Санкт-Петербургский государственный университет

**Кафедра технологии программирования**

**Широбоков Михаил Владимирович**

**Магистерская диссертация**

**Реализация высокопроизводительных растровых вычислений для Web ГИС**

Направление 02.04.02

Фундаментальная информатика и информационные технологии

Научный руководитель,  
канд. техн. наук,  
доцент  
Блеканов И. С.

Санкт-Петербург

2018

# Содержание

[Содержание 1](#_Toc506746365)

[1. Постановка задачи 2](#_Toc506746366)

[2. Обзор методов выполнения растровой кластеризации и классификации 3](#_Toc506746367)

[2.1 Масштабирование системы 3](#_Toc506746368)

[Hadoop 3](#_Toc506746369)

[Spark 4](#_Toc506746370)

[Akka 4](#_Toc506746371)

[2.2 Варианты выбора СУБД 5](#_Toc506746372)

[MySQL 5](#_Toc506746373)

[PostGIS 6](#_Toc506746374)

[MongoDB 6](#_Toc506746375)

[SpatiaLite 7](#_Toc506746376)

[CouchDB 8](#_Toc506746377)

[2.3 Распределение данных по кластеру 8](#_Toc506746378)

[2.4 Параллельная обработка растровых данных внутри одной машины 9](#_Toc506746379)

[Advanced Vector Extensions 9](#_Toc506746380)

[Nvidia CUDA 10](#_Toc506746381)

[OpenCL 10](#_Toc506746382)

[Список литературы 12](#_Toc506746383)

# Введение

Различные геоинформационные системы уже давно стали неотъемлемой частью современного информационного мира. Они используются в совершенно различных областях прикладных теоретических и, главным образом, прикладных наук, находя себе применение в таких дисциплинах, как урбанистика, экология, биология и экономика. Их история берет начало еще в 80-х годах прошлого века, когда были предприняты первые успешные попытки эколого- и биогеографического моделирования для исследования и выявления разнообразных закономерностей в распределении видов животных и его зависимости от различных природно-экологических факторов. Впоследствии подобные системы стали использоваться очень широко, и в настоящее время технологии растрового геоинформационного моделирования и анализа применяются в совершенно не связанных между собой областях, начиная от прогнозирования ареалов распространения биологических объектов и метеорологии и заканчивая анализом автомобильного трафика в мегаполисах.

Все сильнее ускоряющееся развитие цифрового мира неизбежно влечет за собой увеличение общего объема данных, подлежащих обработке. Справиться с их анализом силами одного компьютера или даже целого кластера часто бывает сложно, а иногда и невозможно по финансовым или техническим причинам. Это явление не обошло стороной и рынок географических информационных систем, нагрузка которых часто включает в себя работу не только с относительно легковесными векторными объектами, но и, зачастую, с космическими снимками поверхности, могущими иметь очень высокие разрешения. В настоящее время существуют космические аппараты, способные получать информацию об объектах, отстоящих друг от друга на расстоянии в 31 сантиметр, что дает примерно 10 пикселей на квадратный метр земной поверхности, и этот показатель постоянно увеличивается.

На данный момент существует достаточно много программных продуктов, так или иначе взаимодействующих с этой стороной географических данных. Среди них есть как и известные всем продукты массового потребления, такие как Google Maps и Яндекс.Карты, так и менее известные во многом благодаря своей специфике сервисы, позволяющие применять подход геоинформационных систем, например, к проблемам урбанистики и экономики. Одним из таких проектов является O-GIS -- <описание системы и перечисление её плюсов и отличий от существующих аналогов>.

Но, как и любой проект, находящийся в стадии разработки, O-GIS имеет нерешенные проблемы, одной из которых является невысокая скорость выполнения операций растровой алгебры, включающих в себя как различные прямые пользовательские запросы, так и проективные преобразования. В рамках данной работы предпринята попытка переработки вычислительного модуля системы, включающего в себя как общие методы реализации высокопроизводительных систем, так и специфичные, могущие найти свое применение непосредственно в рамках обозначенной системы.

# Постановка задачи

Опорными моментами, определяющими производительность выполнения операции над растрами в Географической Информационной Системе, являются:

* возможность горизонтального и вертикального масштабирования Географической Информационной Системы;
* логическая организация хранения данных внутри СУБД, управляющей политикой доступа к структурированной и неструктурированной части;
* производительность операций ввода-вывода данных из базы и преобразования сырых данных в структуры, позволяющие дальнейшие вычисления;
* распределение «тяжелых» данных по отдельным машинам внутри вычислительной сети;
* оптимизация текущей нагрузки между отдельными вычислительными узлами;
* приоритизация задач с учетом специфики порядка получения результатов конечным пользователем;
* возможность параллелизации операций доступа к данным, их изменения и произведения вычислительных операций над ними в рамках одной машины.

# Обзор методов выполнения растровой кластеризации и классификации

Рассмотрим по-отдельности имеющиеся в распоряжении архитектора Географической Информационной Системы методы и продукты, позволяющие решить ту или иную задачу.

## Масштабирование системы

В то время как вертикальное масштабирование представляет собой ограниченное по техническим причинам оснащение сервера дополнительными вычислительными модулями и модулями памяти, возможность горизонтального масштабирования существенно зависит от выбранного технологического стэка программного обеспечения и представляет более трудоемкую задачу по организации кластера вычислительных устройств и распределенной системы хранения данных. Рассмотрим наиболее используемые в настоящее время продукты, в той или иной мере позволяющие смоделировать и реализовать подобную систему.

