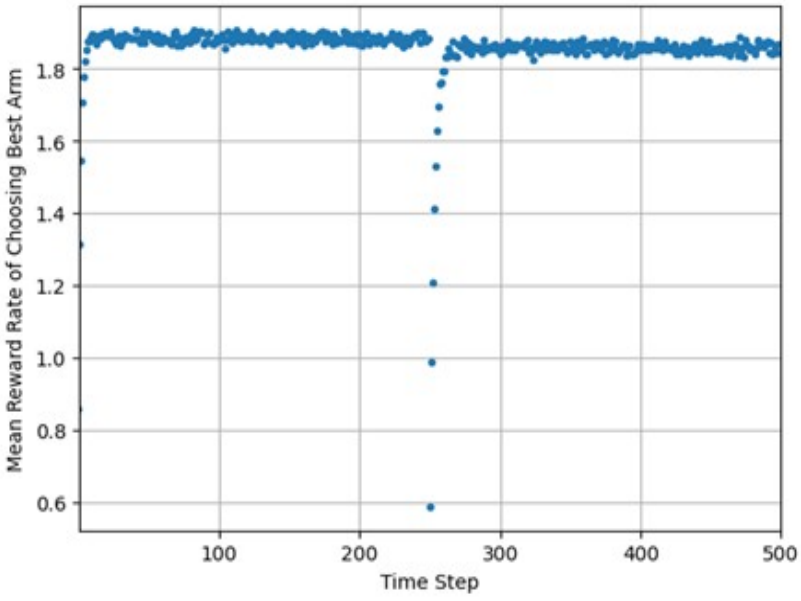

Рисунок 4.13. Среднее значение вознаграждения при выборе лучшего устройства ИВ (5 и 7 устройств, 2000 симуляций)

Указанный модифицированный алгоритм показывает стабильность и применимость разработанного алгоритма для дальнейшего его масштабирования. Например, для сокращения числа транзакций между устройством и главным узлом в принципе для систем, построенных на ИВ и обеспечения автономности устройств внутри систем на ИВ [101].

4.5 Реализация алгоритма трассировки луча на РИСИВ

Тестирование работы алгоритма при отправке тестового задания трассировки луча

Использование метода назначения заданий вычислительным узлам, основанном на машинном обучении с подкреплением, дает возможность подойти к проблеме выполнения распределенных вычислений с использованием РИСИВ [119]. Таким образом, становится возможным:

- использование распределенной информационной системы на основе устройств ИВ, имеющей гетерогенные узлы с изменяющимися параметрами, для выполнения распределенных вычислений;
- уменьшение затрат на модификацию математического и программного обеспечения, которое было ранее разработано для выполнения на параллельных или распределенных системах, и использовать его для вычислений на вычислительных узлах на основе устройств ИВ;
- осуществить повышение эффективности вычислительных систем с помощью включения в их состав вычислительных узлов, реализованных на основе встраиваемых устройств ИВ.

Эксперимент с вычислительной задачей осуществляется на основе задачи трассировки луча, имеющую высокую степень распараллеливания, для расчетов которой может использоваться распределенная информационная система с вычислительными узлами на основе устройств ИВ.

Вычислительная задача трассировки лучей

Применение трехмерной модели для решения задачи по формированию изображения, то есть визуализации, является актуальной задачей в таких областях, как компьютерная анимация, компьютерные игры, моделирование в системах автоматизации проектных работ, в компьютерной графике и других. Имеется множество алгоритмов визуализации, которые возможно классифицировать по следующим основным группам в зависимости от их подходов к получению результирующего изображения:

- проецирование (алгоритм растеризации и т.д.). В данной группе случае объекты сцены проецируются на экран наблюдения.

- трассировка (трассировка пути, прямая и обратная трассировка лучей, и т.д.). В данной группе вне зависимости от методов проведения лучей между сценой и экраном получение результирующего изображения всегда основывается на физических моделях сцены и луча, что означает требование наличия существенных вычислительных ресурсов для выполнения задачи.

Метод трассировки лучей (Ray tracing) основывается на построении траекторий лучей между экраном и поверхностями моделируемых объектов с учетом их взаимодействия [120]. Для проведения расчетов используются математические модели луча и, собственно, трехмерные модели объектов, изображение которых необходимо получить.

Элементами схемы трассировки луча являются: сцена, камера, окно просмотра.

Сцена формируется набором моделей объектов, изображения которых необходимо получить. Камерой (или наблюдателем) является точка сбора освещения, отраженного от объектов сцены. Наблюдателю доступна только часть сцены, которая видна в окне просмотра (viewport).

Трассировка луча имеет несколько подходов:

- прямая. В данном случае рассматривается луч, который распространяется от объекта сцены до камеры.

- обратная. В данном случае рассматривается луч, который распространяется от камеры к объекту сцены.

При прямой трассировке луча при построении изображений учитывается взаимное влияние лучей отражений объектов друг на друга, что даёт возможность более глубокой проработки изображений. Однако вместе с этим на реализацию подхода требуется больше вычислений, и как следствие, большие вычислительные затраты.

Подход обратной трассировки луча предполагает отслеживание только тех лучей, которые попадают в камеру, что требует меньше вычислений. Уменьшение вычислений позволяет получать различную степень «реалистичности» для получаемого изображения за счет проведения дополнительных вычислений учитывающих степень влияния лучей друг на друга.

На рисунке 4.14 приведена схема трассировки лучей, которая показывает условные обозначения элементов, используемых в алгоритме. Для алгоритма были приняты следующие условности:

- местоположение камеры определяет начало системы координат – $O=(O_x, O_y, O_z)=(0, 0, 0)$;

- направление обзора определяется осью Oz;

- окно просмотра ортогонально направлению обзора камеры;

- окно просмотра расположено на расстоянии d от начала координат;

- характеристики окна просмотра V_h и V_w (высота и ширина, соответственно), стороны окна просмотра параллельны плоскости xOy.

Расстояние до камеры и размер окна просмотра задают область видимости (field of view), то есть те элементы сцены, которые будут видны камерой и эти элементы попадут на конечное двумерное изображение. Канва (сетка изображения) ставится в соответствие окну просмотра. Канва определяет разрешение итогового изображения, которое формируется пикселями изображения (pixel - picture element). Каждый пиксель изображения соотносится с окном просмотра с помощью ячеек сетки.

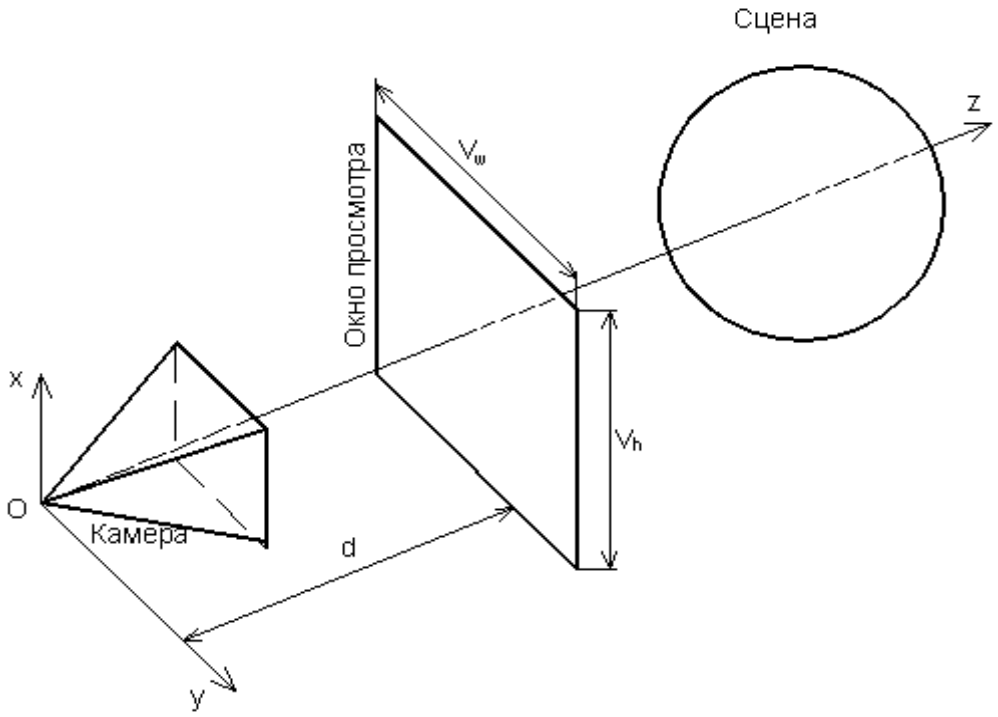

Рисунок 4.14. Схема трассировки лучей

Алгоритм формирования двухмерного изображения сцены можно сформулировать следующим образом:

Шаг 0. Начало алгоритма.

