Федеральное государственное автономное
образовательное учреждение
высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт космических и информационных технологий
институт

Вычислительная техника
кафедра

УТВЕРЖДАЮ
Заведующий кафедрой

_____ О. В. Непомнящий
подпись инициалы, фамилия
« ____» _____ 2020 г.

МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ

Способы управления задачами в распределенных децентрализованных мобильных системах
Тема

09.04.01 «Информатика и вычислительная техника»
код и наименование направления

09.04.01.06 «Микропроцессорные системы»
код и наименование магистерской программы

Руководитель	_____	зав. каф. ВПВ, канд.техн.наук., доцент	Д.А. Кузьмин
	подпись, дата	должность, ученая степень	инициалы, фамилия
Выпускник	_____		И.В. Якимов
	подпись, дата		инициалы, фамилия
Нормоконтролер	_____		Д.А. Кузьмин
	подпись, дата		инициалы, фамилия
Рецензент	_____	канд.техн.наук., доцент	Е.Ю. Белоголовкин
	подпись, дата	должность, ученая степень	инициалы, фамилия

Красноярск 2020

Федеральное государственное автономное
образовательное учреждение
высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт космических и информационных технологий
институт

Вычислительная техника
кафедра

УТВЕРЖДАЮ
Заведующий кафедрой

_____ О.В. Непомнящий
подпись инициалы, фамилия
« _____» _____ 2020 г.

ЗАДАНИЕ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ
в форме магистерской диссертации

Студенту Якимову Игорю Владимировичу

Группа КИ18-01-6М. Направление (специальность) 09.04.01, «Информатика и вычислительная техника».

Тема выпускной квалификационной работы: «Способы управления задачами в распределенных децентрализованных мобильных системах».

Утверждена приказом по университету № 19249/с от 21.12.2018
Руководитель ВКР Д.А. Кузьмин, канд. техн. наук, доцент, заведующий кафедрой «Высокопроизводительных вычислений».

Исходные данные для ВКР: задание на ВКР.

Перечень разделов ВКР: Анализ объекта исследования, Исследование системы и построение архитектуры прототипа, Техническая реализация прототипа, заключение.

Перечень графического или иллюстративного материала с указанием основных чертежей, плакатов, слайдов: презентационные слайды pdf.

Руководитель ВКР	_____ подпись, дата	Д.А. Кузьмин инициалы, фамилия
Задание принял к исполнению	_____ подпись, дата	И.В. Якимов инициалы, фамилия

« ___ » _____ 2020 г.

РЕФЕРАТ

Выпускная квалификационная работа магистра по теме «Способы управления задачами в распределенных децентрализованных мобильных системах» содержит в себе 39 страниц текстового документа, 23 использованных источника, 13 иллюстраций.

РАСПРЕДЕЛЕННЫЕ ВЫЧИСЛЕНИТЕЛЬНЫЕ СИСТЕМЫ, МОБИЛЬНЫЕ СИСТЕМЫ, ПРОТОТИПИРОВАНИЕ, КЛИЕНТ-СЕРВЕРНОЕ ПРИЛОЖЕНИЕ.

Цель работы: исследование поведения самоорганизующихся распределенных вычислительных систем в условиях переменного количества узлов-клиентов, и разработка прототипа такой системы на классе устройств на платформе android..

Документ содержит предварительные исследования предметной области, описание обоснований разработки, описание модели для создания прототипа, работа прототипа.

Научная новизна диссертационного исследования заключается в следующем:

1. Предложена архитектура распределенной вычислительной системы для мобильных устройств на базе ОС Android

2.. Разработан прототип данной системы, работа которого основывается на исследованиях данной работы.

В итоге реализован прототип, отвечающий вышеописанным требованиям.

СОДЕРЖАНИЕ

АННОТАЦИЯ

Выпускная квалификационная работа посвящена созданию решения по организации распределенной вычислительной среды на мобильных устройствах работающих под управлением ОС Android.

В работе представлена модель системы. Реализован архитектура и прототип. Система строится на клиент-серверной архитектуре, и должна справляться с ресурсоёмкими задачами путём передачи и распределения задач между узлами-клиентами.

Представлен прототип системы с помощью которого можно оценить ее возможности и провести численные эксперименты

ВВЕДЕНИЕ

Актуальность работы. Постоянно расчет спрос на компьютеры, работающие с все более и более высокой скоростью. Фармацевты постоянно разрабатывают новые лекарственные вещества и препараты с помощью компьютера. Астрономы пытаются воспроизвести историю Вселенной, начиная с большого взрыва и заканчивая сегодняшним днем. Разработчики летательных аппаратов могли бы получать более точные результаты, не строя огромные аэродинамические трубы, а всего лишь проектируя свои конструкции на компьютере. Не смотря на все возрастающую мощность компьютеров, для решения многих нетривиальных задач (промышленных, научных, технических) никогда не хватит их возможностей. [1]

Особенно актуальным является мобильная отрасль, которая семимильными шагами развивается в последнее время. Вместе с ней растет и производительность мобильных платформ. Несомненно, мобильным платформам далеко до производительности домашних компьютеров, процессоры которых построены на другой архитектуре, но, несмотря на это, их средний рост производительности более высокий, чем рост производительности процессоров в ПК.

Сфера параллельных распределенных вычислений с каждым годом растет и развивается, однако никто не затрагивает нишу создания таких вычислений, построенных на мобильных устройствах. К тому же, очень сильно развивается отрасль, связанная с разработкой умных домов, распознаванием образов, сканирования трехмерных объектов. Соответственно растут сложности задач, связанных с данными отраслями. Для таких задач нередко требуется вычислительный потенциал немобильной высокопроизводительной вычислительной системы.

Актуальность темы, в основном, обусловлена развитием технологий и концепцией интернета вещей, смысл которых заключается в построении сетей между физическими устройствами, которые, как правило, участвуют во

взаимодействии со внешним миром, и которые должны уметь решать вышеописанные задачи.

Основной целью является исследование поведения самоорганизующихся распределенных вычислительных систем в условиях переменного количества узлов-клиентов, исследование управления распределением задач между устройствами-клиентами, и разработка прототипа такой системы на классе устройств на платформе Android.

1 АНАЛИЗ ОБЪЕКТА ИССЛЕДОВАНИЯ

1.1 Анализ объекта исследования

1.1.1 Распределенные вычислительные системы

Распределенные вычислительные системы — физические компьютерные, а также программные системы, реализующие тем или иным способом параллельную обработку данных на многих вычислительных узлах. [2]

Главное устройство-сервер занимается решением задач по анализу принятых данных, распределению этих данных между клиентами системы, а также хранит информацию о спецификации каждого из устройств, состояния надежности его сетевой связи в системе.

Как и любая другая система, состоящая из атомарных вычислительных компонентов, она включает в себя работу с модулями событий, которые обслуживают работу принятия данных, отправку и проверку целостности приятых/отправленных данных.

1.1.2 ARM

Самой широко используемыми процессорами в большинстве мобильных устройств являются процессоры на архитектуре ARM (Advanced RISC Machine).

В семействах процессоров ARM Cortex-A есть возможность использования специального модуля VFP, который позволяет выполнять операции с плавающей точкой, что обуславливает их применимость для широкого спектра задач в области высокопроизводительных вычислений. [18]

Именно наличие данного модуля, хорошая энергоэффективность самих процессоров, и возможность решать огромное количество задач в области высокопроизводительных вычислений. [19]

1.1.3 Динамика развития мобильных процессоров

За ближайшие 10 лет, скорость развития производительности процессоров на архитектуре ARM, значительно выше, чем у процессоров на архитектуре, производной x86_64, используемой в чипах intel и amd.

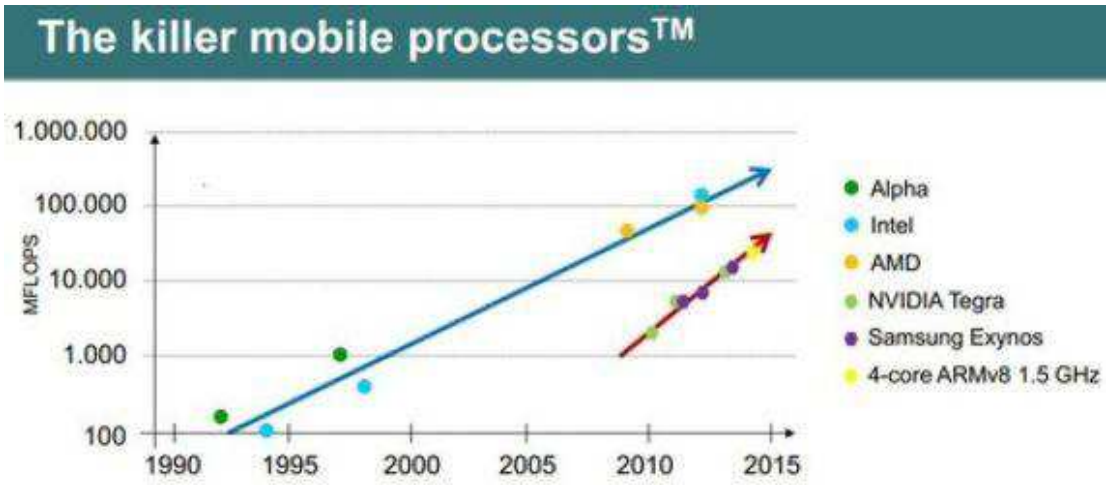

Рисунок 1 – График роста производительности различных процессоров

В результате исследования, проведенного испанским центром Barcelona Supercomputing Center, в ближайшее время будет наблюдаться тенденция, связанная с развитием мобильных микропроцессоров, которая была аналогичной с RISC и CISC процессорами в конце 20 века. [17]

Исследователи прогнозируют большой спрос на процессоры, используемые в мобильных устройствах. Такой спрос обусловлен низкой ценой на один кристалл, большим спектром применимости процессора и быстрым ростом вычислительной скорости.