### Hadoop

Apache Hadoop -- это проект фонда Apache Software Foundation, свободно распространяемый набор утилит, библиотек и фреймворк для разработки и выполнения распределённых программ, работающих на кластерах из сотен и тысяч узлов [1]. Используется для реализации поисковых и контекстных механизмов многих высоконагруженных веб-сайтов, в том числе, для Yahoo! и Facebook. Разработан на Java в рамках вычислительной парадигмы MapReduce [2], согласно которой приложение разделяется на большое количество одинаковых элементарных заданий, выполнимых на узлах кластера и естественным образом сводимых в конечный результат. Проект состоит из четырёх модулей -- Hadoop Common (связующее программное обеспечение -- набор инфраструктурных программных библиотек и утилит, используемых для других модулей и родственных проектов), HDFS (распределённая файловая система), YARN (система для планирования заданий и управления кластером) и Hadoop MapReduce (платформа программирования и выполнения распределённых MapReduce-вычислений).

### Spark

Apache Spark -- программный open-source комплекс для реализации распределённой обработки неструктурированных и слабоструктурированных данных, входящий в экосистему проектов Hadoop. В отличие от классического обработчика из ядра Hadoop, реализующего двухуровневую концепцию MapReduce с дисковым хранилищем, использует специализированные примитивы для рекурентной обработки в оперативной памяти, благодаря чему позволяет получать значительный выигрыш в скорости работы, в частности потому, что дает возможность многократного доступа к загруженным в память пользовательским данным [3]. Проект предоставляет программные интерфейсы для языков Java, Scala, Python, R. Состоит из ядра и нескольких расширений, таких как Spark SQL (позволяет выполнять SQL-запросы над данными), Spark Streaming (надстройка для обработки потоковых данных), Spark MLlib (набор библиотек машинного обучения), GraphX (предназначено для распределённой обработки графов). Может работать как в среде кластера Hadoop под управлением YARN, так и без компонентов ядра Hadoop, поддерживает несколько распределённых систем хранения -- HDFS, OpenStack Swift, NoSQL-СУБД Cassandra, Amazon S3.

### Akka

Akka -- набор инструментов с открытым исходным кодом, позволяющими в режиме реального времени упрощать конструирование многопоточных и распределенных приложений, использующий Java Runtime Machine [4]. Akka поддерживает множество программных модулей для многопоточности, но особое внимание уделяется модели, связанной с акторами, имеющей корни в языке Erlang. Поддерживаются программные интерфейсы для Java и Scala. Сама библиотека написана на Scala, и с версии 2.10 реализация Akka включена в стандартную библиотеку этого языка. Основными особенностями Akk являются основанная на сообщениях, асинхронная конкуренция между акторами, благодаря которой не требуется использовать примитивы синхронизации и разделение изменяемых объектов между отдельными акторами, и единый интерфейс для взаимодействия как между акторами, находящимися на одном хосте, так и между физически разделенными по кластеру.

## Варианты выбора СУБД

Логическая организация данных осуществляется при помощи СУБД, имеющей возможность хранения географических или просто растровых данных, в случае использования неструктурированных карт-изображений. В то время как очевидным представляется решение вопроса путем самостоятельного описания структуры данных с использованием примитивов, предоставленных интерфейсом СУБД, такой подход может снизить производительность из-за дополнительных издержек на выполнение внешнего кода. В связи с этим имеет смысл рассмотреть СУБД, поддерживающие хранение географических данных «из коробки» или имеющие расширения, позволяющие добавить требуемую функциональность.

### MySQL

MySQL хранит географические данные в формате WKT (Well-known Text) или в бинарном эквиваленте, WKB (Well-knowo Binary). Имеются примитивы для хранения точек, линий и многоугольников, а также встроенные средства для нахождения пересечений и расстояний между объектами [5]. WKT -- достаточно старый стандарт хранения географических данных, представляющий точки и геометрию в виде набора текстовых инструкций. Например, для хранения точки с широтой и долготой 12 и -75, соответственно, в WKT-файле будет содержаться строка «POINT (12 -75)» [6]. MySQL предоставляет множество функций для вычисления расстояния между точками, выбора точек внутри квадрата и т. д. Недостаток хранения данных в MySQL заключается в том, что все эти вычисления происходят в евклидовом пространстве, а значит -- на плоскости. Поэтому для работы именно с географическими данными, а не просто с абстрактной геометрией, недостаточно имеющихся процедур обработки данных и возникает необходимость написания процедур на языке SQL или внешнего кода, что снова приводит к дополнительным издержкам.

### PostGIS

PostGIS -- это расширение для СУБД PostgreSQL, добавляющее функции работы с географическими данными [7]. Оно работает во многом схоже с MySQL, когда хранит данные в формате WKT или WKB. Оно имеет такие же функции для определения дистанций, пересечений и т. п. Главная разница заключается в том, что PostGIS поддерживает специальные «географические» координаты, что означает поддержку PostGIS работы с точками на глобусе, а не в Евклидовом пространстве.