Шаг 1. Определить характеристики взаимного расположения элементов схемы (камеры, окна просмотра и сцены).

Шаг 2. Выполнить следующие действия для каждой ячейки сетки канвы:

- выбрать на изображении пиксель, который соответствует текущей ячейке канвы;
- вычислить цвет части сцены, которая «видна» камерой через выбранную ячейку канвы;
- «заполнить» соответствующий ячейке канвы пиксель указанным цветом.

Шаг 3. Конец алгоритма.

Таким образом, задача получения изображения, согласно приведённому, алгоритму может быть распараллелена, поскольку пиксели и шаги получения их цветов независимы. Данная задача может быть выполнена на параллельной и распределённой вычислительной системе, в частности на РИСИВ [121].

Расчетные формулы трассировки лучей

Для реализации задачи получения двухмерных изображений по модели трехмерной сцены с использованием РИСИВ рассматривается Шаг 2 приведенного ранее алгоритма с учётом ранее принятых допущений по размещению и значений параметров окна просмотра и камеры. Модель освещения сцены не учитывается для простоты. В качестве сцены рассматривается модель одного объекта - сфера.

Преобразование к координатам точки пространства $V=(V_x,\ V_y,\ V_z)$ от координат пикселей канвы задаётся уравнением:

$$\begin{cases} V_x = C_x \dfrac{V_w}{C_w}, \\ V_y = C_y \dfrac{V_h}{C_h}, \\ V_z = d, \end{cases} \tag{4.1}$$

где C_w - ширина канвы, C_h - высота канвы, $C_x u C_y$ - соответственен, координаты x и y пикселя изображения (ячейки сетки канвы).

С учетом того, что луч является прямой, то используя параметрическое уравнение луча, можно задать его произвольную точку P:

$$P = O + t(V - O), \tag{4.2}$$

где t - произвольное действительное число.

Идущие из точки O лучи проходят через ячейку канвы и окно просмотра. Лучи соприкасаются с моделью сцены - сферой радиуса r и с центром C, что можно описать следующим уравнением:

$$|P - C| = r, \tag{4.3}$$

где P - точка на поверхности сферы.

Если в уравнении (4.2) выполнить замену $(V-O)=\vec{D}$, а также учесть, что $|P-C|$ - длина вектора $\overrightarrow{PC}$, которая рассчитывается как квадратный корень из его скалярного произведения на себя, то можно получить следующую систему уравнений:

$$\begin{cases} P=O+t\cdot\vec{D}, \\ \langle P-C;P-C\rangle=r,^2 \end{cases} \tag{4.4}$$

решением которой будет точка пересечения луча со сферой.

Преобразование уравнения к виду:

$$t^2\langle\vec{D},\vec{D}\rangle+t\left(2\cdot\langle\overrightarrow{OC},\vec{D}\rangle\right)+\langle\overrightarrow{OC},\overrightarrow{OC}\rangle, \tag{4.5}$$

где $\overrightarrow{OC}=O-C$,

при выполнении замены $a_1=\langle\vec{D},\vec{D}\rangle$, $a_2=2\langle\overrightarrow{OC},\vec{D}\rangle$ и $a_3=\langle\overrightarrow{OC},\overrightarrow{OC}\rangle$ получается квадратное уравнение вида:

$$a_1\cdot t^2+a_2\cdot t+a_3=0, \tag{4.6}$$

решением которого является

$$t_{1,2}=\frac{-a_2\pm\sqrt{a_2^2-4\cdot a_1\cdot a_3}}{2\cdot a_1}. \tag{4.7}$$

Имея значения $t_{1,2}$ из первого уравнения системы (4.2) получаем точку пересечения сферы и луча. Поскольку на схеме элементов трассировки луча сцена располагается за окном просмотра, то в качестве корней уравнения подойдут только значения t большие 1, так как только они описывают точки сцены (в случаях, если корень уравнения меньше нуля, то описывается точка, находящаяся позади камеры, а если корень положительный, но менее 1, то описывается точка, расположенная перед камерой, но до окна просмотра).

Так как для каждого пикселя изображения расчеты производятся независимо от других пикселей, именно процедуру вычисления точки можно выполнять в параллельном режиме, то для реализации всего алгоритма построения изображения с помощью трассировки луча на РИСИВ выбирается подпрограмма трассировки каждого отдельного луча.

Поскольку процедура определения точки пересечения заключается в нахождении корней уравнения (4.6), которое строится независимо для каждой точки, то для распределения заданий по узлам распределенной вычислительной системы можно выбрать подпрограмму, выполняющую трассировку отдельного луча (подпрограмма rayTrace).

Алгоритм построения изображения трехмерной сцены:

Шаг 0. Начало алгоритма.

Шаг 1. Определить начало координат и местоположение наблюдателя (камеры) - $O=(0,0,0)$.

Шаг 2. Выполнить следующие действия для каждой ячейки с координатами (x, y) сетки канвы:

- преобразование координат пиксела холста к координатной сетке окна наблюдения D = canvasToWindow(x,y);

- color = *rayTrace* (O, D, 1, ∞).

- paintPixel (x, y, color).

Шаг 3. Конец алгоритма.

Алгоритм подпрограммы вычисления цвета пикселя *rayTrace* :

Входные параметры подпрограммы (O, D, t_{min}, t_{max}):

Шаг 1. ближайший_t = ∞ .

Шаг 2. ближайший_объект = NULL.

Шаг 3. Для каждого Объекта сцены:

Шаг 3.1. $t_1, t_2 = transRayObject(O, D, объект)$

Шаг 3.2. Если t_1 в $[t_{min}, t_{max}]$ и t_1 < ближайший_t

тогда ближайший_t = t_1 .

ближайший_объект = Объект.

Шаг 3.3. Если t_2 в $[t_{min}, t_{max}]$ и t_2 < ближайший_t

тогда ближайший_t = t_2 .

ближайший_объект = Объект.

Шаг 4. Если ближайший_объект = NULL

тогда вернуть цветФона.

иначе вернуть цветОбъекта.

Шаг 5. Конец подпрограммы.

Алгоритм подпрограммы *transRayObject* :

Шаг 0. Входные параметры подпрограммы $(O, D, сфера)$

Шаг 1. $c = сфера.центр$.

Шаг 2. $r = сфера.радиус$.

Шаг 3. $OC = O - C$.

Шаг 4. $a_1 = \langle D, D \rangle$.

$a_2 = 2 \cdot \langle OC, D \rangle$.

$a_3 = \langle OC, OC \rangle - r^2$.