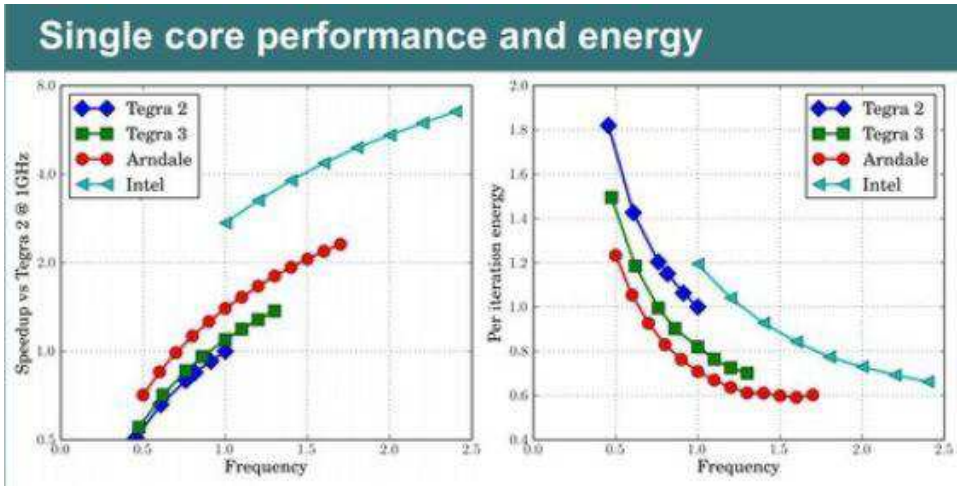

Рисунок 2 – Графики результатов тестов производительности и потребления энергии одного ядра на процессорах Intel, Tegra, Arndale

Из приведенных графиках выше, можно сделать вывод, что процессоры, разработанные на архитектуре ARM для мобильных устройств, более энергоэффективны, по сравнению с процессорами от intel.

Также, в испанском центре, благодаря финансовой поддержке испанского правительства и ЕС, в настоящее время уже проектируются и ведутся разработки серверов, работающих на базе актуальных мобильных ARM процессоров.

1.1.4 Мобильные устройства

Мобильные устройства, на основе которых будут производиться вычисления, обладают, как правило, набором беспроводных интерфейсов средств связи, таких как, Bluetooth, Wi-Fi (стандарт IEEE 802.11).

Большинство устройств по состоянию на 2019 год обладают чипами с реализацией Wi-Fi последнего поколения (выше 802.11n, со скоростью передачи данных до 600 мбит) [3].

В качестве операционной системы на устройствах будет использоваться система Android с открытым исходным кодом. Прикладные приложения на этой операционной системе описываются на языке программирования Java.

Библиотеки и наборы инструментов, которые предназначены для разработки под данную платформу позволяют получить доступ к созданию собственных клиент-серверных приложений в любой локальной сети.

Вычислительные мощности мобильных платформ в некоторых случаях могут быть быстрее, чем ПК, а системы, построенные на них, могут быть более отзывчивы за счет того, что процессоры архитектуры ARM, на которых базируются мобильные устройства , обладают меньшим числом команд на кристалле (в среднем, на 30%)[4], что позволяет им быть быстрее в ряде выполнения обычных инструкций, для нужд поддержания ОС.

1.1.5 ОС Android

Android – операционная система, разработанная компанией google. Основана на ядре операционной системы linux, и программном обеспечении, разработанном в google.

Операционная система преимущественно используется в смартфонах, однако, в связи с огромной популярностью, есть реализации под множество таких устройств, как умные часы, планшеты, ноутбуки, телевизоры. А также, имеет 2,5 миллиардов активных пользователей, на момент написания работы.

Для среды выполнения прикладных приложений используется java virtual machine. Каждое выполняемое приложение внутри системы находится в отдельном экземпляре виртуальной машине, что позволяет изолировать процессы друг от друга.

Для связи между процессами могут использоваться используется встроенная библиотека, работающая для передачи данных с ядром ОС через системные вызовы.

1.1.6 Java Virtual Machine

Java Virtual Machine (далее JVM) представляет из себя среду для выполнения программ, разработанных на языке java. [23]

Имеет множество разных реализаций под различные архитектуры процессоров и операционных систем, на которых выполняется.

Каждая программа, написанная на языке java преобразуется компилятором в объектный код, который подается на вход JVM. Виртуальная машина, в свою очередь, умеет транслировать полученный код в двоичные файлы формата .class, и затем преобразовывать его в язык понятный операционной системе и конечному устройству, путём интерпретации полученного бинарного файла.

В качестве реализации java виртуальной машины на платформе android выступает проприетарная реализация – android runtime.

1.2 Обзор и современное состояние предметной области

1.2.1 BOINC

BOINC – открытая программная платформа для организации распределенных вычислений.

Состоит из серверной и клиентской частей. Первоначально разрабатывался для крупнейшего проекта добровольных вычислений — SETI@home, но впоследствии разработчики из Калифорнийского университета в Беркли сделали платформу доступной для сторонних проектов. На сегодняшний день BOINC является универсальной платформой для проектов в области математики, молекулярной биологии, медицины, астрофизики и климатологии. BOINC даёт исследователям возможность задействовать огромные вычислительные мощности персональных компьютеров со всего мира. [5]

Прежде всего BOINC позиционируется как система для grid вычислений связанных с решением конкретно научных задач.

Данная реализация системы позволяет использовать вычислительную мощность удаленных компьютеров, однако имеет несколько недостатков, среди которых можно отметить тот факт, что устройства, задействованные в вычислениях, могут быть недостаточно надежными в плане производительности и отказоустойчивости.

Исходя из спецификации BOINC, приложение-клиент может использовать сети 2g/3g/4g, а также Wi-Fi для создания системы.

Также BOINC поддерживает мониторинг выполнения вычислительных процессах на узлах системы. [6]

1.2.2 HTC Power to give

HTC power to give – система добровольных вычислений от компании HTC, представленная в 2014 году для устройств на базе android. [11]

В действительности, power to give основана на технологии BOINC, поэтому в качестве условий для предоставления устройством вычислительной мощи требуется заряженный смартфон, а также подключение через Wi-FI.

Исходя из этого, к недостаткам данной реализации, можно отнести все недостатки, которые присущи системе BOINC.

Поддержка данной системы не осуществляется с 2016 года.

1.2.3 Dreamlab

Drealmab – приложение, позволяющее использовать вычислительную мощность android смартфонов для добровольных вычислений, разработанная компанией Vodafone.

Устройства, с установленным dreamlab объединяются в единую GRID систему для решения общих задач.

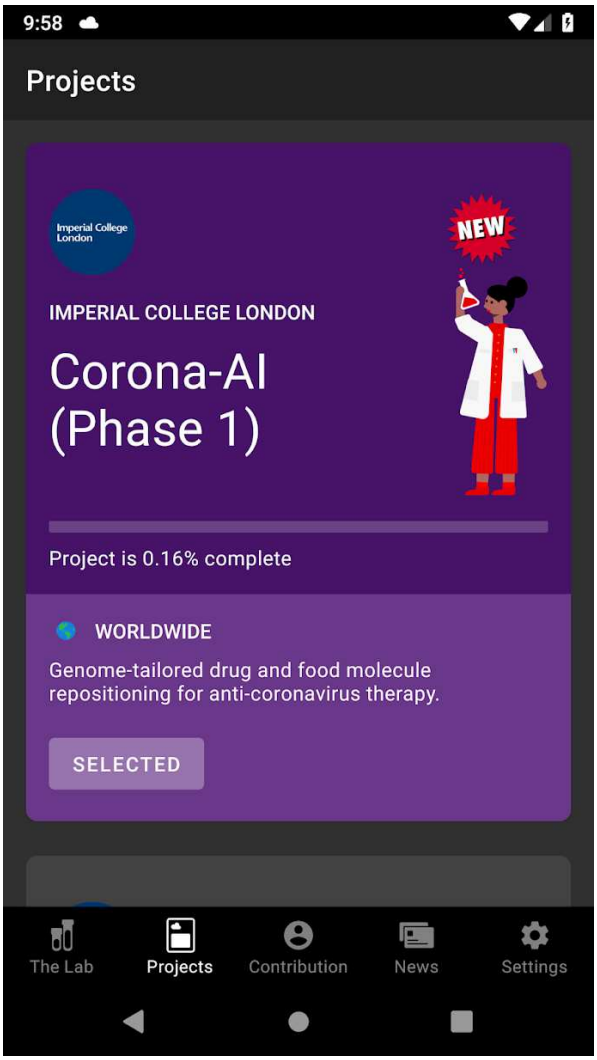

Рисунок 3 – Интерфейс Dreamlab в процессе выполнения добровольных вычислений

Вычислительная мощность, предоставляемая устройствами-клиентами через dreamlab преимущественно используется для медицинских исследований различных паталогий, с которым борется человечество в настоящий момент.