### MongoDB

MongoDB с 2010 года поддерживает индексацию географических данных, и сохраняет местоположение в виде т. н. географического хэша [8], что позволяет хранить широту и долготу в виде одномерного значения. Более того, на самом деле геохэш хранит не просто точку, а некоторый участок пространства на глобусе, в котором находится объект. MongoDB позволяет хранить только точки, хранение многоугольников и линий недоступно. В то время как в общем случае это может быть не удовлетворительным решением, во многих частных ситуациях подобной функциональности вполне хватает. Из встроенных функций следует отметить возможность поиска точек вблизи некоторой заданной или внутри прямоугольника, работающую с достаточно высокой производительностью. Существенным минусом является предположение о сферической форме Земли.

### SpatiaLite

SpatiaLite является расширением SQLite для работы с географическими, в первую очередь векторными, данными. Эта база данных является свободно-распространяемой и кроссплатформенной и добавляет в SQLite возможности по работе с проекциями и позволяет выполнять более сложные операции над векторными типами данных. Поддержка растровых типов данных предоставлена только в базовом виде [9]. Преимуществами SQLite являются хорошая поддержка векторных типов данных. К недостаткам стоит отнести слабую поддержку растровых типов данных. Простая персональная архитектура данной БД -- модуль встраивается в само приложение, а данные хранятся как обычные файлы -- может быть как преимуществом, так и недостатком, в зависимости от того, в каких условиях SpatiaLite будет использоваться. В случае с разрабатываемой в рамках данной работы системы, это является именно недостатком, поскольку для нее требуется полноценная и независимая от каких-либо контейнеров приложений база данных.

### CouchDB

CouchDB является свободно распространяемой NoSQL объектно-ориентированной базой данных на основе BerkeleyDB. Для хранения данных в CouchDB используются контейнеры JSON, а запросы в формате функций MapReduce могут быть написаны на Java, JavaScript или других подобных языках. Встроенной поддержки географических типов данных не имеет, но необходимый функционал может быть реализован через функции внутри базы данных или же как часть функции Reduce запроса [10, 11]. Основным достоинством CouchDB является ее ориентированность на web: эта база данных является само-публикующейся и может использоваться в том числе как web-сервис и web-сервер. Широкий спектр возможностей, который может быть запрограммирован также является преимуществом CouchDB [10]. К недостаткам стоит отнести возможно довольно сложные запросы и требования к уровню подготовки администратора БД. В рамках решаемой задачи, отсутствие нативной поддержки растровых и векторных типов данных тоже можно считать недостатком CouchDB.

## Распределение данных по кластеру

Растровые географические данные страдают от таких проблем как плохая масштабируемость и большие объемы используемой памяти, в первую очередь, оперативной. Для решения или обхода этих проблем используются техники хранения в виде пирамиды и мозаики. Использование пирамиды уменьшает объем производимых вычислений за счет избыточности данных: при меньшем приближении используются заранее подготовленные растры меньшего разрешения. Всего в пирамиде может быть не более уровней [12]. Использование пирамиды увеличивает объем хранимых на диске данных не более, чем в два раза.

Мозаика используется для уменьшения нагрузки на оперативную память при работе с большими растрами путем разбиения файла, содержащего в себе все данные данного слоя или данного уровня пирамиды, на набор файлов меньшего размера [13]. Вместе эти две техники позволяют минимизировать использование оперативной памяти и уменьшить нагрузку на процессор при получении данных из указанной области слоя.

При использовании кластера для обработки растра, разбитого на пирамиду и мозаику, наиболее эффективное использование вычислительных ресурсов будет достигаться в том случае, когда все одновременно используемые элементы мозаики располагаются на различных узлах кластера.

## Параллельная обработка растровых данных внутри одной машины

Нативная сущность географических данных, представленных в виде карты, предоставляет широкие возможности для ускорения операций растровой реклассификации и растровой алгебры путем параллелизации и использования SIMD-инструкций. Кроме того, существующие графические процессоры позволяют, при соответствующих модификациях алгоритмов, существенно ускорить подобные операции [14, 15]. В настоящее время наиболее широко применяемыми технологиями параллельной обработки данных являются следующие:

### Advanced Vector Extensions

Advanced Vector Extensions (AVX, AVX2) -- это расширение к набору инструкций для микропроцессоров от Intel и AMD, предложенное в 2008 году компание Intel и предлагающее новый набор инструкций, функций и схем программирования [16]. Начиная с микроархитектур Sandy Bridge и Bulldozer, соответственно, процессоры от Intel и AMD имеют набор векторных регистров с шириной в 128 или 256 бит, позволяющих загрузить множество данных для однотипной обработки.

### Nvidia CUDA

CUDA, или Compute Unified Device Architecture -- программно-аппаратная архитектура параллельных вычислений, которая позволяет существенно увеличить вычислительную производительность благодаря использованию графических процессоров фирмы Nvidia [17]. CUDA SDK позволяет программистам реализовывать на специальном упрощённом диалекте языка программирования C алгоритмы, выполнимые на графических процессорах Nvidia, и включать специальные функции в текст программы на C. Архитектура CUDA даёт разработчику возможность по своему усмотрению организовывать доступ к набору инструкций графического ускорителя и управлять его памятью.

### OpenCL

Open Computing Language -- фреймворк для написания компьютерных программ, связанных с параллельными вычислениями на различных графических и центральных процессорах, а также FPGA [18]. В OpenCL входят язык программирования, который основан на стандарте языка программирования C-99, и интерфейс программирования приложений. OpenCL обеспечивает параллелизм на уровне инструкций и на уровне данных и является осуществлением техники GPGPU. OpenCL является полностью открытым стандартом, его использование не облагается лицензионными отчислениями. Цель OpenCL состоит в том, чтобы дополнить открытые отраслевые стандарты для трёхмерной компьютерной графики и звука OpenGL и OpenAL возможностями GPU для высокопроизводительных вычислений. OpenCL разрабатывается и поддерживается некоммерческим консорциумом Khronos Group, в который входят много крупных компаний, включая AMD, Apple, ARM, Intel, Nvidia, Sony Computer Entertainment и другие.