Шаг 5. $Дискриминант = a_2 \cdot a_2 - 4 \cdot a_1 \cdot a_3$.

Шаг 6. Если Дискриминант < 0

тогда вернуть ∞, ∞

иначе $t_1, t_2 = \dfrac{-a_2 \pm \sqrt{(Дискриминант)}}{2 \cdot a_1}$

вернуть t_1, t_2 .

Шаг 7. Конец подпрограммы.

Структура экспериментального стенда РИСИВ

С целью демонстрации работоспособности метода назначения заданий вычислительным узлам РИСИВ разработанный алгоритм распараллеливания вычислительной задачи получения изображения был протестирован на созданной экспериментальной установке.

При этом, поскольку системы Интернета вещей строятся на основе гетерогенных устройств, то они не ограничены в своем многообразии, и не

накладывают строгих ограничений на состав элементов как в самих системах Интернета вещей, так и в реализуемой экспериментальной установке.

Экспериментальная установка РИСИВ составлена из вычислительных узлов в виде аппаратных и виртуальных узлов общим количеством 15 шт. Аппаратные узлы представлены Raspberry Pi в количестве 10 шт., виртуальные — виртуальными машинами в количестве 5 шт. с характеристиками, аналогичными аппаратным узлам.

Все узлы находятся в одном сегменте сети, они соединены друг с другом с помощью следующих каналов связи:

- аппаратные узлы подключены через точку доступа сети WiFi;

- виртуальные узлы подключены через виртуальную сеть Ethernet, которая соединена с физической через мост.

Структура распределенной информационной системы представлена на рисунке 4.15.

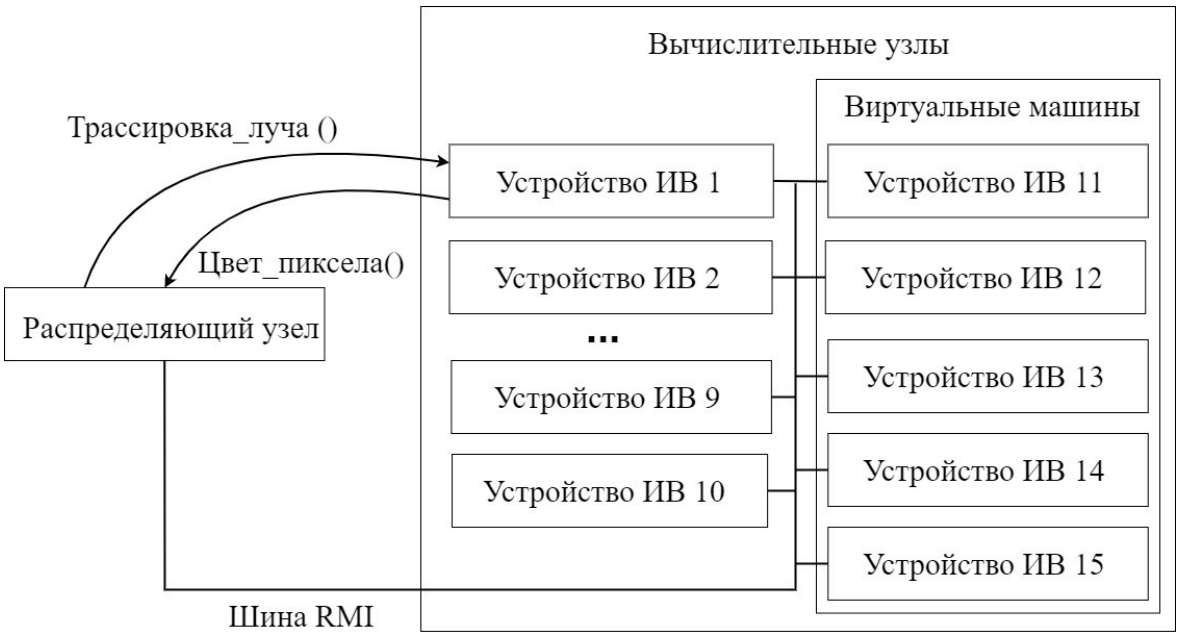

Рисунок 4.15. Архитектура стенда распределенной информационной системы на основе объектов ИВ

Согласно реализованным модели, методу и алгоритмам назначения, роль распределяющего узла для назначения вычислительных задач может выполнять

как шлюз, сервер/а управляющих ЦОД, облачные ЦОД, так и устройство ИВ и персональный компьютер.

В указанной структуре экспериментального стенда в качестве распределяющего узла используется персональный компьютер (характеристики которого соответствуют Raspberry Pi, что несущественно, так как он выполняет только задачу назначения заданий), подключенный по проводному каналу Ethernet к точке доступа WiFi.

Для нивелирования риска влияния особенностей цифровых платформ (на базе микропроцессоров Intel, ARM,) и компонент, их составляющих от различных производителей, на взаимодействие РИСИВ предлагается использование операционной системы GNU/Linux.

Поскольку широкое многообразие производителей компонент в каждом случае могут иметь свои несущественные особенности, которые, однако, в конечном итоге, могут приводить к непереносимости программного кода для каждой из платформ.

В данном случае используется для аппаратных узлов операционная система Raspbian, для виртуальных - Ubuntu.

Как указывалось ранее в пп. 4.1 и 4.2 в качестве инструментальных средст реализации была выбрана платформа Java, а в качестве передачи данных - реализованный в Java удаленный вызов процедур- RMI [108].

Использование платформы Java позволила разработать универсальный программный код, в котором можно программным образом изменять количество вычислительных узлов, без необходимости повторной компиляции и/или пересборки проекта. Так как устройства ИВ как правило имеют ограниченные возможности для хранения библиотек, то было протестирована возможность развертывания платформы Java ME с добавлением недостающих библиотек сетевого взаимодействия.

4.6 Результаты эксперимента и анализ экспериментальных данных

В качестве выходных характеристик, демонстрирующих функционирование распределённой информационной системы Интернета вещей по решению вычислительной задачи трассировки луча согласно предложенным и реализованным: модели РИСИВ, методу и алгоритму назначения заданий — были рассмотрены следующие характеристики эффективности:

- зависимость времени, потраченного на решение вычислительной задачи, от количества вычислительных узлов (от 2 до 15 устройств ИВ);

- зависимость времени, затраченного на обработку одного задания одним устройством ИВ, от количества вычислительных узлов;

- зависимость времени, затраченного на выполнения всей задачи по трассировке лучей, от количества пикселей двухмерного изображения.

Для получения зависимости времени, затраченного на выполнение всей задачи трассировки луча, от количества вычислительных узлов в РИСИВ распределяющий узел формирует совокупность последовательности заданий, число которых равно10 000 вследствие зафиксированной канвы изображения 100x100 пикселей. После каждое задание из последовательности назначается вычислительному узлу с наиболее высоким значением сигнала вознаграждения и отправляется ему.

При добавлении числа вычислительных узлов в РИСИВ время выполнения вычислительной задачи трассировки луча снижается (Рисунок 4.16). Замечено, что время решения вычислительной задачи является одинаковым в случае для двух и четырёх вычислительных узлов в распределённой информационной системе Интернета вещей.

Поскольку в случае наличия только двух вычислительных узлов в РИСИВ алгоритм на основе машинного обучения с подкреплением (глава) ограничен в выборе вычислительного узла их количеством на назначение задания, то есть накладные расходы по времени на пересчёт модели и выбор вычислительного узла минимальны.