По состоянию на середину 2020 года, большинство всех задач, решаемые добровольными вычислениями данной системы, связаны с расшифровкой ДНК раковых клеток, и вируса covid-19. [10]

1.2.4 MontBlanc

Mont-blanc - это набор европейских проектов, стартовавших в 2011 году, по разработке серверов, кластеров и суперкомпьютеров на энергоэффективных мобильных ARM процессорах.

Основной целью проектов mont-blanc является достижение максимальной энергоэффективности, и минимальных затрат на процессоры, на высокопроизводительных вычислительных системах, по сравнению с актуальными решениями.

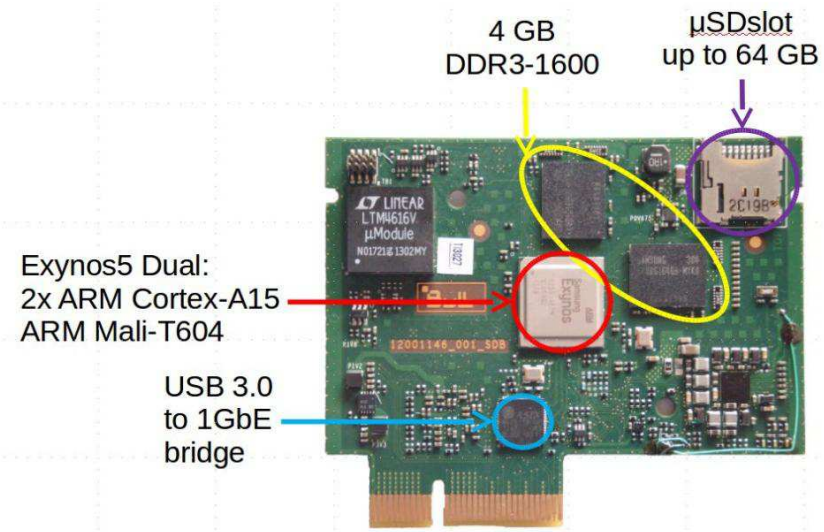

Рисунок 4 – Серверный модуль Mont Blanc размером 8.5 x 5.6 см

Три первых проекта, до 2018 года, продемонстрировали жизнеспособность кластеров при данной реализации, а поспособствовали созданию соответствующей программной экосистемы. [19]

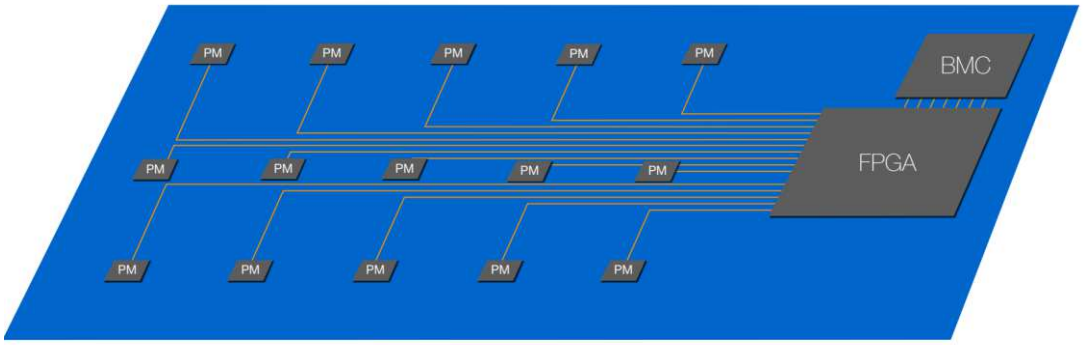

Рисунок 5 – Схема архитектуры мониторинга энергопотребления Mont Blanc

Для мониторинга и анализа энергопотребления, каждый серверный модуль параллельно отправляет данные об энергопотреблении на специальную ПЛИС. После обработки, плис отправляет данные на контроллер управления BMC (Board management controller), который сохраняет среднее значение энергопотребления, посчитанное за 1 секунду. Затем усредненное значение отправляется в распределенное хранилище ключ-значений, которое является также базой данных для анализа энергопотребления.

1.3 Постановка задач

Для исследования работы управления задачами, и адаптации к переменному количеству клиентов-узлов высокопроизводительной вычислительной системы, построенной на базе мобильных устройств на ОС Android, необходимо разработать прототип данной системы.

Для достижения этой цели, потребуется произвести:

1. Обзор существующих средств разработки
2. Обзор интерфейсов для коммуникации между узлами системы
3. Создание технического задания прототипа
4. Описание требований к конечной системе
5. Описание требований к видам обеспечения
6. Описать архитектуру работы данной системы
7. Выделение конкретного размера единицы выполняемой задачи в системе
8. Описание работы серверного приложения
9. Описание работы клиентского приложения
10. Разработку прототипа на основе описания работы и архитектуры конечной системы

1.4 Выводы к главе

В результате работы над данной главой, была изучена предметная область, необходимая для выполнения исследования.

Сформулированы и поставлены конкретные задачи для достижения основной цели выпускной квалификационной работы.

2 ИССЛЕДОВАНИЕ СИСТЕМЫ И ПОСТРОЕНИЕ АРХИТЕКТУРЫ ПРОТОТИПА

2.1 Обзор средств разработки для прототипа

2.1.1 Kotlin

Kotlin – достаточно новый мультипарадигменный язык программирования общего назначения, разработанный компанией Jetbrains. [15]

Программы, написанные на данном языке, скомпилированные в байт-код могут быть выполнены виртуальной машиной Android ART, а также язык может импортировать в свой код и использовать уже существующие библиотеки, написанные на языке java.

С момента релиза, в 2017 году, компания Google объявила о том, что данный язык является официальным, для разработки приложений на ОС android.

2.1.2 Java

Java – объектноориентированный язык программирования со строгой типизацией, разработанный компанией sun microsystems. Используется для разработки прикладных приложений ОС android с момента первого релиза системы.

Основным языком для разработки прототипа был выбран язык Java, так как он значительно старше, чем kotlin, и, соответственно наиболее стабилен для реализации любых программных целей. Также, в меру возраста, на данном языке уже существует огромное количество библиотек, имеющих документацию с примерами, написанными на самом java.

2.1.3 OpenMP

OpenMP является стандартом для описания параллельных программ для языков C, C++, fortran, работающий на SMP системах. [21]

При разработке на OpenMP, разработчик может не задумываться о количестве ядер, процессоров и потоков у системы, под которую пишется программа, так как стандарт позволяет описывать программы таким образом, что они могут выполняться и на однопроцессорных системах в режиме последовательного выполнения операций.

OpenMP позволяет достичь параллелизма с помощью многопоточности, в которой главный поток создает набор потоков-клиентов, и задача распределяется между ними. Предполагается, что потоки выполняются параллельно на машине с несколькими процессорами (количество процессоров не обязательно должно быть больше или равно количеству потоков).

Задачи и данные, выполняемые и обрабатываемые потоками, требуемые для выполнения , описываются с помощью специальных директив препроцессора соответствующего языка — прагм.

Изначально стандарт разработан для достижения параллелизма, в котором каждый параллельный поток имеет доступ ко данным всех потоков.

2.1.4 MPI

MPI - Message passing interface является API для обеспечения связи между ветвями параллельного приложения. [22]

MPI позволяет разработчикам использовать единый механизм описания для распараллеливания различных приложений, абстрагированный от архитектуры конкретного устройства, количества его ядер и тому подобное.

В основе механизма лежит приём и передача сообщений между конечными вычислительными узлами. Сообщение несёт в себе передаваемые данные и информацию, позволяющую принимающей стороне осуществлять

их выборочный приём. Передача сообщений между узлами может быть блокирующей и нелокирующей.

С помощью MPI можно достичь такого паралелизма, в котором каждый параллельный процесс работает в своем собственном адресном пространстве изолированно от других.

Так как среднее время выполнения вычислений, на примере задачи поиска данных MapReduce, меньше у интерфейса MPI, чем у OpenMP, было решено использовать его для потенциального распараллеливания задач на конечном устройстве-клиенте [16]

2.2 Обзор интерфейсов соединения клиентов

2.2.1 USB

Шина или интерфейс USB (Universal Serial Bus) — это универсальная последовательная шина, предназначенная для подключения периферийных устройств. В свое время шина USB пришла на смену уже морально устаревшим интерфейсам COM, LPT, PS/2 и GamePort. [13]

Современной и актуальной версией, которая преимущественно используется в новых устройствах на android - usb спецификации 3.1 с симметрическим разъёмом type-c.