# Оптимальное распределение текущей нагрузки между хостами в вычислительной сети

В последние десятилетия все большее распространение получают технологии распределенных и облачных вычислений, призванные облегчить и ускорить выполнение ресурсоемких задач, не привлекая к их выполнению суперкомпьютеры и специальное оборудование \cite{cloud}. Одним из ключевых отличий таких систем от классического кластера часто является принципиальная неоднородность устройств, вовлеченных в процесс: например, для вычислений на движке браузера могут быть использованы практически произвольные устройства, вплоть до смартфонов и иных видов КПК \cite{volunteer}. При этом обычно задача распределения нагрузки в таких системах отдельно не решается, что может приводить как к чрезмерной загрузке отдельных устройств, что в случае добровольного согласия на предоставление вычислительных мощностей может обернуться рядом неудобств для пользователей системы, так и к появлению «бутылочного горлышка» в точке распределения. В рамках данного раздела рассмотрим способ оценки вычислительной мощности устройств в распределенной сети с учетом ее возможного изменения во времени.

## Постановка задачи

При построении модели принимаются в качестве базовых следующие утверждения:

* устройства в сети производят вычисления независимо друг от друга;
* внутри одного устройства время последующего вычисления не зависит от времени предыдущего;
* время работы, подлежащее измерению, значительно превосходит доступный квант времени;
* вычислительная сеть имеет большую нагрузку и всегда имеет очередь задач, готовых к выполнению.

Для оценки вычислительной способности устройства производится оценка времени выполнения им отдельного практически значимого задания. При построении модели будем предполагать эквивалентность таких операций, в противном случае придется каким-либо образом производить нормировку полученных сведений --- например, измерять время выполнения какого-либо числа заданных элементарных операций или работы над заданным объемом информации.

Целью раздела является моделирование системы, способной в автоматическом режиме оценивать вычислительную мощность доступных устройств, входящих в состав распределенной сети, изменяющуюся с течением времени.

## Построение модели

### Статическая модель

Независимость устройств друг от друга позволяет моделировать их по отдельности. Предположим на данном этапе, что выбранное устройство статично в том смысле, что не меняет своих параметров с течением времени. В качестве объекта наблюдения будем использовать момент окончания выполнения отдельной задачи. Тогда из приведенных выше утверждений следует независимость количества таких моментов на произвольно взятом отрезке временной шкалы от других, а вкупе с очевидными свойствами процесса, такими как неотрицательное число измерений, следует применимость и адекватность модели пуассоновского процесса \cite{stochastic}, в котором на произвольно взятом отрезке времени будет ожидаться законченных вычислений, где --- это частота выполнения операций, полагаемая постоянной.

Для оценки этого параметра перейдем от непосредственно распределения числа вычислений к распределению времени между двумя последовательными их окончаниями. Для пуассоновского процесса это распределение является экспоненциальным, при этом параметр этого распределения совпадает с параметром «родительского» распределения Пуассона \cite{stochastic}. Воспользуемся байесовским подходом \cite{Bure} для определения распределения этого параметра:

В качестве априорного распределения параметра имеет смысл выбрать гамма-распределение \cite{Bure}: про него изначально известна лишь неотрицательность принимаемых значений, и, кроме того, при таком выборе апостериорное значение параметра тоже будет иметь вид гамма-распределения:

Это свойство позволяет легко интерпретировать априорные значения параметров и : поскольку при наличии серии измерений времени выполнения операций параметр дополняет их количество, а ~--- сумму, начальные значения стоит выбирать так, чтобы они отражали ожидаемую производительность. Таким образом, выбрав начальные значения и из имеющихся априорных сведений об устройстве, мы можем получить начальное значение как реализацию случайной величины, подчиняющейся гамма-распределению с заданными параметрами, после каждой новой выполненной операции обновлять сведения о распределении и получать его новое значение как реализацию новой случайной величины.

### Добавление динамики

Для учета динамики параметров будем вносить «поправки» в модель в случае, если реальное время обработки не согласуется с ней. Поскольку реальное распределение времени обработки запроса нам неизвестно, в качестве подхода к определению «несогласованности» можно использовать относительное отклонение от текущего среднего значения. «Исправление» модели заключается в «забывании» старых значений, при этом чем хуже модель согласуется с реальностью, тем больше старых значений будет «забыто». При этом параметры и изменяются соответствующим образом, и значение имеет смысл «уверенности» в модели, поскольку его увеличение уменьшает дисперсию распределения оценки исходного параметра. Воспользуемся следующей эвристикой: при относительной погрешности более заданного заранее будем оставлять последних полученных значений, при этом под будем понимать ошибку относительно максимального из текущего среднего и полученного значения, что даст модели некоторую устойчивость к выбросам.

На рис.~1 слева показана динамика параметров при изменении среднего времени выполнения операций на устройстве с 200 до 600 условных единиц, происходящем на 50-ом отсчете, а справа~--- вид кривой распределения по окончании моделирования.