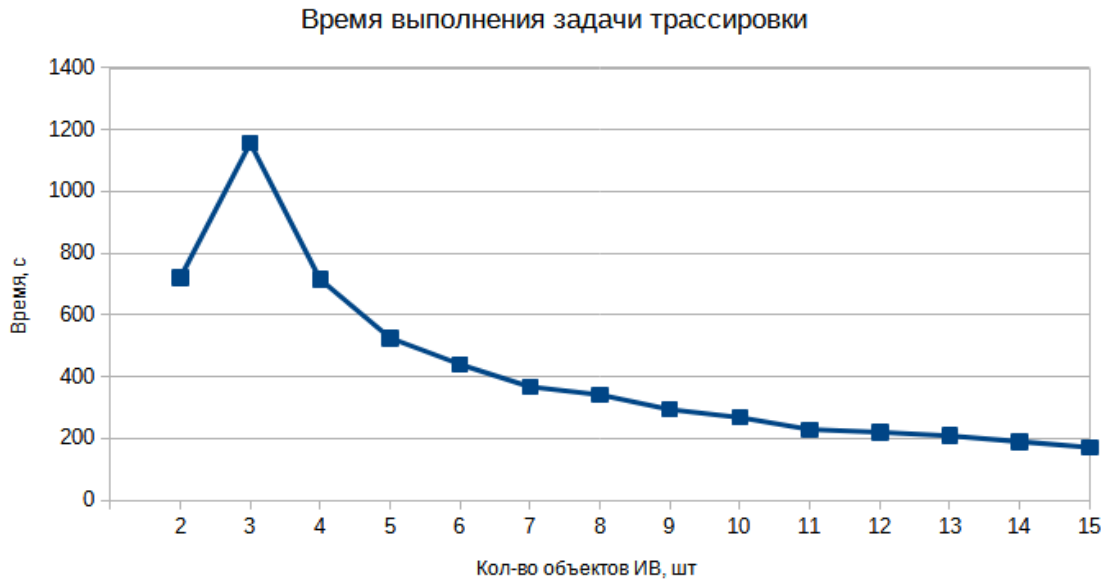

Рисунок 4.16. Время выполнения вычислительной задачи

В случае наличия трех вычислительных узлов производительность РИСИВ падает вследствие существенного увеличения накладных расходов по обработке вычислительной задачи — время выполнения вычислительной задачи увеличивается.

При увеличении вычислительных узлов до четырёх накладные расходы на назначение вычислительной задачи снижаются. Начиная от одиннадцати вычислительных узлов в РИСИВ время решения вычислительной задачи снижается, так же. как и скорость её выполнения.

Оценка времени выполнения одного задания вычислительной задачи трассировки луча на вычислительном узле РИСИВ формируется из времени:

- затраченного на определение алгоритмом назначения заданий вычислительного узла, предназначенного для обработки задания;

- непосредственного выполнения задания на вычислительном узле;

- переназначения задания другому вычислительному узлу, если задание не может быть обработано изначально назначенным вычислительным узлом.

На Рисунке 4.17 показана зависимость времени выполнения одного задания от количества вычислительных узлов в РИСИВ также при изображении в 100x100 пикселов (10 000 заданий).

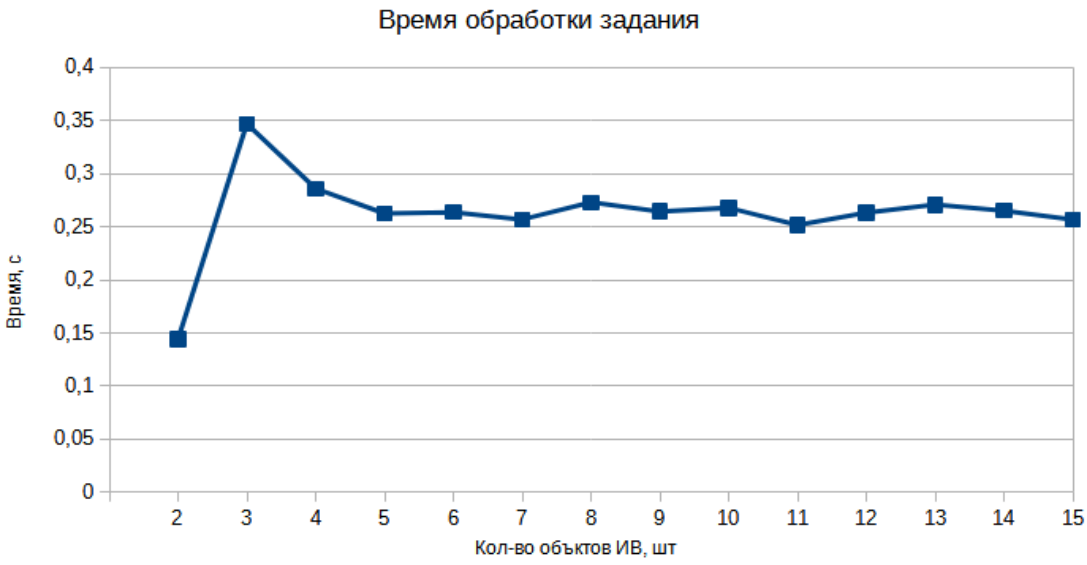

Рисунок 4.17. Время выполнения одного задания вычислительной задачи

Вследствие минимальных накладных расходов при двух вычислительных узлах, время исполнения заданий также сохраняется минимальным. Поскольку оценка времени выполнения задания состоит только из времени непосредственного функционирования самого вычислительного узла.

В случае, когда количество вычислительных узлов более пяти, то среднее время обработки одного задания имеет минимальный разброс. Учитывая накладные расходы на выбор, переназначение узлов, в случае отсутствия ответа от устройства ИВ, время выполнения одного задания несущественно отличается от среднего времени. С увеличением количества вычислительных узлов (более 5 устройств ИВ), когда алгоритм назначения заданий в РИСИВ находится в стабильном состоянии, то время выполнения одного задания перестает зависеть от количества используемых в РИСИВ устройств ИВ.

Оценка времени выполнения вычислительной задачи трассировки луча в зависимости от размера изображения проводилась при размерах изображения, начиная от 100x100 пикселов с увеличением шага в 50 пиксела по каждому

измерению до значения размера в 750x750 пикселов. Количество вычислительных узлов в РИСИВ в данном случае было неизменным и сохранялось равным десяти.

Результаты оценки времени выполнения вычислительной задачи трассировки луча в зависимости от размера изображения приведены на Рисунке 4.18.

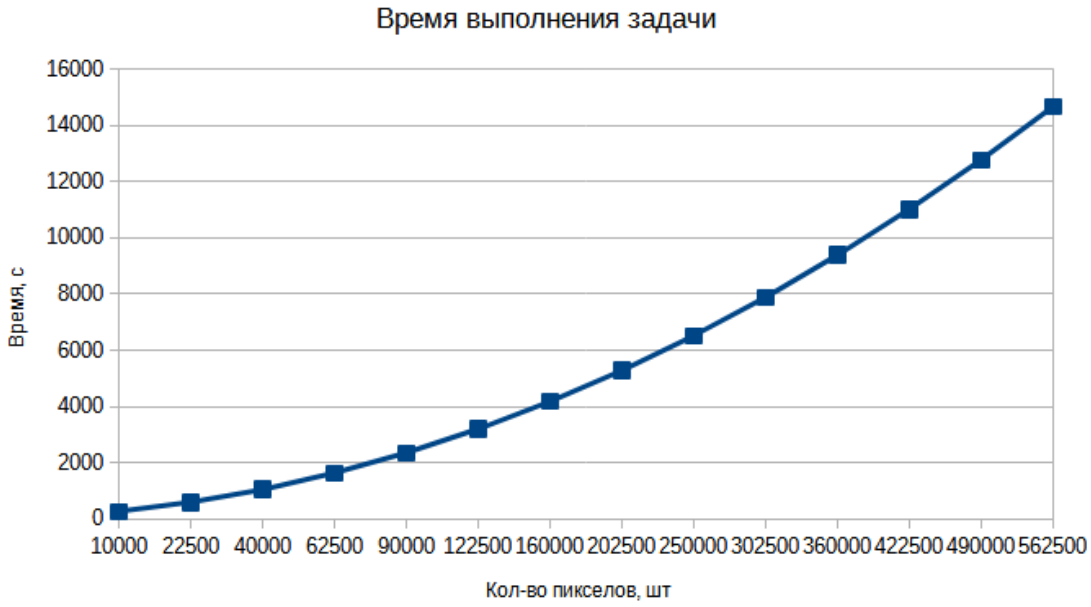

Рисунок 4.18. Время выполнения вычислительной задачи в зависимости от размера обрабатываемого изображения

В случае увеличения числа заданий время обработки, затраченное на построение одного изображения, также увеличивается в РИСИВ, при этом кривая графика времени растет быстрее, чем линейно (что соответствует закону Амдала).

