МИНОБРНАУКИ РОССИИ

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ**

**ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ**

**“ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ”**

Факультет *компьютерных наук*

Кафедра *программирования и информационных технологий*

*Применение технологий Hadoop для анализа "больших данных" на примере задач геоаналитики и машинного обучения*

ВКР *Магистерская диссертация*

09.04.02 Информационные системы и технологии

Технологии разработки информационных систем

Допущено к защите в ГЭК

Зав. кафедрой \_\_\_\_\_\_\_\_\_\_\_\_\_\_*Н.А. Тюкачёв, к. ф - м. н., доцент* \_\_.\_\_.2016

Студент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*А.Г. Чурсин, 2 курс, д/о*

Руководитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ *Н.А. Тюкачёв, к. ф - м. н., доцент*

Консультант \_\_\_\_\_\_\_\_\_\_\_\_\_ *Д.И. Соломатин*, *старший преподаватель*

Воронеж 2016

МИНОБРНАУКИ РОССИИ

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ**

**ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ**

**«ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»**

Факультет \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Кафедра\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

УТВЕРЖДАЮ

заведующий кафедрой

\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_

*подпись, расшифровка подписи*

\_\_.\_\_.20\_\_

**ЗАДАНИЕ**

**НА ВЫПОЛНЕНИЕ ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЫ**

**ОБУЧАЮЩЕГОСЯ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

*фамилия, имя, отчество*

1. Тема работы \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_, утверждена решением ученого совета \_\_\_\_\_\_\_\_\_\_\_ факультета от \_\_ .\_\_.20\_\_

2. Направление подготовки / специальность \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*шифр, наименование*

3. Срок сдачи законченной работы \_\_.\_\_ 20\_\_

4. Календарный план: (строится в соответствии со структурой ВКР)

|  |  |  |  |
| --- | --- | --- | --- |
| № | Структура ВКР | Сроки выполнения | Примечание |
|  | Введение | 4.09.15-9.09.15 |  |
| 1 | Постановка задачи | 10-12.09.15 |  |
| 2.1 | Анализ предметной области | 15-28.09.15 |  |
| 2.2 | Анализ существующих решений | 29-14.10.15 |  |
|  | *…* |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  | Заключение |  |  |
|  | Список литературы |  |  |
|  | Приложения |  |  |

Обучающийся \_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*Подпись расшифровка подписи*

Руководитель \_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*Подпись расшифровка подписи*

Реферат

Текстовый документ 100 с., 35 рисунков, 10 таблиц, 4 главы, 6 приложений 35 источников.

БОЛЬШИЕ ДАННЫЕ, ГЕОАНАЛИТИКА, МАШИННОЕ ОБУЧЕНИЕ, ПРОМЫШЛЕННЫЙ КЛАСТЕР, СИСТЕМЫ ХРАНЕНИЯ ДАННЫХ, ТЕЛЕКОММУНИКАЦИИ, СОТОВАЯ СВЯЗЬ, БАЗОВАЯ СТАНЦИЯ, ЦЕНТРЫ ОБРАБОТКИ ДАННЫХ, BUSINESS INTELLIGENCE, ETL, MAP REDUCE, MACHINE LEARNING, HADOOP, SVM.

Объектом исследования являются обезличенные данные абонентов крупного поставщика услуг сотовой связи.

Цель работы – комплексный анализ данных сотового оператора, исследование путей наилучшего развития бизнеса компании, разработка программных продуктов для выгрузки показателей для единой системы отчетности, а также оценка эффективности полученной методики реализации алгоритмов формирования единовременных выгрузок на примерах задач геоаналитики и машинного обучения.

В процессе выполнения работы проводились анализ предметной области, исследование подходов решения задачи и выбор технических средств реализации. Для задачи геоаналитики были построены алгоритмы формирования вспомогательных справочников (единовременных выгрузок), реализованы программные модули для решения задачи. В рамках задачи машинного обучения построена и обучена модель принятия решения с использования метода опорных векторов.

