**Федеральное государственное автономное образовательное учреждение   
высшего образования**

**КАЗАНСКИЙ (ПРИВОЛЖСКИЙ) ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ**

**ВЫСШАЯ ШКОЛА ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ И   
ИНФОРМАЦИОННЫХ СИСТЕМ**

Направление подготовки: 09.03.03 Прикладная информатика

Профиль: Прикладная информатика в экономике

**ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА**

**Разработка клиент-серверного фреймворка для высоконагруженных вычислений на базе Linux и MPI**

**Работа завершена:**

«\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_2017 г.

Студент группы 11-308 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_И.А. Габидуллин

**Работа допущена к защите:**

Научный руководитель

«\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_2017 г. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_А.С. Тощев

Директор Высшей школы ИТИС

«\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_2017 г. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ А.Ф. Хасьянов

Казань – 2017 г.

Содержание

[Введение 4](#_Toc482389964)

[1. Обзорно-аналитическая часть 6](#_Toc482389965)

[1.1 Принципы параллельных вычислений 6](#_Toc482389966)

[1.2 Описание фреймворка 6](#_Toc482389967)

[2. Технологическая часть 7](#_Toc482389968)

[2.1 Язык программирования 7](#_Toc482389969)

[2.2 Средства разработки 8](#_Toc482389970)

[2.3 Средства проектирования 9](#_Toc482389971)

[2.4 Разработка серверной части 9](#_Toc482389972)

[2.5 Разработка клиентской части 10](#_Toc482389973)

[3. Проектирование и разработка фреймворка 12](#_Toc482389974)

[3.1 Проектирование фреймворка 12](#_Toc482389975)

[3.1.1 Общая структура проекта 12](#_Toc482389976)

[3.1.2 Диаграмма компонентов 12](#_Toc482389977)

[3.1.3 Диаграмма развёртывания 12](#_Toc482389978)

[3.2 Разработка серверной части 12](#_Toc482389979)

[3.2.1 Разработка MPI сервиса 12](#_Toc482389980)

[3.2.2 Разработка сервера управления 12](#_Toc482389981)

[3.3 Разработка клиентской части 12](#_Toc482389982)

[3.3.1 Разработка консольного приложения 12](#_Toc482389983)

[3.4 Взаимодействие клиента с сервером 12](#_Toc482389984)

[Заключение 13](#_Toc482389985)

[Список использованных источников 14](#_Toc482389986)

[Приложение 15](#_Toc482389987)

# Введение

В мире существует множество задач, которые можно разделить на мелкие подзадачи и решать их параллельно. Особенно часто это встречается в науке. Разнообразные математические преобразования и вычисления. И в мире, в котором даже на мобильных телефонах по несколько процессоров, необходимо использовать возможности параллельных вычислений.

Параллельные вычисления уже давно стали важной частью Computer Science и активно используются на практике. Во многих научных организациях, например, в Казанском Федеральном Университете существует кластер электронно-вычислительных машин для выполнения параллельных вычислений на базе технологии MPI. Но не существует простого и удобного способа управлять вычислениями: разворачивать программный код, запускать, отслеживать, получать результаты или данные о скорости работы. Сейчас всё это выполняется прямо на кластере, пользователь подключается к кластеру, загружает нужные ему файлы и запускает процесс.

Исходя из вышеизложенного, было принято решение создать клиент-серверный фреймворк для работы с параллельными вычислениями. Данный фреймворк должен позволять гибко настраивать взаимодействия с сервером параллельных вычислений. Единожды развернув и настроив фреймворк пользователь получает удобный и гибкий способ управления своими вычислениями. Пользователь получает возможность организовать работу извне, не подключаясь напрямую к высокопроизводительному кластеру. Таким образом, данная работа несет в себе практическую пользу в научной сфере.

Фреймворк базово должен поддерживать такие вещи как:

1. загрузка программы на сервер;
2. выполнение программы;
3. получение статистики;
4. возможность управлять ходом работы программы;
5. возможность легко дополняться и изменяться.