Полученные в данной работе результаты соответствуют результатам исследований параллельных и распределенных вычислительных систем [122, 123, 124, 125], как в аспекте выполнения одного задания на вычислительном узле, так и в общем времени выполнения задачи на слабосвязанных вычислительных системах, построенных на основе классических подходов [63, 77], а также соответствуют результатам, которые получены исследователями в области суперкомпьютерных и облачных системах [126, 127].

Таким образом, проведенный эксперимент и полученные результаты демонстрируют возможность использования, а также подтверждают работоспособность распределённой информационной системы на основе Интернета вещей, как нового типа вычислительных мощностей, для выполнения вычислительных заданий на основе разработанных: модели, метода и алгоритма назначения.

4.7 Рекомендации по применению системы назначения заданий в РИСИВ

Анализ результатов эксперимента позволяет выработать ряд практических рекомендаций по использованию метода назначения заданий вычислительным узлам РИСИВ для выполнения вычислительной задачи:

1) величина параметра ε оказывает существенное влияние на скорость стабилизации системы, но в общем случае его значение не столь существенно, так как система в любом случае перейдет в стабильное состояние (только с различной скоростью), обеспечивая баланс между этапами исследования и эксплуатации, но тем не менее стоит избегать значений параметра, близкие к 0 и 1; в общем случае рекомендуется выбирать значения из диапазона $[0{,}1 \ldots 0{,}9]$ с шагом 0,1;

2) необходимо подобрать значение параметра ε в зависимости от поведения вычислительных узлов РИСИВ: при высокой степени изменяемости вычислительных узлов необходимо брать значение больше, если параметры вычислительных узлов меняются сравнительно редко, то значение ε взять меньше;

3) при существенном изменении параметров вычислительных узлов предложенный метод распределения заданий позволяет получить итоговый результат решения задачи независимо от закономерностей изменения параметров узлов; при увеличении количества узлов среднее значение вознаграждения будет приближаться к значениям при работе в стационарном режиме;

4) при большом количестве вычислительных узлов в РИСИВ (более 10) значение среднего времени выполнения одного задания достигает некоторой постоянной величины, что позволяет оценить общее время выполнения вычислительной задачи;

5) учитывая, что время выполнения задачи в зависимости от количества входных данных растет быстрее, чем линейная функция, то увеличение количества вычислительных узлов не ведет к такому же росту производительности;

6) для некоторых задач, требующих значительных вычислений, при увеличении количества вычислительных узлов в РИСИВ можно проводить их группировку и вводить промежуточные распределительные узлы на наиболее надежных устройствах ИВ, что позволит уменьшить величину широковещательного сетевого трафика в каналах передачи данных [108, 101, 109].

4.8 Выводы по четвертой главе

В результате исследования была проведена экспериментальная оценка эффективности различных режимов работы модели машинного обучения с подкреплением, в различных условиях и с различными значениями параметра. Для реализации назначения заданий были проанализированы различные подходы для взаимодействия распределительного узла с вычислительными узлами, в результате чего стало возможным реализовать функциональность РИСИВ в виде программного кода.

По результатам экспериментальной оценки алгоритмов стало возможным разработать ряд рекомендаций по использованию системы назначения заданий вычислительным узлам РИСИВ. Был разработан программный продукт по трассировке лучей для построения двухмерных изображений по модели сцены, который позволяет продемонстрировать применимость предложенного в работе подхода для осуществления решения вычислительной задачи на распределенной информационной системе, в качестве вычислительных узлов которой выступают

устройства Интернета вещей. Результаты работы программы трассировки лучей показали, что распределенные расчеты на РИСИВ соответствуют результатам, полученным для других распределенных вычислительных систем, и подтверждают предлагаемые в диссертационной работе предположения о возможности эффективного использования РИСИВ для решения вычислительных задач.

Общие выводы и заключение

К основным результатам диссертационной работы можно отнести следующее:

1. На основе анализа объектов Интернета вещей, распределенных вычислительных систем и методов машинного обучения сделан вывод о возможности построения РИСИВ на основе устройств Интернета вещей при использовании машинного обучения с подкреплением.

2. Формальная постановка задачи назначения (распределения) заданий позволяет подойти к рассмотрению вычислительной задачи в виде графа, который преобразуется в последовательность заданий, отправляемых вычислительным узлам РИСИВ.

3. На основе модели машинного обучения с подкреплением разработан метод назначения (распределения) заданий по узлам распределенной информационной системы; метод позволяет учитывать особенности каждого вычислительного узла и состояние каналов связи между ними.

4. На основе анализа стационарной/нестационарной среды и изменения e-жадной стратегии среди одного агента и множества действий для модели машинного обучения с подкреплением стало возможным построение алгоритма, который положен в основу РИСИВ.

5. Разработано программное обеспечение, реализующее метод назначения заданий вычислительным узлам на основе машинного обучения с подкреплением и позволяющее реализовать РИСИВ.

6. Для определения эффективности предложенного метода назначения заданий были проведены исследования поведения модели машинного обучения с подкреплением при различных вариациях параметров работы модели, а также по выполнению задачи трассировки луча и при модификации организации способа взаимодействия устройств ИВ с минимальным участием распределяющего узла. Анализируя полученные экспериментальные данные, были предложены практические рекомендации и методика по использованию РИСИВ на основе

машинного обучения с подкреплением для инженеров и исследователей, использующих устройства Интернета вещей в качестве распределенной информационной системы для решения вычислительных задач.

Дальнейшее развитие данной работы предполагает разработку подходов, позволяющих агенту использовать априорную информацию о структуре и поведении окружающей среды, что позволит учитывать возможные закономерности в поведении устройств ИВ (например, ослабление сигнала радиоканала для удаляющегося от базовой станции устройства, изменение устройством используемой базовой станции, маршрутизация и т. п.). Также будет модернизирована программная система путем введения унифицированных вычислительных примитивов на вычислительных узлах, что позволит расширить спектр задач в различных предметных областях, которые могут быть решены с использованием РИСИВ.

Список использованных источников

1. SETI@home//SETI@home. 2021. URL. https: www.setiathome.berkeley. edu/ (дата обращения 16.03.2021)

2. Folding@home// Folding@home. 2021.URL. https:www.foldingathome.org/ (дата обращения 16.03.2021)

3. Степанова М.В. Концепция Интернета вещей на базе платформы IBM Bluemix // Современные тенденции развития науки и технологий. Периодический научный сборник по материалам XII Международной научно-практической конференции. Белгород. 2016. №3-2. С.138-141.

4. Степанова М.В. Обеспечение безопасности в IoT системах // Современные тенденции развития науки и технологий. Периодический научный сборник по материалам XIX международной научно-практической конференции «Современные тенденции развития науки и технологий». Белгород. 2016. №10-1. С.112-115.