КАРТИНКИ

Предложенный метод оценки вычислительной мощности устройств, входящих в состав распределенной вычислительной сети, может найти наилучшее применение в добровольных и гибридных вычислительных сетях. В то время как в первых он позволяет динамически отслеживать мощность доступных ресурсов и управлять загруженностью устройств, в сетях второго типа он может понадобиться при оценке реальных вычислительных возможностей сети и в разгрузке основных серверов с переносом части нагрузки на клиентскую сторону и вторичные устройства, находящиеся в сети.

# Реализация отложенных вычислений в контексте Геоинформационной системы

Отложенные вычисления, зачастую в комбинации с мемоизацией, в общем случае применяются для уменьшения времени работы произвольной вычислительной системы за счет отсутствия необходимости производить вычисления в точке объявления результата и его кэширования после успешного выполнения операции в том случае, когда высока возможность его повторного использования. Существует множество языков программирования, поддерживающих ленивые вычисления на уровне ядра, но, поскольку специфика поставленной задачи подразумевает распределенные вычисления, в общем случае они не принесут существенной пользы.

В контексте распределенных растровых вычислений отложенные вычисления в первую очередь связаны с частым отсутствием необходимости производить заданные пользователем системы операции над всем требуемым объемом данных, поскольку их результаты не всегда требуются для отображения на устройстве аналитика, использующего сервис. Используемая в предыдущем разделе модель гетерогенной сети позволяет разделить задачу отложенных вычислений как минимум на две части:

* реализация отложенных вычислений на серверной стороне системы;
* реализация отложенных вычислений при использовании клиентской стороны в качестве вычислительной.

При этом необходимо учитывать разнородность клиентских устройств, поскольку даже при наличии возможности производить часть вычислений на стороне мобильного устройства их выполнение может быть связано с рядом трудностей, возникающих из-за сторонних причин, таких как нестабильное сетевое соединение или недостаточная пропускная способность сетевого канала. В этом случае, несмотря на большую вычислительную мощность, получающуюся при добавлении в систему нового узла в виде клиентской машины, необходимо структурировать вычисления такие образом, чтобы сбалансировать время, требующееся на передачу неструктурированных и структурированных данных между клиентским и серверным устройствами, и время, необходимое для непосредственного выполнения запрашиваемых операций, опять же, на серверной и клиентской стороне.

## Определение структуры вычислительной схемы

В этом пункте полученная в предыдущем разделе модель будет применяться для определения типа вычислительной схемы (серверная, клиентская или гибридная) при выполнении запрошенных операций.

Исходно будем основываться на предположении о том, что для каждого типа устройства , где --- множество всех известных типов устройств, существуют и определены в процессе работы системы априорные значения гиперпараметров и распределения параметра .

Поскольку принципиальным критерием, по которому должны выбираться тот или иной подход к реализации вычислений, является время выполнения операции, необходимо учитывать как время работы системы, так и время пересылки данных по сети. В модели системы, предложенной в предыдущем разделе, оба типа временных затрат учитываются при оценке единственного параметра --- ожидаемой частоты выполнения операций. При продолжительном времени работы системы априорная частота для определенного типа устройств будет усредненной как по их мощности, так и по сетевому соединению, характерному для них. В рамках определения схемы вычислений такой подход может быть не совсем корректным: из-за возможного расхождения в параметрах сети для похожих устройств, относящихся к одному классу, при ориентировании только лишь на априорные сведения время сетевой передачи растровых данных может быть сильно недооценено.

Эта проблема имеет как минимум два решения: предоставление возможности конечному пользователю самому дать изначальную информацию о типе и скорости своего соединения, и оценка его при помощи отправки какого-либо набора тестовых данных. Первый случай достаточно тривиален, но допускает возможность человеческой ошибки, поэтому может быть скомбинирован со вторым для достижения уверенности в результате.

Дальнейшая работа с устройством зависит от того, насколько наши ожидания от работы с ним совпадают с реальностью. Для определения степени этого совпадения применим следующую процедуру:

1. Устройству пользователя посылается небольшое тестовое «задание», имеющее априорную оценку времени работы , где --- количество элементарных операций внутри тестового блока, в которых измерялась сложность операции внутри статистической модели.
2. Измеряется реальное время работы над тестовыми данными .
3. Выдвигается гипотеза о принадлежности полученного значения к генеральной совокупности, подчиняющейся гамма-распределению с параметрами и .
4. При выполнении условий применимости априорного распределения гиперпараметры и применяются для дальнейшего построения апостериорной модели устройства, в противном случае --- отбрасываются и модель работы устройства строится «с нуля».

Таким образом, посланные тестовые данные могут быть использованы сразу для двух поставленных целей: определения качества сетевого соединения и построения статистической модели устройства. При этом объем переданных данных должен быть достаточно небольшим, чтобы не сильно повлиять на общее время работы, которое в этом случае должно максимально возможно состоять лишь из времени пересылки данных по сети.

Будем считать, что на данном этапе модель устройства определена (либо из априорных соображений, либо путем ее перестроения в случае неудовлетворительной точности априорной модели), как определено и качество сетевого соединения. В качестве критерия оптимальности работы системы примем общее время решения некоторой поставленной задачи , которую можно разбить на независимые подзадачи , которые, в свою очередь, могут быть распространены по узлам вычислительной сети. Задача состоит из запросов пользователей к системе, и результаты ее выполнения требуется переслать на соответствующие узлы сети. Под распространением подзадачи будем понимать пересылку по сети как информации о самой подзадаче, так и данных, над которыми будут производиться манипуляции. Будем полагать, что пропускной канал головных серверов с данными достаточно широк для удовлетворения потребности в их пересылке на вычислительные узлы в параллельном режиме. Тогда общее время работы над задачей будет равно максимуму времени обработки своей подзадачи среди всех вычислительных узлов, что и предстоит минимизировать.