Данная дипломная работа посвящается разработке клиент-серверного фреймворка для управления параллельными MPI задачами.

Для достижения цели дипломной работы необходимо было выполнить следующие задачи:

1. проектирование архитектуры фреймворка, клиентской и серверной части;
2. проектирование архитектуры взаимодействия клиентской и серверной части;
3. изучение и применение найденных способов управления процессами в операционной системе;
4. разработка серверной части фреймворка;
5. разработка клиенсткой части фреймворка.

# Обзорно-аналитическая часть

В мире существует большое количество высокопроизводительных кластеров, и очевидно, что в каждом случае существует своя система управления ими. Невозможно найти и изучить все эти системы, но можно попробовать создать достаточно фреймворк, на базе которого возможно создать практически любую систему управления вычислениями.

## Принципы параллельных вычислений

В 1966 году профессором Стэндфордского университета Майклом Финном была предложена, так называемая «Таксономия Флинна» (Flynn's taxonomy). В ней электронно-вычислительные машины были разделены по принципу параллелизма в потоках команд и потоках данных. Рассмотрим её:

* ОКОД — Вычислительная система с одиночным потоком команд и одиночным потоком данных (SISD, Single Instruction stream over a Single Data stream);
* ОКМД — Вычислительная система с одиночным потоком команд и множественным потоком данных (SIMD, Single Instruction, Multiple Data);
* МКОД — Вычислительная система со множественным потоком команд и одиночным потоком данных (MISD, Multiple Instruction Single Data);
* МКМД — Вычислительная система со множественным потоком команд и множественным потоком данных (MIMD, Multiple Instruction Multiple Data).

[ссылка <https://en.wikipedia.org/wiki/Flynn%27s_taxonomy>]

## Технологии параллельных вычислений

## Описание фреймворка

# Технологическая часть

Важной частью в разработке какого-либо проекта является выбор инструментов и технологий, которые будут использоваться.

Выбирать следует исходя из целей и задач, так как неверно выбранная технологическая база в дальнейшем может помешать разработке. Или же в худшем случае придется начать работу сначала, используя уже подходящие технологии.

В сфере разработки программного обеспечения следует выделять следующие аспекты, в которых нужно выбирать те или иные инструменты:

1. Язык программирования и программная платформа (фреймворк);
2. Средства разработки;
3. Средства проектирования.

В данной главе будет рассмотрен выбор тех или иных инструментов и технологий для каждого аспекта разработки и проектирования, которые будут использоваться на данном проекте.

## Язык программирования

В качестве языка разработки был выбран C# 7-ой версии. Данный язык программирования подходит для разработки разнообразных видов приложений: приложений для мобильной платформы, веб приложений, бизнес приложений, а также высоконагруженных систем. По этой причине данный язык как нельзя подходит для данного проекта, так как в нём сочетается разные виды приложений. Используя данный язык можно написать, как клиентскую часть, так и серверную.

Также данный язык выбран из-за того, что он достаточно популярный и распространенный. Разработчиков, которые знакомы с ним и работают используя его очень большое количество и найти таких специалистов не составляет труда. По индексу TIOBE (индекс, оценивающий популярность языков программирования) язык программирования C# на май 2017 года занимает 5 место по популярности (ссылка наhttps://www.tiobe.com/tiobe-index/).

В каждой новой версии C# добавляются новые возможности для разработчика и становится всё больше способов сделать программный код более структурированным и читабельным. Например, в последних версиях языка добавились такие возможности как интерполяция строк и локальные функции.

Язык C# поддерживается несколькими фреймворками, в дальнейшем будет описано какие именно фреймворки были выбраны для клиентской и серверной части проекта.

## Средства разработки

Для того, чтобы воспользоваться самыми новыми возможностями языка программирования, следует воспользоваться самой новой средой разработки. В данном случае выбор пал на Visual Studio 2017 Community.