Так как USB 3.1 версии с типом разъёма type c зарекомендовал себя относительно недавно, в большинстве смартфонов используется интерфейс usb 2.0, поддерживающий максимальную скорость до 480 мбит/с.

USB 2.0 — активно используется до сих пор в двух типах разъемов: Type-A и Type-B. На мобильных гаджетах используются их более компактные версии: Micro-B и Mini-B. Предельная скорость в USB 2.0 ограничена 480 Мбит/с, максимальный ток — 500 мА;

2.2.2 Wi-Fi

Технология беспроводной связи Wi-Fi используется во всех смартфонах текущего поколения. Больше половины мобильных устройств, выпущенных после 2014 года, поддерживают спецификации 802.11 b/a/g/n/ac.

Спецификация Wi-Fi 5 или 802.11ac позволяет развивать скорость до 6 Гбит/с с использованием нескольких антенн [12] и до 1.2 Гбит/с.

В рамках работы было решено использовать передачу данных по интерфейсу Wi-Fi, поскольку скорость этого способа связи значительно выше. Также, при использовании сетей Wi-Fi не потребуются провода и потенциальные дополнительные конструкции связывания в виде usb-хабов и коммутаторов.

2.3 Техническое задание

2.3.1 Требование к системе

Прототип системы должен состоять из двух приложений – серверного и клиентского.

Система должна уметь функционировать в рамках локальной сети через интерфейс wifi. Также должна обладать надежностью и устойчивостью к выходу из строя узла, или уметь адаптироваться к подключению новых устройств. Должна уметь выполнять вычисления одиночной операции над множеством потоком данных, множество операций над одиночным потоком данных, и множественные операции над множественным потоком данных.

Система должна обладать свойством расширяемости и легкости модернизации. Каждая часть приложения должна иметь плавающую и модульную программную архитектуру, которая допускает последующую масштабируемость системы.

2.3.2 Требование к видам обеспечения

Для проведения тестирования прототипов, потребуется от 3 до 5 мобильных устройств на ОС Android.

Каждое из устройств должно поддерживать root доступ к операционной системе, для установки утилиты debootstrap. С помощью утилиты debootstrap, на устройство, в съемную карту памяти, должна быть установлена подсистема chroot, использующая ядро linux устройства, работающая под управлением системы Debian ARM. На этой подсистеме будет выполняться потенциально распараллеливаемая задача, полученная от сервера.

Аппаратное обеспечение:

Операционная система

- Android 7.0 nougat или выше

Процессор

На 64-разрядной архитектуре ARM Cortex A

Оперативная память

Не меньше 2ГБ

Объём внешней памяти

- **Для клиента:** Наличие SD Card ёмкостью не меньше 16 Гб
- **Для сервера:** Наличие SD Card ёмкостью не меньше 128 Гб

Сеть

Поддержка Wi-Fi стандарта 802.11 b/a/g/ac 5Ггц

2.4 Архитектура системы

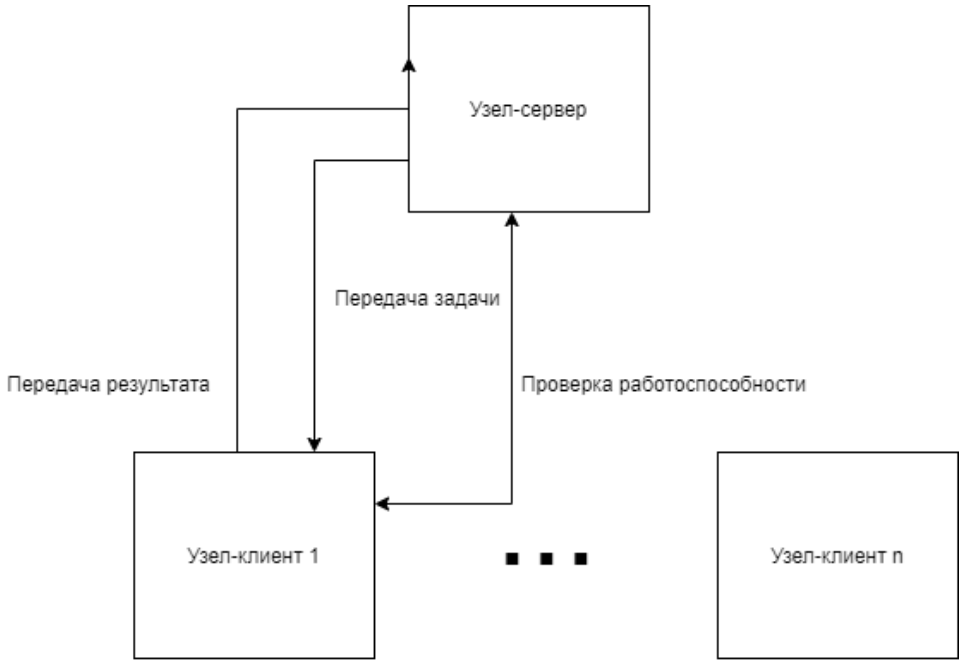

Рисунок 6 – Общая архитектура системы

Система строится на основе клиент-серверной архитектуры. Сервер получает задачи извне, декомпозирует их и распределяет между клиентами, после выполнения клиентами задач, получает результаты и склеивает в правильном порядке, и затем сохраняет на устройстве решеную задачу.

2.5 Самоорганизация

Первое построение системы происходит перед этапом запуска вычислений, во время подключения клиентов: сервер отправляет широковещательное сообщение всем устройствам в локальной сети, содержащее информацию о своём существовании. Клиенты, получив его, могут стать частью системы.

Сервер, асинхронно делает запросы о состоянии каждого из клиентов, чтобы своевременно иметь информацию о том, каким узлам можно отправлять задачи, или, каким образом можно изменить логистику распределения задач между ними в зависимости от условий работы.

Каждый узел-клиент системы, в свою очередь, может находиться в состоянии:

- Готовности
- Выполнения задачи
- Отправки или получения задачи

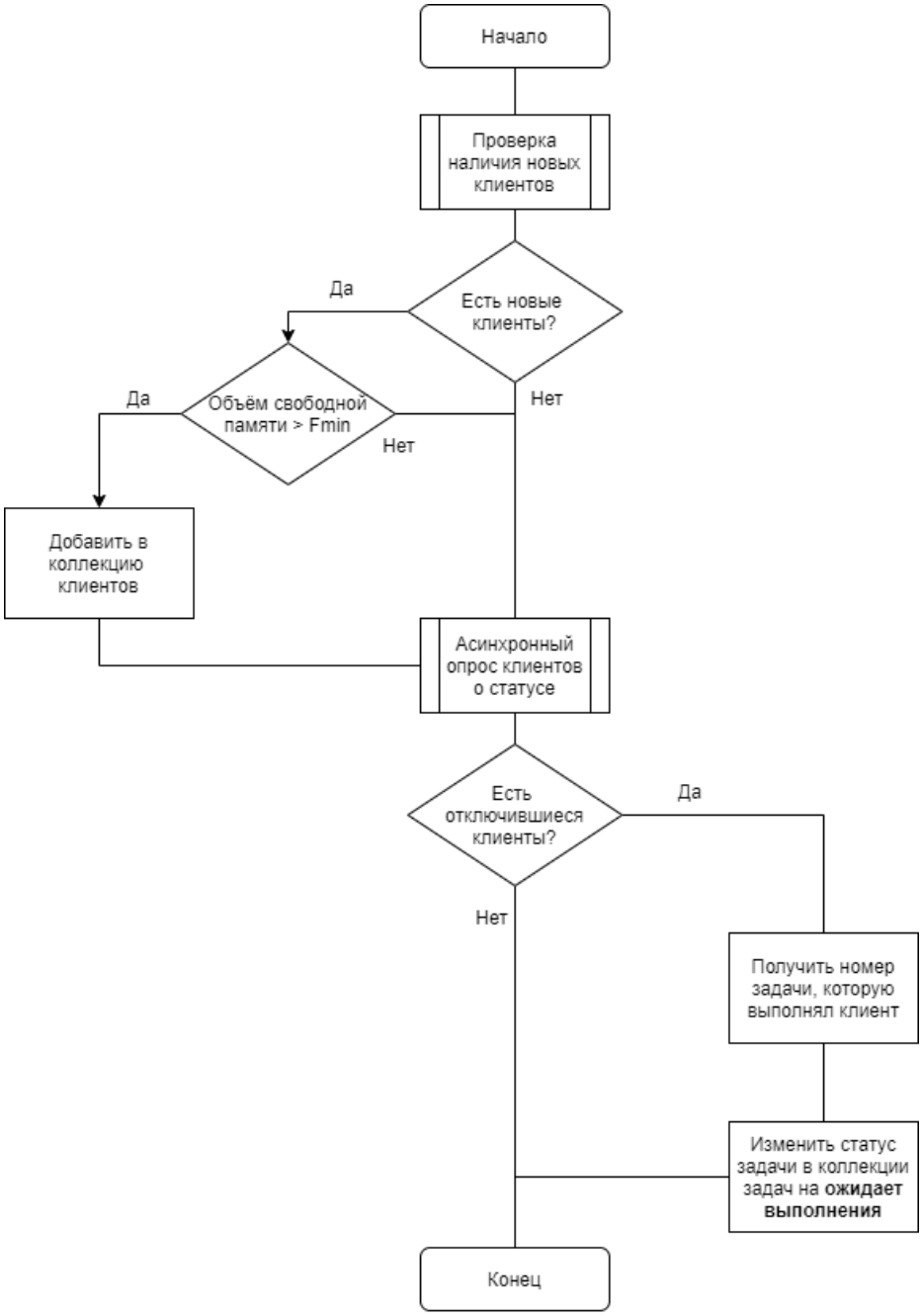

Рисунок 7 –Алгоритм самоорганизации

В процессе вычислений, для поддержания самоорганизации используется алгоритм, предложенный выше.