5. Kumar D., Verma C., Dahiya .S, Singh PK., Raboaca MS., Illés Z., Bakariya B. Cardiac Diagnostic Feature and Demographic Identification (CDF-DI): An IoT Enabled Healthcare Framework Using Machine Learning // Sensors. 2021. Vol.21, №19. P.6584.

6. Особенности и тенденции развития технологии LoRaWAN//QUEST: Дистрибьютор электронных компонентов. 2017. URL. https:www.icquest.ru/publications/4-semtech/2017/697-osobennosti-i-tendentsii-razvitiya-tekhnologii-lorawan-2017 (дата обращения 20.02.2020).

7. Строгалев В.П., Толкачёва И.О. Имитационное моделирование. М.: Издательство МГТУ им. Н. Э. Баумана, 2017. 295 с.

8. Брехов О.М., Звонарева Г.А., Корнеенкова А.В. Имитационное моделирование. М.: Издательство МАИ, 2015. 324 с.

9. Галанин М. П., Исаев А. В., Конев С. А. Математическая модель образования сажи при диффузионном горении толуола // Математическое моделирование. 2020. Т. 32, № 6. С. 66-80.

10. Хачумов В.М., Хачумов М.В. Конвейерные и разрядно-параллельные вычисления в бортовых системах навигации и управления. М.: КРАСАНД, 2019. 208 с.

11. Гонасалес Р., Вудс Р. Цифровая обработка изображений. М.: Техносфера, 2005. 1072 с.

12. Божко А.Н. Жук Д.М. Маничев В.Б. Компьютерная графика. М.: МГТУ им. Н.Э.Баумана, 2007. 396с.

13. Норенков И.П., Кузьмик П.К. Информационная поддержка наукоемких изделий. CALS-технологии. М.: МГТУ им. Н.Э. Баумана, 2002. 320с.

14. Ли К. Основа САПР (CAD/CAM/CAE). Спб.: Питер, 2004. 560 с.

15. Сюзев В.В. Основы теории цифровой обработки сигналов. М.: РТСофт, 2014. 749 с.

16. Оппенгейм А., Шафер Р. Цифровая обработка сигналов. М.: Техносфера, 2012. 1046 с.

17. The industrial internet of things (IIoT): An analysis framework / H. Boyes [et al.]. Computers in Industry. 2018. 101p.

18. Черняк Л. Платформа Интернета вещей // Открытые системы. СУБД. 2012. № 7. С. 44.

19. Leong Y., Fenn J. Key Trends to Watch in Gartner 2012 Emerging Technologies Hype Cycle // Forbes. 2012. URL. https:www.forbes.com/sites /gartnergroup/2012/09/18/key-trends-to-watch-in-gartner-2012-emerging-technologi es-hype-cycle-2/?sh=65edf8b67036 (дата обращения 10.11.2017).

20. An Extended Review on Internet of Things (IoT) and Its Promising Applications /Dr. Y. Perwej[et al.]. Communications on Applied Electronics. 2019.

21. Salman T., Jain R. A Survey of Protocols and Standards for Internet of Things // Advanced Computing and Communications. 2017. Vol. 1, No. 1.

22. Irons-Mclean R., Sabella A., Yannuzzi M. Orchestrating and Automating Security for the Internet of Things: Delivering Advanced Security Capabilities from Edge to Cloud for IoT. Cisco Press, 2018. 968 p.

23. Шевчук Е. В., Шевчук Ю. В. Современные тенденции в области хранения и обработки сенсорных данных // Программные системы: теория и приложения. 2015. №4. С. 157-176.

24. Kitchin R., McArdle G. What makes Big Data, Big Data? Exploring the ontological characteristics of 26 datasets. Big Data & Society. 2016.

25. Klonoff DC. Fog Computing and Edge Computing Architectures for Processing Data From Diabetes Devices Connected to the Medical Internet of Things // Journal of Diabetes Science and Technology. 2017. Vol.11, №4. P.647-652.

26. Fog Computing: An Overview of Big IoT Data Analytics /Rizwan Anawar M.R.[et al.]. Wireless Communications and Mobile Computing, 2018. 22p.

27. Yassein M., Hmeidi I., Shatnawi F., Rawasheh S.Fog Computing: Characteristics, Challenges and Issues //2020 International Conference on Mathematics and Computers in Science and Engineering (MACISE). 2020. P. 240-245.

28. Dineva K., Atanasova T. Design of Scalable IoT Architecture Based on AWS for Smart Livestock // Animals (Basel). 2021. Vol.11, №9.

29. Amazon, Google and Microsoft Solutions for IoT: Architectures and a Performance Comparison /Pierleoni P. IEEE Access. 2020.

30. Google. IoT & Devices // Google. URL. https:www.cloud.google.com /blog/products/iot-devices (дата обращения 01.04.2020).

31. Microsoft. What is Azure Internet of Things (IoT)? // Microsoft. URL. https:www.docs.microsoft.com/en-us/azure/iot-fundamentals/iot-introduction (дата обращения 02.04.2020).

32. Azraq A., Aziz A.H., Siddiqui U. Essentials of Application Development on IBM Cloud. An IBM Redbooks publication. 2017. 206 p.

33. Ju H., Liu L. Innovation Trend of Edge Computing Technology Based on Patent Perspective // Wireless Communications and Mobile Computing. 2021. Vol. 2021. P. 1-10.

34. Gumaei A., al-Rakhami M., Hassan M. M. Deep learning and blockchain with edge computing for 5G-enabled drone identification and flight mode detection // IEEE Network. 2021. Vol. 35, № 1. P. 94–100.

35. Olaniyan R., Fadahunsi O., Maheswaran M.,Zhani M. F. Opportunistic edge computing: concepts, opportunities and research challenges // Future Generation Computer Systems. 2018. Vol. 89, № DEC. P. 633–645.

36. Reznik A. What is Edge? //ETSI. 2018. URL. https:www.etsi.org/news room/blogs/entry/what-is-edge (дата обращения 05.04.2020).

37. Ericsson Mobility Report 2017. Ericsson AB, 2017. 36 p.

38. Knud Lasse Lueth .Top 10 IoT applications in 2020 // IOT ANALYTICS. 2020. URL. https:www.iot-analytics.com/top-10-iot-applications-in-2020/ (дата обращения 12.11.2020).

39. Ray P.P. A survey on Internet of Things architectures // Journal of King Saud University - Computer and Information Sciences. 2018. Vol.30. P.291-319.

40. Keeley T. IoT to IoAT: Internet of Autonomous Things devices provides solutions // Compsim. 2016. URL. https:www.controleng.com/articles/iot-to-ioat-internet-of-autonomous-things-devices-provides-solutions/ (дата обращения 12.11.2020).

41. Кили Т. От IoT к IoAT: Интернет автономных вещей // Control Engineering Россия. 2016. URL. https:www.controleng.ru/internet-veshhej/ioat/ (дата обращения 12.11.2020).

42. Gerber A., Romeo J. Choosing the best hardware for your next IoT project//IBM.IBM Developer.2020.URL. https:www.developer.ibm.com/articles/iot-lp101-best-hardware-devices-iot-project/ (дата обращения 12.11.2020).

43. Global Internet of Things(IoT) Chip Market – Growth, Trends, COVID-19 Impact, and Forecast (2021-2026)//Mordor Intelligence. 2020. URL. https:www.mordorintelligence.com/industry-reports/iot-chip-market (дата обращения 13.11.2020).

44. Интернет Вещей начинается там, где физический мир соединяется с цифровым// Analog Devices. URL. https:www.analog.com/ru/applications /technology/internet-of-things.html (дата обращения 14.11.2020).