Подзадача , выполняемая -ым устройством в сети, состоит из суммы независимых подзадач, результат части из которых необходим на самом узле, а остальные необходимо переслать другим участникам сети напрямую или через головной сервер. Обозначим эти части как и ; . Тогда время работы произвольного узла над своей задачей можно представить в виде суммы времен:

* пересылки по сети подзадачи, результат которой необходим на самом узле;
* работы над этой частью задачи;
* пересылки по сети подзадачи, результат выполнения которой необходим на другом узле;
* работы над ней;
* минимуму из времени пересылки результатов, необходимых на других узлах:
  + непосредственно;
  + через один из центральных узлов.

В формульном виде это можно представить следующим образом:

где

* --- время пересылки единицы данных по сети с одного из центральных узлов на -ый вычислительный узел;
* --- время пересылки единицы данных по сети с -ого узла на один из центральных (в общем случае );
* --- коэффициент, показывающий, как в среднем соотносится количественно информации в ответе с ней же в задаче; поскольку под задачами в рамках данной работы подразумевается работа с растрами, этот коэффициент равен , где --- среднее количество растров, оперирование над которыми в конечном итоге даст один растр, содержащий результаты вычислений. Применимость такого «усредненного» подхода к отношению количества информации в задаче и результате следует из статистического характера применяемой модели, призванной работать в условиях большого числа как вычислительных узлов, так и данных;
* --- время пересылки единицы данных по сети с -ого узла на -ый (в общем случае );
* --- часть данных, обрабатываемых на -ом узле, результат выполнение которых ожидается на -ом.

Центральные выделенные сервера также входят во множество проиндексированных вычислительных узлов; их особенность состоит в нулевых значениях и .

Сформированный вектор представляет собой набор времен выполнения выданных заданий на всех узлах. Поскольку операции на различных узлах выполняются параллельно и независимо друг от друга, в качестве функционала качества на данном этапе имеет смысл выбрать максимум по из всех (задача считается выполненной тогда, когда закончены все вычисления на всех узлах):

Здесь за обозначена матрица, элементами которой являются переменные , отвечающие за количество данных, работа на которыми будет произведена на -ом узле, а результаты необходимо переслать на -ый узел. Эта матрица является параметром, по которому может быть произведена оптимизация.

Заметив, что все элементы вектора являются неотрицательными, выражение можно переписать как векторную норму:

где обозначает стандартную норму в Евклидовом пространстве вида

При этом саму задачу можно описать как попытка минимизации этого функционала:

Явный вид функционала зависит от выражения, заменяющего собой функцию минимума из . Саму задачу его минимизации можно решать различными способами, но независимо от его выбора элементы вектора имеет смысл упростить. Рассмотрим для начала случай, когда для -го вычислительного узла выбран режим работы, при котором он пересылает результаты вычислений непосредственно на другие узлы, минуя центральный сервер:

В векторно-матричном виде:

Здесь

* --- вектор, состоящий из ;
* --- вектор, состоящий из ;
* обозначает диагональную матрицу, на главной диагонали которой находятся элементы вектора , а остальные являются нулями;
* --- вектор-столбец из единиц;
* --- матрица, на позиции которой стоит время пересылки единицы данных с -го узла на -ый (очевидно, по диагонали матрицы стоят нули);
* за обозначено произведение Адамара.

В случае, если на -ом вычислительном узле выбран режим работы, при котором он пересылает результаты вычислений на центральный узел, выражение преобразовывается несколько по-другому:

В векторно-матричном виде это выражение удобно представить, разложив матрицу на сумму матриц и , где состоит из главной диагонали матрицы с нулями на всех недиагональных элементах, а матрица аддитивно дополняет ее до матрицы . В этом случае его можно переписать в виде

Обобщением рассмотренных выше случаев является гибридный вариант, при котором часть данных передается напрямую узлу, нуждающемуся в них, а часть при этом проходит один из центральных узлов. В этом случае, воспользовавшись равенством

выражение можно переписать:

Далее его можно усилить (в силу необходимости его дальнейшей минимизации):

Окончательно, обозначив за , можно записать

что в векторно-матричном представлении дает

где за обозначена матрица, на позиции которой стоит .

### Решение формализованной задачи

Несмотря на очевидную линейность получившихся во всех трех случаях выражений, только вариант передачи данных через сервер не предполагает использования произведения Адамара. Кроме того, матрица переменных всегда домножается на константы с обеих сторон. Тем не менее, сама по себе задача является типичной задачей на минимакс и может быть решена стандартными методами (вплоть до перебора по конечной сетке в пространстве допустимых значений передаваемой информации) при помощи какой-либо системы компьютерной алгебры, поддерживающей решение задач на условную оптимизацию.