2.6 Описание работы сервера

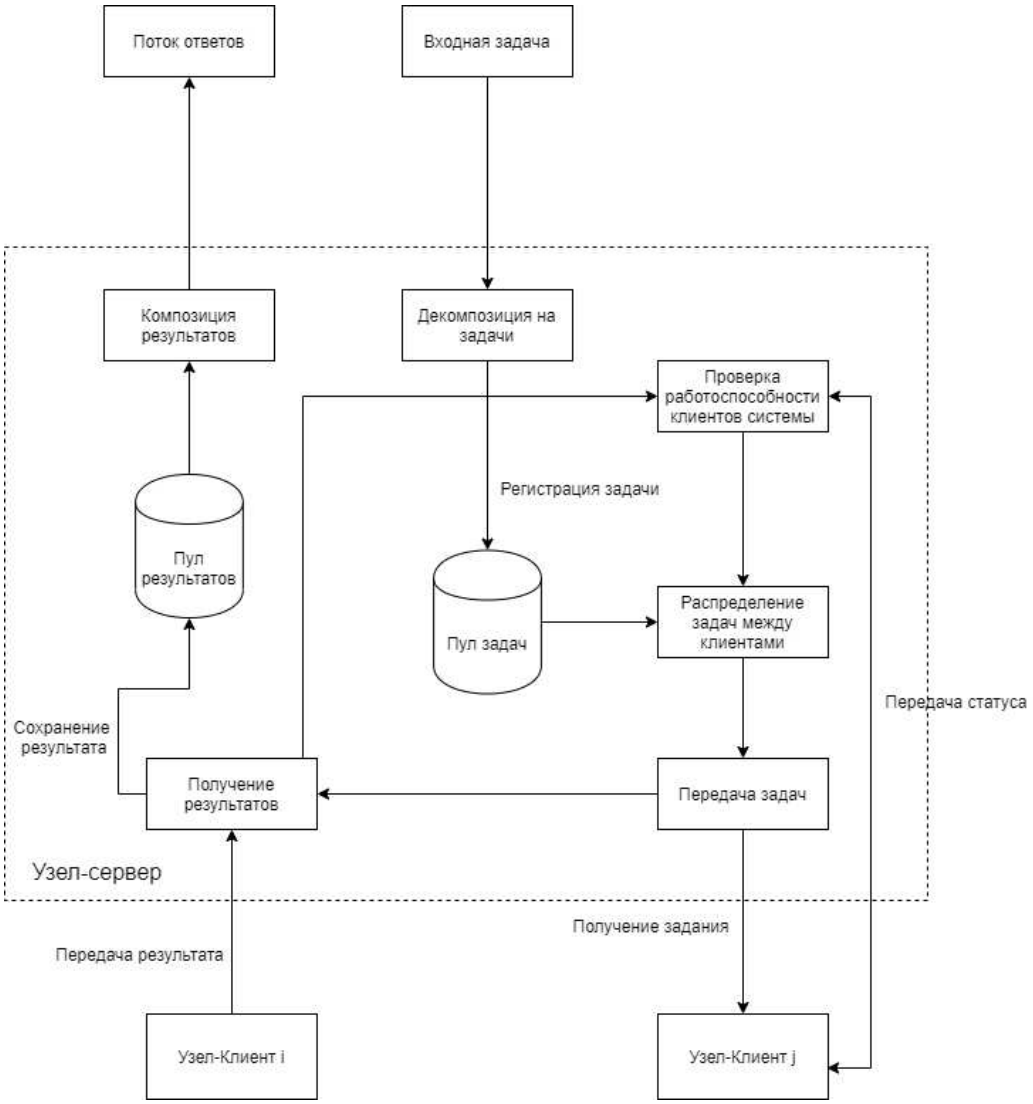

Рисунок 8 – Функциональная схема сервера

Работа сервера представляет из себя работу бесконечного цикла, который является жизненным циклом серверной программы.

Для хранения декомпозированных задач, сервер располагает пулом задач. Аналогичным образом хранятся результаты задач в пуле результатов.

После запуска, серверное приложение получает на вход данные. Перед стартом выполнения задач, сервер передает всем участником локальной сети через специальный протокол, о том, что к нему можно подключиться.

После подключения двух или более клиентов, сервер начинает анализ минимального объёма данных, которого можно отправить каждому клиенту.

Затем, сервер начинает декомпозицию большой задачи на мелкие подзадачи, которые будут переданы клиентам системы.

Когда задачи разбиты и помещены в пул задач, приложение приступает к распределению задач между клиентами. Каждый клиент, находящийся в состоянии готовности, получает по одной задаче.

Чтобы сервер располагал информацией о подключенных клиентах, их состоянии, каждую итерацию жизненного цикла происходит асинхронный опрос клиентов для проверки статуса состояния работы.

Если ответ от клиентов не был получен за установленное время Time to live (далее TTL), то узел-клиент не считается больше частью системы и задачи, переданные ему, возвращаются обратно в пул, и могут быть повторно перераспределены между активными клиентами системы.

2.7 Описание работы клиента

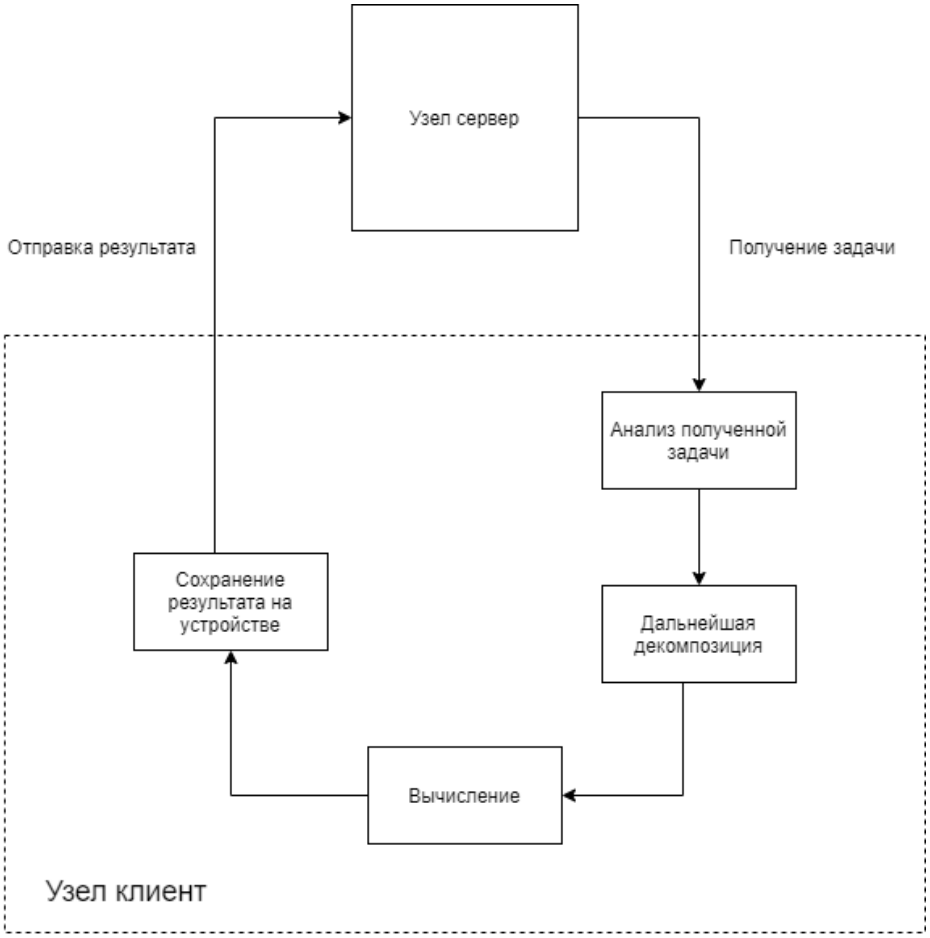

Рисунок 9 – Функциональная схема работы клиента

После получения задач, происходит анализ количества операций и объёма полученных данных.