45. Цифровизация в сельском хозяйстве: технологические и экономические барьеры в России Сентябрь 2017 года (Аналитический отчет) // JSON. TV. 2017. URL. https:www.json.tv/ict_telecom_analytics_view/tsifrovizatsiya-v-selskom-hozyaystve-tehnologicheskie-i-ekonomicheskie-barery-v-rossii-20170913024550 (дата обращения 15.11.2020).

46. Smart Sensors - Clever Data - Actionable Insights // iotag. URL. https:www.iotag.com.au/20170913024550 (дата обращения 16.11.2020).

47. Автономный IoT «Интернет вещей»//EnOcean. URL. http:www.eno-cean.ru/technology/self-powered-internet-of-things/ (дата обращения 17.11.2020).

48. Распоряжение от 22 октября 2021 г. № 2998-з. Правительство Российской Федерации. 2021. 14 с.

49. Ежова Н.А. Модель параллельных вычислений для оценки масштабируемости итерационных алгоритмов на кластерных вычислительных системах: дис. ... канд. тех. наук. Челябинск. 2019. 137 с.

50. Пахомова О.А. Математическое и программное обеспечение процессов параллельной и распределенной обработки графической информации в реальном масштабе времени: дис. ... канд. тех. наук. Воронеж. 2019. 129 с.

51. Колганов А.С. Автоматизация распараллеливания Фортран-программ для гетерогенных кластеров: дис. ... канд. тех. наук. Москва. 2020. 135 с.

52. Brodtkorb A.R. , Hagen T. R., Sætra. M. L. Graphics processing unit (GPU) programming strategies and trends in GPU computing // Journal of Parallel and Distributed Computing. 2013. Vol. 73. P. 4-13.

53. Huqqani A.A, Schikuta E., Ye S., Chen P.,Multicore and GPU Parallelization of Neural Networks for Face Recognition // Procedia Computer Science. 2013. Vol. 18. P. 349-358.

54. Bouzidi H., Ouarnoughi H., Niar S., Cadi A.A.E. Performance prediction for convolutional neural networks on edge GPUs//CF '21: Proceedings of the 18th ACM International Conference on Computing Frontiers. 2021. P.54-62.

55. Flynn M.J. Very high speed computers/Michael J. Flynn // Proceedings of the IEEE 54. 1966. №12. P.1901-1909.

56. Flynn M.J. Some computer organizations and their effectiveness // IEEE transactions on computers 100. 1972. № 9. P. 948-960.

57. Гергель В. П. Высокопроизводительные вычисления для многопроцессорных многоядерных систем. М.: Физматлит: Изд-во Моск. ун-та: Нижегородский гос. ун-т., 2010. 539 с.

58. Гергель В.П., Стронгин Р.Г. Основы параллельных вычислений для многопроцессорных вычислительных систем. НН.: Изд-во ННГУ им. Н.И. Лобачевского, 2003. 184 с.

59. Buyya R. High Performance Cluster Computing: Architectures and Systems. Prentice Hall, 1999.

60. Buyya R. High Performance Cluster Computing: Programming and Applications. Prentice Hall, 1999.

61. Roosta S.H. Parallel Processing and Parallel Algorithms: Theory and Computation. Springer-Verlag, 2000.

62. Wilkinson B., Allen M. Parallel programming. Prentice Hall, 1999.

63. Steen M. van, Tanenbaum A.S. Distributed Systems. distributed-systems.net, 2017.

64. Ходар А.А. Математическое и программное обеспечение процессов динамической балансировки нагрузки в распределенных облачных вычислениях: ... канд. тех. наук. Воронеж. 2020.

65. Тихомиров А.И. Методы и средства организации системы управления вычислительными заданиями в территориально распределенной сети суперкомпьютерных центров коллективного пользования: … канд. тех. наук. Москва. 2020.143 с.

66. Терехов Д.В. Математическое и программное обеспечение процессов управления потоками данных в гетерогенных информационных системах специального назначения: дис. ... канд.тех.наук. Воронеж. 2020. 161 с.

67. Singh M. An Overview of Grid Computing//2019 International Conference on Computing, Communication, and Intelligent Systems (ICCCIS). 2019. P. 194-198.

68. Guharoy R. A theoretical and detail approach on grid computing a review on grid computing applications//2017 8th Annual Industrial Automation and Electro-mechanical Engineering Conference (IEMECON). 2017. P. 142-146.

69. IEEE. IEEE Smart Grid Vision for Computing: 2030 and Beyond Roadmap. IEEE, 2016.

70. Agarwal P, Owzar K. Next Generation Distributed Computing for Cancer Research // Cancer Informatics. 2015. Vol.13. P. 97–109.

71. Phister G.F. In Search of Clusters. Prentice Hall PTR, 1998.

72. Rajak R. Cluster Computing: Emerging Technologies, Benefits, Architecture, Tools and Applications // International Journal of Advanced Science and Technology. 2020. Vol.29, № 4. P. 4008–4015.

73. Савяк В. Эффективные кластерные решения//iXBT.com. 2002. URL. https:www.ixbt.com/cpu/clustering.shtml

74. Tripp D. Performance measurement of distributed systems. University of Canterbury. 1986.

75. Tanenbaum A.S., Kaashoek M.F., Renesse R., Bal H. The Amoeba distributed operating system — A status report // Computer Communications. 1991. Vol.14. P.324-335.

76. Stepanova M. Applying Kolmogorov Complexity for High Load Balanc-ing Between Distributed Computing System Nodes // Conference proceedings of eLearning and Software for Education (eLSE). 2019. P. 376-382.

77. Воеводин В.В., Воеводин Вл.В. Параллельные вычисления. СПб.: БХВ-Петербург, 2002. 608 с.

78. Ершов А. П. Современное состояние теории схем программ // Проблемы кибернетики. 1973. № 27. С.87-110

79. Воеводин Вл.В. Статистический анализ и вопросы эффективной реализации программ // Вычислительные процессы и системы. 1993. № 9. С.249-301.

80. Демичев А.П., Ильин В.А., Крюков А.П. Введение в грид-технологии. М.: НИИЯФ МГУ, 2007. 87 с.

81. Li B., Wang M., Zhao Y., Pu G., Zhu H., Song F. Modeling and Verifying Google File System // Proceedings of IEEE International Symposium on High Assurance Systems Engineering. 2015. P. 207-214.

82. Ghemawat S., Gobioff H., Leung. S. The Google File System // Proceedings of the 19th ACM Symposium on Operating Systems Principles. NY. 2003. P. 20-43.

83. Cardellini V., Colajanni M., Yu P. S. Dynamic load balancing on Web-server systems // IEEE Internet Computing. 1999. Vol.3, №3. P. 28–39.

84. What Is Load Balancing? // NGINX. 2014. URL. https:www.nginx.com /resources/glossary/load-balancing/ (дата обращения 10.06.2020).

85. Zhang H. Architecture of Network and Client-Server model. ArXiv, 2013.

86. Mathur G., Desnoyers P., Chukiu P., Ganesan D., Shenoy P. Ultra-low power data storage for sensor networks // 2006 5th International Conference on Information Processing in Sensor Networks. 2006. P. 374-381.

87. Люгер Дж. Ф. Искусственный интеллект. Стратегии и методы решения сложных проблем. М.: Вильямс, 2003. 864 с.

88. Каллан Р. Основные концепции нейронных сетей. М.: Издательский дом Вильямс, 2003. 287 с.

89. Хайкин С. Нейронные сети. Полный курс. М.: Вильямс, 2006. 1103 с.