В результате исследования для задачи геоаналитики реализованы алгоритмы получения справочников уникальных базовых станций для крупных городов России, справочника соседних базовых станций, сетки с зонами административного деления с привязкой к конкретным станциям. По задаче машинного обучения построена предиктивная модель определения пола абонента для всей большей части абонентской базы оператора.

Степень внедрения. В ходе работы над задачей геоаналитики были разработаны все основные и вспомогательные модули. Результат работы программ, а именно полученный отчет, соответствуют критериям качества со стороны компании-заказчика, все программные модули прошли тестирование и приняты как продуктивные задачи для автоматизированной работы на промышленном кластере.

Эффективность работы подтверждена экспертной группой аналитиков со стороны заказчика.

**Содержание**

[Введение 6](#_Toc454131824)

[1 Постановка задачи 11](#_Toc454131825)

[2 Анализ задачи 14](#_Toc454131826)

[2.1 Актуальность выбора технологий 14](#_Toc454131827)

[2.2 Анализ требований и подходов к проекту 18](#_Toc454131828)

[2.3 Анализ существующих решений. Подсистемы Hadoop 18](#_Toc454131829)

[2.4 Сравнение исполняемых сред 24](#_Toc454131830)

[2.5 Языки запросов к данным 29](#_Toc454131831)

[2.6 Анализ организации хранения данных в системе 29](#_Toc454131832)

[2.7 Анализ требований к отчетным формам задачи геоаналитики 31](#_Toc454131833)

[2.8 Справочник базовых станций 32](#_Toc454131834)

[2.9 Справочник покрытия БС 34](#_Toc454131835)

[2.10 Справочник соседних БС 38](#_Toc454131836)

[2.11 Справочник сетки 500 на 500 39](#_Toc454131837)

[2.12 Анализ источников входных данных 41](#_Toc454131838)

[2.13 Анализ архитектуры приложения 44](#_Toc454131839)

[3 Аппаратные и программные требования 46](#_Toc454131840)

[3.1 Программные средства для тестового окружения 48](#_Toc454131841)

[3.2 Программные средства реализации 50](#_Toc454131842)

[4 Реализация 51](#_Toc454131843)

[4.1 Реализация модуля формирования справочника уникальных базовых станций 51](#_Toc454131844)

[4.2 Реализация модуля формирования справочников покрытия, соседних базовых станций и отчета по сетке 500 на 500 55](#_Toc454131845)

[4.3 Реализация результирующего отчета «Монобренд» 62](#_Toc454131846)

[4.4 Реализация модуля по определению пола абонента 66](#_Toc454131847)

[4.5 Автоматизация модулей по формированию отчетов геоаналитики 70](#_Toc454131848)

[4.6 Анализ полученных результатов 71](#_Toc454131849)

[Заключение 76](#_Toc454131850)

[Список использованных источников 78](#_Toc454131851)

[Приложение А. HQL-скрипт для выгрузки эталонных данных 80](#_Toc454131852)

[Приложение В. Содержимое oozie-файла coordinator.xml 86](#_Toc454131853)

[Приложение Г. Содержимое oozie-файла coordinator.properties 92](#_Toc454131854)

[Приложение Д. Содержимое QGIS-файла Nizhniy\_NovgorodGrid.prj 95](#_Toc454131855)

[Приложение Е. Содержимое файла QGIS-файла Nizhniy\_NovgorodGrid.qpj 96](#_Toc454131856)

[Приложение Ж. Содержимое файла модели gender\_model.model 97](#_Toc454131857)

Введение

За последние несколько лет было создано более 90% объема мировой информации. Данные появляются из различных источников. Не так давно получил распространение термин «большие данные», обозначивший новую прикладную область – поиск способов автоматического быстрого анализа огромных объёмов разнородной информации.

Термин Big Data появился сравнительно недавно. Google Trends показывает начало активного роста употребления словосочетания начиная с 2011 года [1][2].