В случае, если задача сводится к выполнению однородных вычислений, то приложение разделяет задачу на n частей, где n – число ядер устройства. В случае, если задача сводится к выполнению большого количества операций, независимых друг от друга, в таком случае операции разделяются на n.

Когда задача готова к обработке, клиентское приложение вызывает программу, использующую интерфейс OpenMP, и передает ей данные для обработки.

Когда программа обработки заканчивает работу, готовые данные отправляются обратно на сервер, а клиент устанавливает свой статус состояния на готовность принимать новые задачи.

2.7.1 Распределение задач

Для того, чтобы узел-сервер мог отправлять данные, необходимо установить строгий вид структуры данных, хранящих и описывающих целевую задачу, которая будет передаваться между сервером и клиентами.

Все большие задачи, пришедшие на сервер, представляют из себя, как правило, набор данных и набор операций, который требуется разбить на атомарные части, которые могут выполняться в рамках одного клиента-узла.

Устройство должно иметь свободного места не меньше, чем удвоенный размер присланной задачи, чтобы обладать пространством для сохранения результата

Алгоритм декомпозиции задач выглядит следующим образом:

1. Нахождение минимального свободного места среди каждого из устройств (Fmin);
2. Сохранение этого значения на стороне сервера;
3. Подсчет кол-ва задач при текущих условиях по формуле (1);

$$N = V / (Fmin * 2) \quad (1)$$

4. Создание структуры, описывающую информацию о задачах, которая имеет вид:
   - Порядковый номер (Идентификатор)
   - Ссылку на начало
   - Ссылку на конец
   - Набор операций, которые требуется произвести

Начало асинхронной передачи данных от сервера клиентам.

В результате окончания вычислительной работы клиента должна сформироваться структура результата, которая будет послана на сервер. Она представляет из себя набор данных и номер задачи.

После того, как все задачи вместе с данными вернулись на сервер, пул задач сортируется в соответствии с номерами результатов, и, в последствии, склеивается в готовое решение.

2.8 Выводы к главе

По итогам проведенного анализа средств разработки, было принято решение о разработке клиент-серверного приложения-прототипа на основе языка java и C++, под android версии не ниже 7.0 для мобильного устройства, у которого также имеется доступ root для установки образа Debian.

Создаваемая система будет работать практически без участия пользователя, кроме случая, когда пользователю надо будет выбрать задачу и отправить её на сервер для дальнейшей обработки. Однако, приложения для клиента и сервера будут отображать всю информацию о статусе выполнения задач.

В данной главе также были описаны технические требования и архитектура системы в целом для клиента и сервера.

3 ТЕХНИЧЕСКАЯ РЕАЛИЗАЦИЯ ПРОТОТИПА

3.1 Разработка серверного приложения

В результате разработки сервера прототипа получилось приложение с шестью классами, каждый из которых участвует в выполнении только одной задачи, а также трёх коллекций для хранения метаданных о задачах, результатах и подключенных клиентах соответственно.

Получившаяся диаграмма классов представлена на рисунке 9.

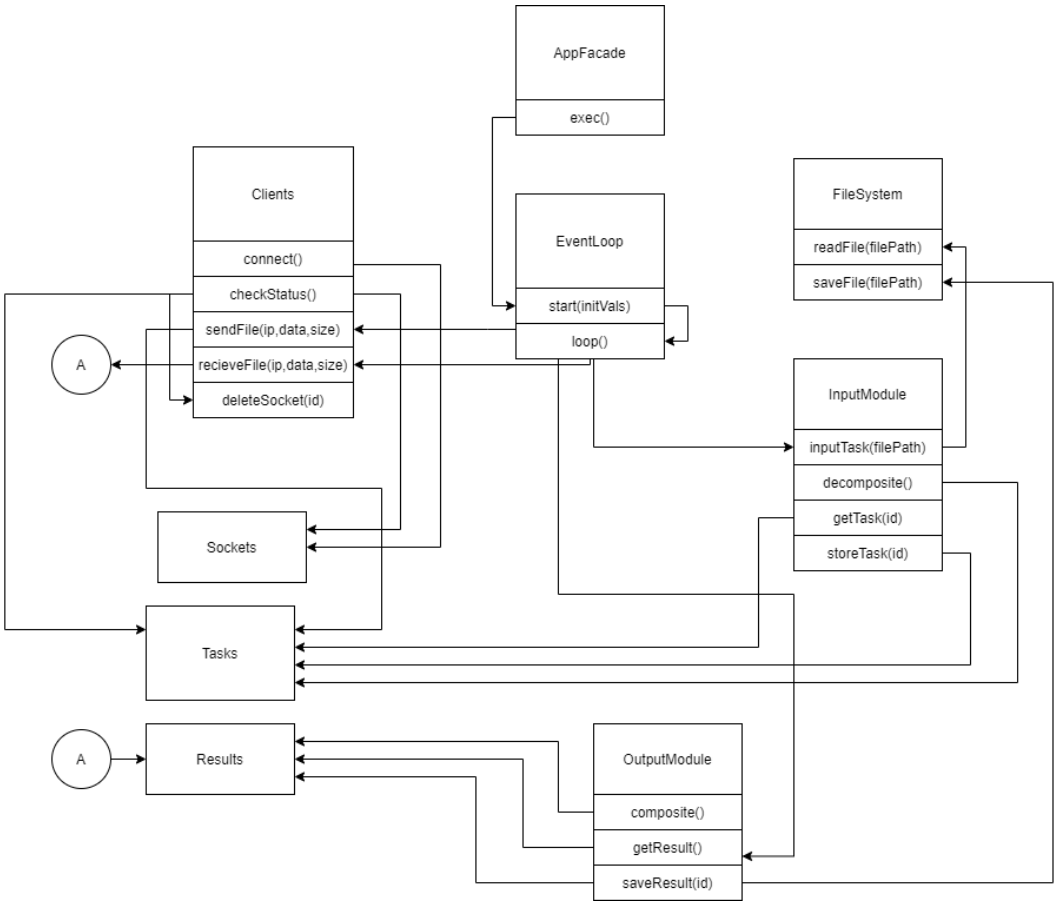

Рисунок 10 – Диаграмма классов серверного приложения

Класс загрузки AppFacade занимается инициализацией и запуском всей системы со стороны сервера. В нём реализован один метод – exec, который создает, настраивает и запускает экземпляр класса EventLoop.

Класс EventLoop обслуживает работу выполнения событийного цикла. Перед запуском выполняется метод start, который инициализирует всех

подключенных клиентов, создает объекты-сокеты и сохраняет их в коллекцию sockets.

Класс Clients отражает слой работы с подключением, передачей и получением информации на низком уровне, между клиентами, и следит за их статусами с помощью метода checkStatus, который вызывается асинхронно. В случае, если checkStatus вернул статус отключен, происходит пометка задачи как невыполненная в коллекции Tasks, а также удаление сокета из списка.

Классы InputModule и OutputModule являются обёртками для композиции, декомпозиции, поддерживания и обновления информации о предстоящих и выполненных задачах.

Класс FileSystem абстрагирован от хранилища данных, и работает с файловой системой устройства на низком уровне, предоставляя другим классам два высокоуровневых метода saveFile и loadFile, для сохранения и загрузки соответственно.

Коллекция Tasks хранит задачи в произвольном порядке, каждый элемент хранит номер задачи, и пути до её физического адреса начала данных и набора операций, и их конца на устройстве, статус задачи – готова к выполнению или выполняется, а также обнуляемое поле - номер сокета клиента, который её выполняет.

После выполнения задачи, запись уходит из коллекции Tasks и добавляется в коллекцию results.

Коллекция Results хранит полученные решёные задачи в виде, подобном коллекции tasks.