ТУТ НЕБОЛЬШОЕ ИССЛЕДОВАНИЕ СВОБОДНЫХ СКА НА ПРЕДМЕТ НАЛИЧИЯ ТАКИХ СПОСОБНОСТЕЙ

Более аккуратное решение задачи можно получить, учитывая ее специфику, а именно --- линейность по всем переменным. Одним из возможных преобразований является «вытягивание» матрицы в вектор по правилу:

где --- число узлов в сети. Далее, поскольку гибридная задача может быть заменой на или преобразована как к задаче распределения нагрузки при непосредственной передаче данных, так и к ее варианту с использованием одного из центральных серверов, преобразуем только выражение . Матрицу заменим на матрицу по правилу

для всех от до ; оставшиеся элементы матрицы установим равными нулю. Произведение Адамара матриц и , каждая из которых имеет размерность , домноженное на вектор из единиц, также имеющий размерность , заменим на матрицу = по правилу

Аналогично предыдущему случаю, и меняются от до , а оставшиеся элементы матрицы устанавливаются равными нулю. Теперь, с использованием новых обозначений, можно переписать :

Введя матрицу как сумму матриц и , можно переписать выражение , используя новые обозначения:

Задача минимизации

должна быть решена при условии

вытекающем из требования каждого узла полностью решить поставленную им задачу и только ее. Она может быть сведена к задаче линейного смешанного целочисленного программирования классическим способом [Stanford] путем посредством введением искусственной переменной , представляющей собой максимальное время работы и передачи информации на всех устройствах:

где --- строки матрицы .

Особенностью данной задачи является отсутствие прямой необходимости нахождения оптимального решения: в силу принципиальной хаотичности как поведения устройств по отдельности, так и пропускной способности сети нас устроит любое решение, близкое к оптимальному. В силу NP-полноты задачи целочисленного программирования, этим стоит воспользоваться для облегчения поиска «удовлетворительного» решения. Во-первых, следует решить соответствующую линейно ослабленную задачу, а затем воспользоваться одним из методов:

* при решении задачи методом «ветвей и границ» после наперед заданного числа итераций при условии, что процесс поиска решения еще не завершен, начинать проверять решение на близость к оптимальному решению соответствующей задачи линейного программирования и при достаточно высокой точности завершать работу алгоритма, не доводя его до конца;
* округлить все полученные значения переменных к ближайшему большему целому, после чего
  1. найти все такие, что
  2. уменьшить на единицу значение для узла с наибольшим значением ;
  3. в случае, если условие все еще выполняется, уменьшить значение для следующего узла с наибольшим значением без учета первого; этот шаг необходимо проделать не более раз в сумме для всех узлов, поскольку в любом случае при округлении всех переменных в большую сторону выполняется неравенство
  4. как только решение начнет удовлетворять ограничениям, закончить процесс и принять его в качестве оптимального.

Несмотря на то, что первый метод имеет под собой лучшую теоретическую основу, второй может работать быстрее, поскольку NP-трудная задача не решается в принципе: вместо нее решается задача линейного программирования и выполняется не более дополнительных шагов для нахождения близкого к оптимальному решения.

# Оптимизация пользовательских запросов

Одной из задач, встающих перед архитектором любой системы, позволяющей пользователям задавать практически произвольные операции над данными в рамках предоставленных примитивов, всегда является автоматизация упрощения подобных запросов. Это в полной мере справедливо и для геоинформационных систем и, в частности, для O-GIS. Пользователь системы имеет возможность производить необходимые вычисления над имеющимися в его распоряжении растровыми полями и массивами данных, декларируя операции в строковом виде.

Спецификой, предлагаемой в рамках создаваемого модуля реализации вычислений, является использование нативного кода на C++ для обработки растровых операций на вычислительных узлах. Поскольку в случае невозможности использования системы компьютерной алгебры из C++ кода появляется необходимость либо в использовании дополнительной программной прослойки, обеспечивающей взаимодействие СКА с кодом, ответственным за непосредственное исполнение растровых операций, неизбежно ведущая к уменьшению производительности системы, одним их параметров исследования существующих продуктов является именно возможность взаимодействия с C++ кодом. Другим вариантом решения проблемы является оптимизация запросов перед их отправкой на вычислительный узел, что так же негативно скажется на производительности системы в целом.

## 5.1. Обзор существующих решений

Поскольку одной из идеологий разработки O-GIS является использование только свободно распространяемого программного обеспечения, существующие проприетарные решения, подобные Wolfram Mathematica, Maple или MATLAB, не включены в данный раздел. Все обозреваемые программные комплексы являются проектами с открытым исходным кодом, ссылки на который представлены в разделе литературы в конце работы.

### Maxima

Maxima --- обособившаяся в 1982 году ветка развития проекта Macsyma, имеющего своим началом 60-ые годы прошлого века и разрабатывающегося в Массачусетском технологическом институте. Проект развивается и сейчас, на момент написания текста последняя доступная версия продукта выпущена в октябре 2017 года. Имеет достаточно широкий круг применения, включая возможности упрощения символьных выражений. Основная часть кода написана на LISP, и крупным недостатком системы в контексте настоящей работы является отсутствие возможности вызова функций Maxima из C++ кода.

### Scilab

Scilab --- свободно распространяемый пакет компьютерной алгебры с открытыми исходными кодами, разработка которого была начата в 1990 году под именем Ψlab (Psilab). На данный момент он представляет собой один из немногих аналогов проприетарной системы компьютерной алгебры Matlab. Функциональность пакета, хоть и существенно меньшая в таких областях применения, как, например, управление, позволяет проводить все стандартный алгебраические операции. В настоящее время существует отдельный инструментарий для работы с символьными вычислениями, основанный на Maxima, а также существует API доступа к ресурсам системы из нативного C++ кода.