В общем смысле большие данные (англ. big data) – серия подходов, инструментов и методов обработки структурированных и неструктурированных данных огромных объёмов и значительного многообразия для получения воспринимаемых человеком результатов, эффективных в условиях непрерывного прироста, распределения по многочисленным узлам вычислительной сети, сформировавшихся в конце 2000-х годов, альтернативных традиционным системам управления базами данных и решениям класса Business Intelligence [3] [4].

Большими данными может оказаться что угодно: результаты научных экспериментов, история банковских транзакций, метеорологические наблюдения [5], профили в социальных сетях – словом, всё, что можно проанализировать и извлечь пользу из этого. Вот лишь несколько примеров того, что может быть источником данных, для которых необходимы методы работы с большими данными:

* + 1. Логи поведения пользователей в интернете
    2. GPS-сигналы от автомобилей для транспортной компании
    3. Данные, снимаемые с датчиков в большом адронном коллайдере
    4. Оцифрованные книги в Российской Государственной Библиотеке
    5. Информация о транзакциях всех клиентов банка
    6. Информация о всех покупках в крупной ритейл сети и т.д.

Непрерывный рост данных и увеличение скорости их генерации порождают проблему их обработки и хранения. Неудивительно, что тема больших данных (Big Data) является одной из самых обсуждаемых в современном ИТ-сообществе.

К задачам, решаемым методами Big Data относят:

* + 1. сбор данных;
    2. хранение данных (построение СХД);
    3. работа с данными (агрегация, анализ, описание);
    4. выявление взаимосвязей и построение трендов (возможно, с конечной целью предсказания).

Исходя из определения Big Data, можно сформулировать основные принципы работы с такими данными:

* + 1. Горизонтальная масштабируемость. Поскольку данных может быть сколь угодно много – любая система, которая подразумевает обработку больших данных, должна быть расширяемой. Если объём данных вырос в 2 раза, то для продолжения стабильной работы следует увеличить количество комплектующих в кластере в 2 раза. Реляционные базы данных маcштабируются достаточно сложно, как правило, в единой системе хранения данных (СХД). Есть горизонтальный порог масштабирования, после которого становится проще писать новую структуру, чем вводить очень сложные аппаратные комплексы.
    2. Отказоустойчивость. Принцип горизонтальной масштабируемости подразумевает, что машин в кластере может быть много. Это означает, что часть этих машин будет гарантированно выходить из строя. Методы работы с большими данными должны учитывать возможность таких сбоев и переживать их без каких-либо значимых последствий.
    3. Локальность данных. В больших распределённых системах данные распределены по большому количеству машин. Если данные физически находятся на одном сервере, а обрабатываются на другом – расходы на передачу данных могут превысить расходы на саму обработку. Поэтому одним из важнейших принципов проектирования BigData-решений является принцип локальности данных – по возможности обрабатываем данные на той же машине, на которой их храним.

Data Mining в отличие от Big Data – это класс программ для нестандартного анализа данных, который работает только со структурированными данными. Big Data – это стек технологий и архитектурный принцип, который решает задачи ETL неструктурированных данных и в дальнейшем может включать и анализ этих данных. Хотя ничего не мешает Big Data после ETL направить данные уже в структурированном виде в Data Mining [6].

Строго говоря, Big Data – это обработка огромных массивов данных, но, например, работа с базой данных под управлением СУБД Oracle на 20 Гб или 4 Пб – это ещё не Big Data, это просто высоконагруженная база данных или highload-БД. Ключевое отличие Big Data от обычных высоконагруженных систем состоит в возможности строить гибкие запросы. Реляционная база данных в силу своей архитектуры предназначена для коротких быстрых запросов, идущих однотипным потоком. Если разработчик решит выйти за пределы таких запросов и собрать новый сложный, то базу придётся переписывать – или же она не справится под нагрузкой [12].

Если углубиться в архитектуру, то можно увидеть, что традиционные базы данных хранят информацию очень дисперсионно. Например, номер абонента может быть на одном сервере в одной таблице, а его баланс — в другой таблице другого сервера. Быстродействие требует минимального разбиения данных. Как только приходится делать сложные join-запросы, производительность резко падает.