3.2 Разработка интерфейса серверного приложения

Серверное приложение состоит из нескольких экранов:

1. Экран выбора файла задачи;

2. Экран списка подключённых клиентов;

3. Экран вывода статистики выполнения задач, подключенных клиентов и их состояний;

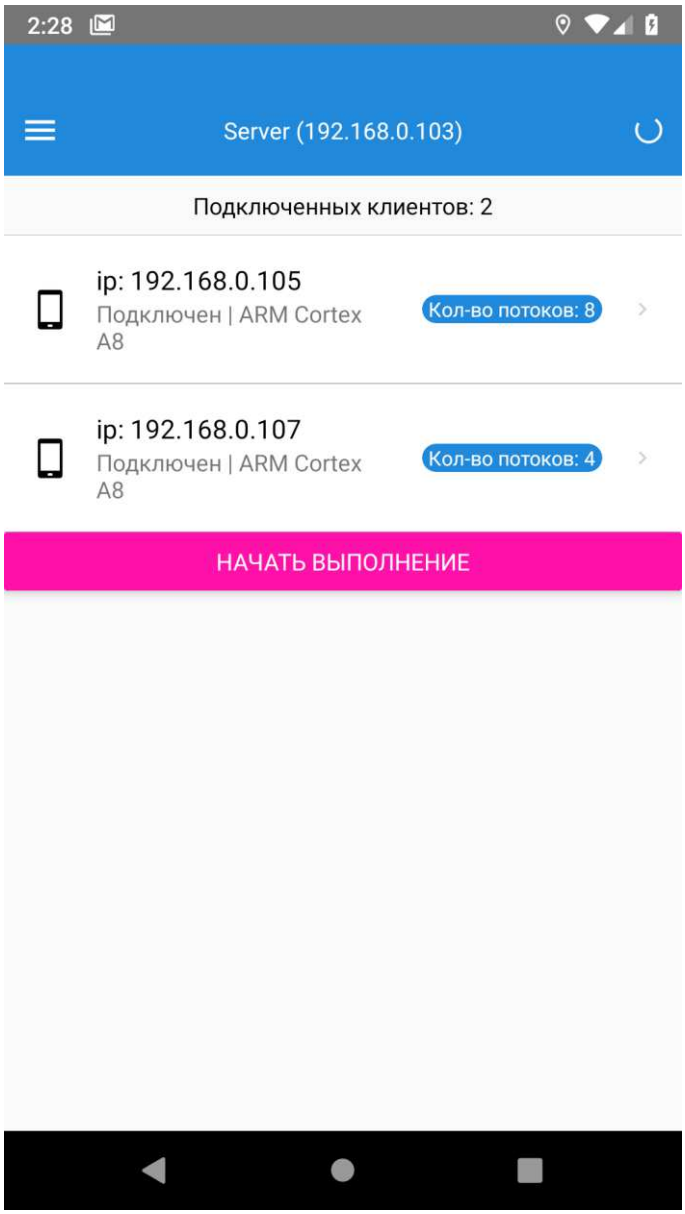

Рисунок 11 – Экран списка подключенных клиентов

Экран списка подключенных клиентов позволяет узнать информацию о количестве ядер, архитектуре процессора, количестве свободного места на каждом из устройств.

Непосредственно отправка задач, и получение результатов, отражение всей информации о фактическом прогрессе выполнения отображается на экране 3.

3.3 Разработка клиентского приложения

В результате разработки клиента, получилось приложение с пятью классами, один из которых инициализирует системный вызова для запуска программы вычислений.

Диаграмма классов представлена на рисунке 10.

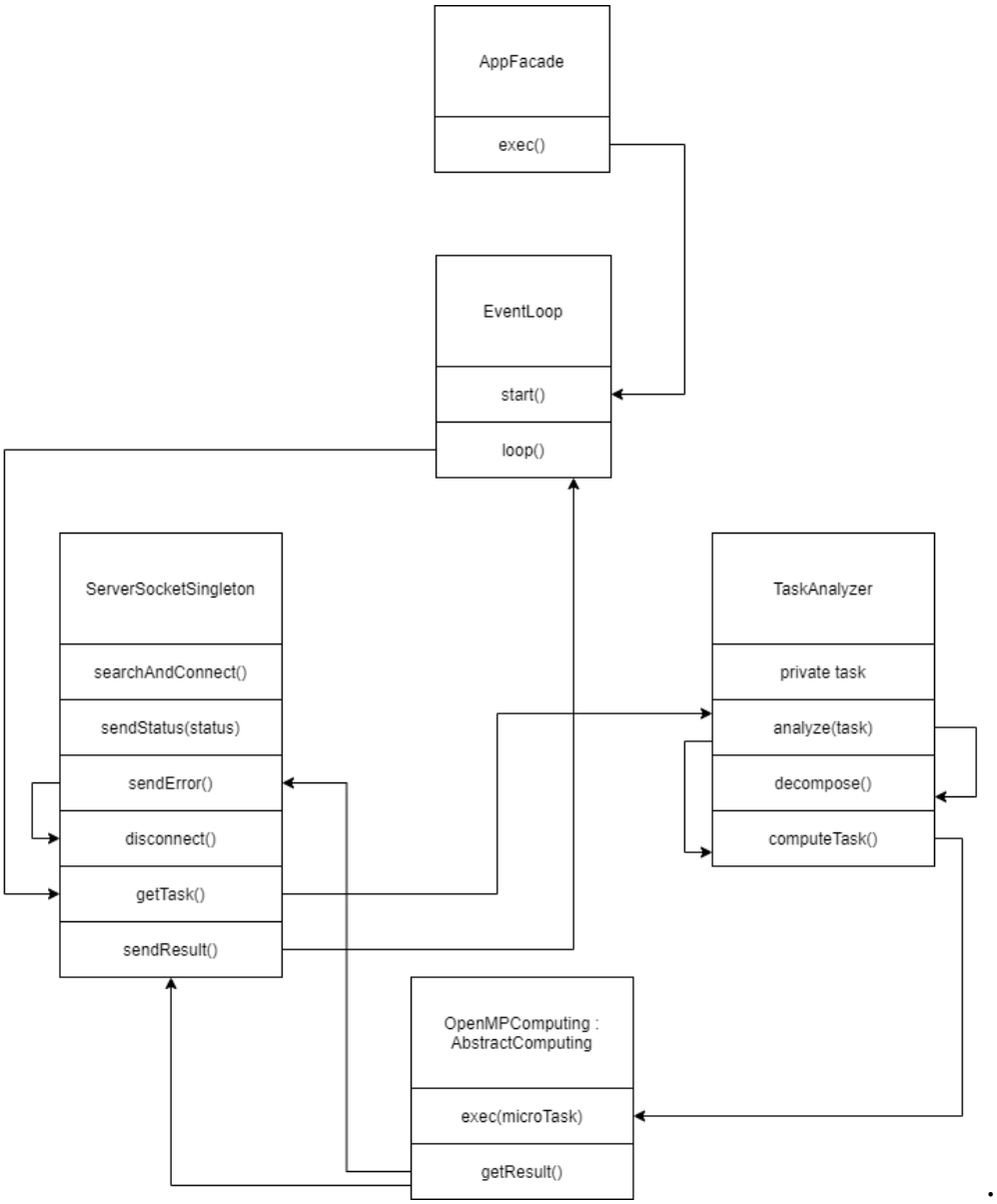

Рисунок 12 – Диаграмма классов клиентского приложения

Классы AppFacade работают аналогичным образом, как и в серверном приложении. Метод start инициализирует цикл получения, вычисления и отправки результата.

Класс TaskAnalyzer производит логистику по распараллеливанию задачи на устройстве, декомпозицию данных на потоки и инициализирует запуск программы для вычислений.

Класс ServerSocketSingleton является API для работы с получением задачи и передачей результата. После запуска метода start класса Eventloop, автоматически запускается метод sendStatus, который ожидает запроса сервера, и, после поступления, отправляет о текущем статусе клиента.

Интерфейс или абстрактный класс AbstractComputing описывает общий вид взаимодействия клиента с приложением для решения полученной задачи. Абстрактный класс позволяет также описывать множество реализаций для того, с какой программой должно взаимодействовать клиентское приложение для решения поставленной задачи. Это отвечает особенности масштабирования системы в целом.

Класс OpenMPComputing - реализация интерфейса AbstractComputing для программы, установленной в клиент и использующей библиотеку OpenMP для выполнения вычислений. Метод exec этого класса передаёт поток управления приложению, которое выполняет вычисления. В результате, приложение отправляет полученный результат обратно через метод sendResult.

В случае, если в результате вычислений произошла ошибка, информация о ней передается на сервер через метод sendError.