90. Sutton R.S., Barto A.G.Reinforcement Learning: An Introduction.Second Edition. MIT Press, 2018.

91. Карпенко А.П. Современные алгоритмы поисковой оптимизации. Алгоритмы, вдохновленные природой. М.: Изд-во МГТУ им. Н. Э. Баумана, 2021. 448 с.

92. Сахаров М.К. Методика проектирования программ для решения задач глобальной параметрической оптимизации на слабосвязанных вычислительных системах: дис. ... канд.тех.наук. Москва. 2020. 123 с.

93. Гельфанд И. М., Пятецкий-Шапиро И. И., Цетлин М. Л. О некоторых классах игр и игр автоматов // Докл. АН СССР. 1963. Т. 152, № 4. С. 845—848.

94. Джонс Т.М. Программирование искусственного интеллекта в приложениях. М.:ДМК Пресс, 2011. 312 с.

95. Алфимцев А.Н. Мультиагентное обучение с подкреплением. М.:Издательство МГТУ им.Н.Э.Баумана, 2021. 222 с.

96. Nair D.S., Supriya P. Comparison of Temporal Difference Learning Algorithm and Dijkstra's Algorithm for Robotic Path Planning// 2018 Second International Conference on Intelligent Computing and Control Systems (ICICCS). 2018. P. 1619-1624.

97. Ерёмин О.Ю., Степанова М.В. Распределение заданий по узлам вычислительной системы на платформе Интернета вещей на основе машинного обучения // Динамика сложных систем-XXI век. 2020. Т.14, №2. С. 84-92.

98. Степанова М.В., Ерёмин О.Ю. Организация распределённых вычислений в инфраструктуре Интернета вещей на основе методов машинного обучения с подкреплением // Математические Методы в Технике и Технологиях – ММТТ. 2020. Т.20, №3. С. 111-114.

99. Себеста Р.У. Основные концепции языков программирования. М.: Вильямс, 2001. 672 с.

100. Гайфулин Т.А., Новиков А.С. Формальный язык описания задачи в распределенных вычислительных системах // Извести ТулГУ. Технические науки. 2014. Вып. 11, Ч. 2. С. 254-258.

101. Степанова М.В. Модифицированный метод назначения заданий в распределённой вычислительной системе интернета вещей // Современные наукоемкие технологии. 2021. № 9. С. 125-132.

102. Степанова М.В., Ерёмин О.Ю. Назначение заданий узлам распределенной системы платформы интернета вещей на основе машинного обучения с подкреплением // Автоматизация процессов управления. 2021. № 1 (63). С. 27-33.

103. Eremin O., Stepanova M. A reinforcement learning approach for task assignment in IoT distributed platform // Cyber-Physical Systems. Studies in Systems. Decision and Control. 2021. Vol.350. P.385-394.

104. Atiyah I., Mohammadpour A.,Taheri S. KC-Means: A Fast Fuzzy Clustering // Advances in Fuzzy Systems. 2018. Vol. 2018. P.1-8.

105. Fauzdar P.,Kindri F. Comparitive Analysis Of K Means And Fuzzy C Means Algorithm // International Journal of Engineering Research & Technology (IJERT). 2013. Vol. 2.

106. Ren M., Liu P., Wang Z., Jing Y. A Self-Adaptive Fuzzy c-Means Algorithm for Determining the Optimal Number of Clusters // Computational Intelligence and Neuroscience, 2016. Vol. 2016.

107. Gosling J. Java: an Overview the original Java whitepaper. 1995.

108. Степанова М.В. Способы реализации взаимодействия между приложениями Интернета вещей // Стратегии исследования в естественных и технических науках: сборник научных трудов по материалам международной научно-практической конференции. Белгород: ООО Агентство перспективных научных исследований (АПНИ). 2018. №10, Ч.1. С.146-152.

109. Stepanova M., Eremin O. Universal Multi-platform Interaction Approach for Distributed Internet of Things // The International Conference on Deep Learning, Big Data and Blockchain (Deep-BDB 2021). Deep-BDB 2021. Lecture Notes in Networks and Systems. 2022. Vol.309. P.147-159.

110. Arndt O., Freisleben B., Kielmann T., Thilo F. A comparative study of online scheduling algorithms for networks of workstations // Cluster Computing. 2000. Vol.3. P. 95–112 (2000).

111. Padole M., Shah A. Comparative Study of Scheduling Algorithms in Heterogeneous Distributed Computing Systems // Advanced Computing and Communication Technologies. 2018. Vol. 562. P.111-122.

112. Khaldi D., Jouvelot P., Ancourt C. Parallelizing with BDSC, a resource-constrained scheduling algorithm for shared and distributed memory systems // Parallel Computing. 2015. Vol. 41. P. 66 –89.

113. Kwok Y., Ahmad I. Static scheduling algorithms for allocating directed task graphs to multiprocessors // ACM Computing Surveys. 1999. Vol. 31, № 4. P. 406–471.

114. Choudhury P., Chakrabarti P., Kumar R. Online scheduling of dynamic task graphs with communication and contention for multiprocessors // IEEE Transactions on Parallel and Distributed Systems. 2012. Vol.23, № 1. P. 126–133.

115. Tang Z., Jiang L., Zhou J., Li K., Li K. A self-adaptive scheduling algorithm for reduce start time // Future Generation Computer Systems. 2014. Vol.43-44. P. 51-60.

116. Hammah R.E., Curran J.H. Fuzzy cluster algorithm for the automatic identification of joint sets // International Journal of Rock Mechanics and Mining Sciences. 1998. Vol. 35. P. 889-905.

117. Zhou K., Yang S. Fuzzifier Selection in Fuzzy C-Means from Cluster Size Distribution Perspective // Informatica. 2019. Vol. 30, №. 3. P.613-628.

118. Caelen O., Bontempi G. Improving the exploration strategy in bandit algorithms // Learning and Intelligent Optimization. 2008. P. 56-68.

119. Ерёмин О.Ю., Степанова М.В., Пролетарский А.В. Трассировка лучей на распределённой вычислительной системе на основе объектов Интернета вещей // Современные наукоемкие технологии. 2021. № 8. С. 72-80.

120. Gambetta G. Computer Graphics from scratch. Gabriel Gambetta. 2021

121. Система распределенных вычислений трассировки луча на основе Интернета вещей: свидетельство о государственной регистрации программы для ЭВМ 2021663723 РФ / М.В. Степанова; заявл.13.08.21;зарег.23.08.21.

122. Dhulipala L., Blelloch G. E., Shun J. Theoretically Efficient Parallel Graph Algorithms Can Be Fast and Scalable // ACM Transactions on Parallel Computing. 2021. Vol. 8, № 1. P.1-70.

123. Nagashima U., Hyugaji S., Sekiguchi S., Sato M., Hosoya H. An experience with super-linear speedup achieved by parallel computing on a workstation cluster: Parallel calculation of density of states of large scale cyclic polyacenes // Parallel computing. 1995. Vol. 21, №. 9. P. 1491-1504.

124. Zhang S., Xia Z., Yuan R., Jiang X. Parallel computation of a dam-break flow model using OpenMP on a multi-core computer // Journal of hydrology. 2014. Vol. 512. P. 126-133.

125. Rautenbach C., Mullarney J. C., Bryan K. R. Parallel computing efficiency of SWAN 40.91 // Geoscientific Model Development. 2021. Vol. 14, №. 7. P. 4241-4247.

126. Komatsu K. [et al.]. Performance Evaluation of a Vector Supercomputer SX-Aurora TSUBASA // SC18: International Conference for High Performance Computing, Networking, Storage and Analysis. 2018. P.685-696.

127. Shen C., Tong W., Choo KK.R. [et al.]. Performance prediction of parallel computing models to analyze cloud-based big data applications // Cluster Computing. 2018. Vol. 21. P. 1439–1454.