Данная среда позволяет быстро и качественно писать код, так как в ней существует большое количество подсказок, которые могут помочь разработчику быстрее и качественнее создавать программный код. В ней удобно следить за исполнением кода. Visual Studio предоставляет множество данных о производительности создаваемой программы, что очень важно в разработке. Также данная среда поддерживает большое количество языков программирования, а значит программисту не надо осваивать большое количество сред разработки, а достаточно лишь изучить одну.

При помощи Visual Studio можно легко собирать проекты, а также разворачивать, либо настраивать развёртывание при необходимости.

Для тестирования веб-сервера использовался плагин для веб-браузера Opera – RestMan. Благодаря интуитивному и удобному интерфейсу, плагином достаточно легко научиться пользоваться.

Данный плагин позволяет отправлять REST веб запросы любого вида к указанному веб серверу и получать ответы. Ответы отображаются в удобном для анализа виде.

## Средства проектирования

Проектирование – это очень важная часть разработки. Надо иметь способ легко создавать нужные для проекта диаграммы. Средой проектирования была выбрана компьютерную программу Visual Paradigm Community Edition.

Данная программа является бесплатной для некоммерческого использования. Программа позволяет создавать разнообразного вида UML диаграмм и сохранять их в одном проекте, что очень удобно.

К недостаткам можно отнести не совсем очевидный пользовательский интерфейс.

## Разработка серверной части

Так как языком разработки был выбран C#, то выбор фреймворка ограничивается разными видами .NET Framework. В свою очередь .NET Framework имеет множество версий, между которыми можно выбирать:

1. .NET Framework – классический Windows фреймворк. Является старейшей платформой среди .NET платформ. Поддерживает большое количество типов приложений (Windows приложение, веб-приложение, мобильное приложение). Также платформа имеет большие возможности взаимодействия с системой. Но данная платформа поддерживает лишь семейство операционных систем Microsoft Windows. Поэтому данный фреймворк не подходит для данного проекта;
2. Mono – некоммерческий кроссплатформенный (Linux, MacOS, Windows) фреймворк. Является перенесенной на другие операционные системы .NET Framework. Но с некоторыми недоработками;
3. .NET Core – является разработанной компанией Microsoft кроссплатформенным фреймворком. В данной платформе используются наработки Mono. Данная платформа активно развивается и добавляет всё большее количество функций из классической .NET Framework, но для большего количества платформ.

Таким образом, стоит выбор между .NET Core и Mono. Был выбран .NET Core. Данная платформа поддерживает все нужные для проекта возможности, позволяет создать кроссплатформенный код. Также с данной платформой легче начать работу, так как Visual Studio 2017 сразу после установки поддерживает разработку для этой платформы.

## Разработка клиентской части

Предполагается то, что клиентская часть можно создать, используя любую платформу, так как взаимодействие с серверной частью происходит через протокол HTTP (Web API).

В данной работе будет создано консольное приложение, которое взаимодействует с сервером по протоколу HTTP. Разработка будет происходить на платформе .NET Core для кроссплатформенности.

Предполагается, что, пользователь данного проекта может с лёгкостью добавить свои новые функции или создать абсолютно новый клиент. Например, сайт, который будет выполнять роль клиента. Благодаря выбранному протоколу взаимодействия и открытому исходному коду серверной части, можно такой клиент легко создать.

# Проектирование и разработка фреймворка

## Проектирование фреймворка

### Общая структура проекта

В основу архитектуры проекта положена модель слоёв, описанная Мартином Фаулером в книге «Архитектура корпоративных программных приложений» [ссылка]. Архитектура заключается в том, что проект разбивается на несколько независимых слоёв (layer). Таким образом достигается гибкая для модернизации и изменений структура, что в случае с фреймворком очень важно.

### Диаграмма компонентов

### Диаграмма развёртывания

## Разработка серверной части

### Разработка MPI сервиса

### Разработка сервера управления

## Разработка клиентской части

### Разработка консольного приложения

## 3.4 Взаимодействие клиента с сервером

# Заключение

# Список использованных источников

# Приложение