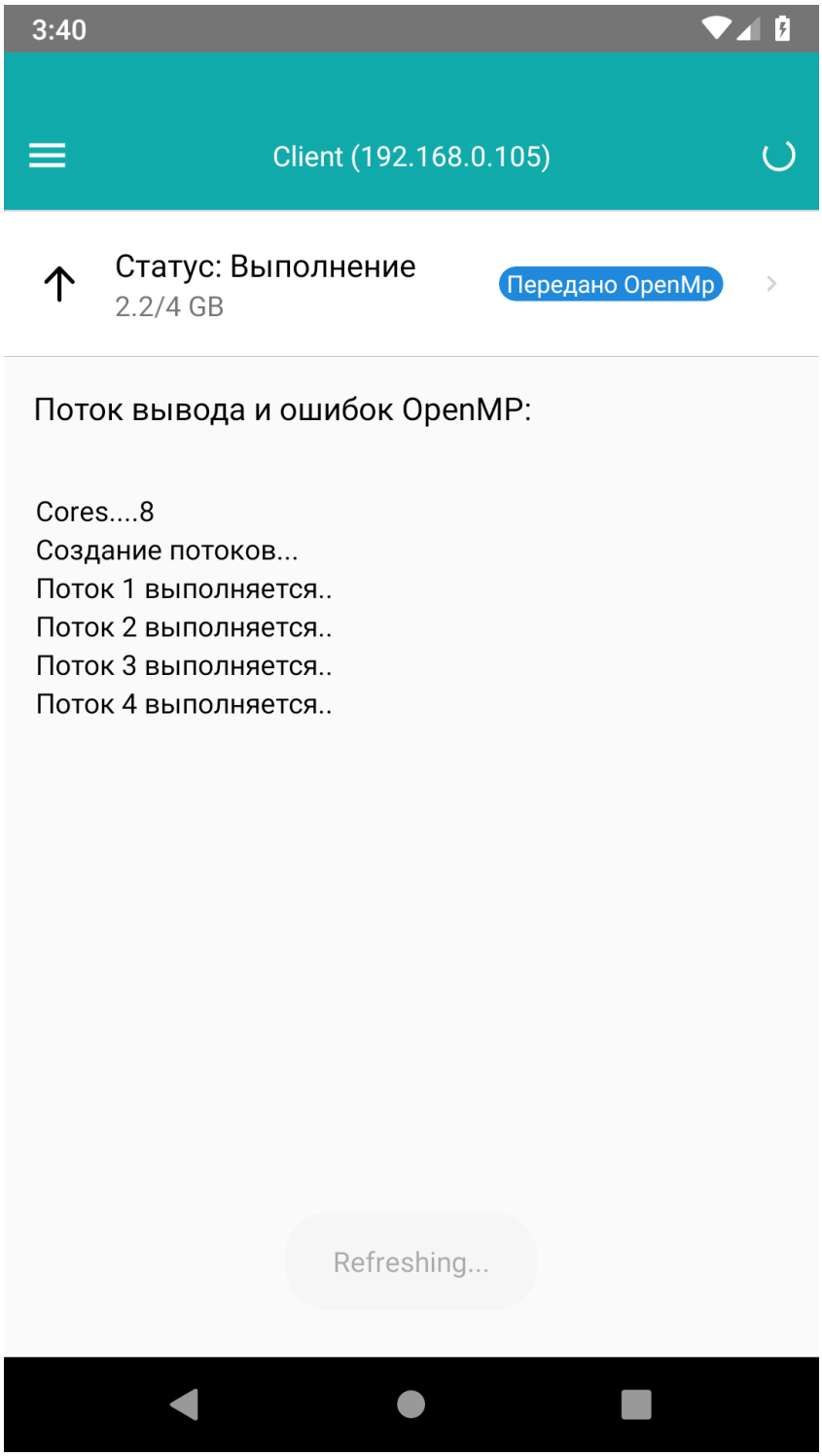

Рисунок 13 – Экран выполнения задачи

Клиентское приложение состоит только из двух экранов:

1. Экран выбора сервера;
2. Экран выполнения задачи.

3.4 Выводы к главе

Результатом данной главы является представление описания реализации серверной и клиентской части прототипа системы с учетом требований описанных во второй главе работы.

В результате запуска и отработки полученной системы была подтверждена адекватность разработанного подхода вычислений и управления задачами, а также соответствие с ожиданиями работы программ.

В частности, удалось достичь надежной работы системы в условиях с переменным количеством клиентов.

ЗАКЛЮЧЕНИЕ

В ходе диссертационной работы проведен анализ предметной области результатом которого стал вывод об актуальности выбранного направления разработки, выявление достоинств и недостатков уже имеющихся систем, показаны перспективы развития данного направления.

В соответствии с полученным заданием на выпускную квалификационную работу предложен и разработан метод управления задачами в самоорганизующихся распределенных вычислительных системах.

Для проверки работоспособности предложенного метода управления, был разработан клиент-серверный прототип системы.

Архитектура разработанной системы обладает свойством расширяемости и легкости модернизации. Однако, в качестве примера приведена всего одна реализация программы для решения ресурсоёмких задач на библиотеке OpenMP.

Результат работы соответствует заданию на ВКР. Дальнейшее развитие работы предполагает проверка возможности использования контейнеров для решения задач, которые позволят вмещать в себя помимо задачи окружение, чтобы позволить выполнять клиентам полноценные прикладные задачи со всеми зависимостями, необходимыми для решения.

СПИСОК СОКРАЩЕНИЙ

ОС – Операционная система

MPI - Message passing interface

ART – Android Runtime

ПК – Персональный компьютер

TTL – Time to live

HPC – High performance computing

ARM – Advanced RISC Machine

RISC - Reduced instruction set computer

API – Application programming interface

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 Блог программиста: электронный ресурс / URL: https://pro-prof.com/forums/topic/актуальность-параллельного-программ

2 Электронная библиотека ИТМО : электронный ресурс / URL : http://neerc.ifmo.ru/wiki/index.php?title=Распределенные_вычислительные_системы

3 Сравнение стандартов 802.11 ac и 802.11 n : электронный ресурс / URL : https://www.ruckuswireless.com/ru/rucktionary/802.11

4 CPU Comparison x86 vs ARM : электронный ресурс / URL : https://fossbytes.com/cpu-comparison-x86-arm-cpu-benchmark/

5 Распределенное вычисление: Краткое введение в проекта : электронный ресурс / URL : https://habr.com/ru/post/390749/

6 Клиент BOINC на Google Play : электронный ресурс / URL : https://play.google.com/store/apps/details?id=edu.berkeley.boinc&hl=en_US

7 Настройка вычислительного кластера на базе Torque : электронный ресурс / URL : https://www.opennet.ru/tips/2496_cluster_torque_maui.shtml

8 There are now 2.5 billion active Android devices : электронный ресурс / URL : https://www.theverge.com/2019/5/7/18528297/google-io-2019-android-devices-play-store-total-number-statistic-keynote

9 Официальный сайт ОС android : электронный ресурс / URL : https://www.android.com/intl/ru_ru/

10 Статья leave your smartphone on overnight to help find ways to fight Covid-19 : электронный ресурс / URL : https://www.standard.co.uk/tech/coronavirus-research-app-vodafone-dreamlab-a4409806.html

11 Статья top distributed computing apps on Android : электронный ресурс / URL : https://ausdroid.net/2016/07/20/will-spend-spare-cpu-cycles-top-5-distributed-computing-apps-android/

12 Стандарт IEEE 802.11ac : электронный ресурс / URL : http://www.tadviser.ru/index.php/%D0%A1%D1%82%D0%B0%D1%82%D1%8C%D1%8F:IEEE_802.11ac

13 Высокоскоростной интерфейс USB 3.0 : электронный ресурс / URL : https://compress.ru/article.aspx?id=19961

14 Статья Google is adding Kotlin as an official programming language for Android development : электронный ресурс / URL : https://www.theverge.com/2017/5/17/15654988/google-jet-brains-kotlin-programming-language-android-development-io-2017

15 Официальный сайт языка kotlin : электронный ресурс / URL : https://kotlinlang.org/

16 Сравнение использования технологий параллельного программирования Microsoft применительно к задаче поиска данных MapReduce: электронный ресурс / URL: http://omega.sp.susu.ru/books/conference/PaVT2010/talks/Fedyukovich.pdf

17 ARM как основа для новых суперкомпьютеров: электронный ресурс / URL: https://habr.com/ru/post/181011/

18 В.П. Никольский, В.В. Стегайлов Эффективность процессоров ARM для расчетов классической молекулярной динамики // Суперкомпьютерные дни в России 2015 // — 2015. — С.614-622

19 Mitra G., Johnston B., Rendell A.P., McCreath E., Zhou J. Use of SIMD Vector Operations to Accelerate Application Code Performance on Low-Powered ARM and Intel Platforms // 2013 IEEE 27th International Parallel and Distributed Processing Symposium Workshops & PhD Forum (IPDPSW). May 2013. P. 1107–1116.

20 Официальный сайт проекта Mont-blanc: электронный ресурс / URL: https://www.montblanc-project.eu/project

21 Учебник по OpenMP: электронный ресурс / URL: https://pro-prof.com/archives/4335

22 MPI. Вводный курс: электронный ресурс / URL: http://www.ssd.sscc.ru/old/old/kraeva/MPI.html

23 Официальная документация по JVM: электронный ресурс / URL: https://docs.oracle.com/javase/specs/jvms/se8/html/

Федеральное государственное автономное
образовательное учреждение
высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт космических и информационных технологий
институт

Вычислительная техника
кафедра

УТВЕРЖДАЮ
Заведующий кафедрой

_____ О. В. Непомнящий
подпись инициалы, фамилия
«____» _____ 2020 г.

МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ

Способы управления задачами в распределенных децентрализованных мобильных системах
Тема

09.04.01 «Информатика и вычислительная техника»
код и наименование направления

09.04.01.06 «Микропроцессорные системы»
код и наименование магистерской программы

Руководитель _____ зав. каф. ВПВ, канд.техн.наук, доцент — Д.А. Кузьмин
подпись, дата — инициалы, фамилия

Выпускник _____ — И.В. Якимов
подпись, дата — инициалы, фамилия

Нормоконтролер _____ — Д.А. Кузьмин
подпись, дата — инициалы, фамилия

Рецензент _____ Директор по проектам ЗАО "КРИС" — Е.Ю. Белоголовкин
подпись, дата — инициалы, фамилия

Красноярск 2020