### JACAL

JACAL --- интерактивная система компьютерной алгебры, разрабатываемая в качестве UNIX-пакета. JACAL написан на алгебраическом языке SCHEMA, разработанным той же командой разработчиков. Его основная цель -- упрощение и манипуляции с алгебраическими уравнениями, а также работа с векторами и матрицами. Несмотря не то, что его функциональности достаточно для нужд разрабатываемого в рамках данной работы модуля, отсутствие какого-либо внешнего интерфейса не позволяет использовать его в качестве средства оптимизации запросов. Проект не разрабатывается с января 2015 года.

### GiNaC

GiNaC (GiNaC is not a CAS) --- библиотека, написанная на C++, специально разрабатываемая для расширения возможностей языка в области прикладных и теоретических алгебраических расчетов, интегрированных в нативные приложения. К сожалению, разработчики сознательно отказались от имплементации алгоритмов, позволяющих упрощать символьные выражения. Хоть в пакете и присутствуют некоторые возможности манипуляций с символьными выражениями, которые можно использовать с целью их упрощения, они крайне лимитированы и не предназначены для использования в этих целях. Конечно, как и всякий другой проект с открытыми исходными кодами, GiNaC дает возможность пользователю самостоятельно реализовать недостающую функциональность, но в рамках данной работы это не представляется рациональным решением, с учетом отсутствия необходимости произведения сложным манипуляций над пользовательскими запросами.

### YACAS

YACAS (Yet Another Computer Algebra System) --- система компьютерной алгебры, имеющая написанное на C++ ядро и представляющая значительные возможности по работе с символьными выражениями, в том числе и возможности по их упрощению. Отдельно распространяется как библиотека для Java, и может быть подключен в качестве статической библиотеки в C++ приложение. Имеет встроенную поддержку упрощения символьных выражений.

### Octave

Octave --- изначально разрабатываемая в качестве свободно распространяемого аналога Matlab система компьютерной алгебры, имеющая возможности вызова своего интерпретатора из C++ кода. Предназначена для численных решений ряда линейных и нелинейных задач и имеет сопоставимые с Matlab возможности визуализации данных. К сожалению, ядро Octave не позволяет выполнять символьные операции, и инструментарий для работы с ними существует в качестве одного из пакетов Octave Forge, что усложняет линковку.

# Список литературы

1. Официальное описание системы распределенного хранения данных Hadoop. <http://hadoop.apache.org/docs/current>
2. Aji, Ablimit, Fusheng Wang, Hoang Vo, Rubao Lee, Qiaoling Liu, Xiaodong Zhang, and Joel Saltz. "Hadoop gis: a high performance spatial data warehousing system over mapreduce." Proceedings of the VLDB Endowment 6, no. 11 (2013): 1009-1020.
3. Zaharia, Matei, Reynold S. Xin, Patrick Wendell, Tathagata Das, Michael Armbrust, Ankur Dave, Xiangrui Meng et al. Apache Spark: A unified engine for big data processing. Communications of the ACM 59, no. 11 (2016): 56-65.
4. Wyatt, Derek. Akka concurrency. Artima Incorporation, 2013.
5. Официальная документация СУБД MySQL. <https://dev.mysql.com/doc/refman/5.7/en/spatial-analysis-functions.html>
6. Описание формата Well Known Text в GIS Wiki. <http://giswiki.org/wiki/Well_Known_Text>
7. Официальное описание проекта пространственной СУБД PostGIS. <http://postgis.net/docs/manual-2.0/>
8. Официальное описание проекта NoSQL СУБД MongoDB. <https://docs.mongodb.com/manual/core/geospatial-indexes/index.html#calculation-of-geohash-values-for-2d-indexes>
9. Официальное описание проекта пространственной СУБД SpatialLite. <http://www.gaia-gis.it/spatialite>
10. Официальное описание проекта NoSQL СУБД CouchDB. <http://couchdb.apache.org/>
11. Clarence J. M. T., Aravindh S., Shreeharsha A. B. Comparative study of the new generation, agile, scalable, high performance NOSQL databases. 2012. <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.258.9602>
12. Oracle Spatial GeoRaster Developer’s Guide: http://docs.oracle.com/cd/B28359\_01/appdev.111/b28398/geor\_intro.htm
13. Xu Hong, Mangtani P. Managing imagery and raster data using mosaic datasets // ESRI International User Conference. Technical Workshop. 2012.
14. Bajerski, Piotr. Optimization of geofield queries. In Information Technology, 2008. IT 2008. 1st International Conference on, pp. 1-4. IEEE, 2008.
15. Carabaño, Jesús, Jan Westerholm, and Tapani Sarjakoski. A compiler approach to map algebra: automatic parallelization, locality optimization, and GPU acceleration of raster spatial analysis. GeoInformatica (2017): pp. 1-25.
16. Lomont C. Introduction to Intel® Advanced Vector Extensions. <https://software.intel.com/en-us/articles/introduction-to-intel-advanced-vector-extensions>
17. Malakar, Ranajoy, and Naga Vydyanathan. A CUDA-enabled Hadoop cluster for fast distributed image processing. Parallel Computing Technologies (PARCOMPTECH), 2013 National Conference on. IEEE, 2013.
18. Stone, J. E., Gohara, D., & Shi, G. (2010). OpenCL: A parallel programming standard for heterogeneous computing systems. Computing in science & engineering, 12(3), pp. 66-73.