Big Data помогает и в борьбе с мошенничеством, и в увеличении продаж, и в планировании офисов. Анализ больших данных помогает блокировать мошенников быстрее, чем раньше: не за три дня, а за три минуты. Рассмотрим несколько кейсов.

Геоаналитика. Оператор связи может знать местоположение абонентов. Например, известно, что в такой-то зоне часто появляются люди с iPhone 6, возможно они пользуются стандартом связи 4G — значит, нужно построить в этой «соте» LTE-базовую станцию. В другом месте хватит 2G.

Еще пример: фирме по продаже средств мобильной связи стало интересно, когда люди переходят на современные смартфоны со старых телефонов. Оказалось, что есть достаточно чёткий порог: для жителя Москвы сочетание старого телефона 2007-го года и использования ряда сервисов вкупе с большим потреблением трафика может означать желание перейти на смартфон. Соответственно, увидев такую границу, люди посмотрели, на какие именно модели переходят абоненты, и пошли дальше – решили находить тех, кто готов прямо сегодня приобрести такой телефон. Предполагается, что в Москве есть около 250 офисов продаж. Команда аналитиков выделяет группу тех, кто в течение недели решит перейти на смартфон согласно полученным данным. Технический отдел и отдел маркетинга со своей стороны отслеживают, когда один из потенциальных покупателей подойдёт на 50 метров к салону связи, затем отправляет ему SMS с текстом рекомендацией зайти и приобрести устройство, если в данном салоне есть смартфон и если он готов к демонстрации. Традиционную систему такая задача просто упрощает на несколько порядков [13].

Рассмотрим следующий случай. Предположим, что у конкретного-лица имеются два устройства – телефон и планшет. Важно знать, что это один и тот же человек. Такая ситуация называется «мультидевайсом». Если две сим-карты постоянно регистрируются в одних и тех же секторах базовых станций – можно сделать вывод, что они принадлежат одному человеку, и предложить ему специальный тариф, который позволяет оплачивать услуги для разных сим-карт с одного счета. [14].

Данная работа посвящена анализу больших массивов данных на примере нескольких задач для крупной телекоммуникационной компании, поставщика услуг сотовой связи и широкополосного интернета, в рамках проекта Hadoop for BigData.

1. Постановка задачи

В данной работе рассматривается применение технологий Hadoop на примере задач геоаналитики и машинного обучения.

Одним из пунктов, рассматриваемых в рамках данной работы, является разработка программного продукта, решающего следующую задачу. Необходимо сформировать единовременную выгрузку, определив количество уникальных абонентов, которые находились в течение рабочего дня в каждой ячейке зоны разбиения города Нижний Новгород за период с 14 по 20 сентября 2015 года для оптимального расположения офисов монобренда.

Отчет формируется относительно всех зон разбиения, zone id, в дальнейшем именуемых для краткости zid. В поле total записывается число абонентов, удовлетворяющих условию: существует принадлежащая рассматриваемой зоне локация (набор сот), в которой абонент находился некоторое отличное от 0 секунд время в течение рассматриваемого интервала. Если в течение рассматриваемого интервала найдутся несколько зон, для которых это условие выполнено, то абонент приписывается к каждой из них. Таким образом, отчет показывает количество уникальных абонентов, прошедшее через каждую зону в течение рассматриваемого периода. Глубина периода анализа – с 8 до 22 часов. Прототип будущего отчета приведен в таблице 1 ниже:

Таблица 1 – Прототип результирующего отчета по задаче геоаналитики

| Измерение | Тип | Описание |
| --- | --- | --- |
| zid | string | Идентификационный номер зоны разбиения. |
| interval | string | Временной интервал. |
| 14/09 | integer | Общее количество уникальных абонентов в каждой ячейке зоны разбиения для каждого временного интервала 14 сентября 2015 г. |
| 15/09 | integer | Общее количество уникальных абонентов в каждой ячейке зоны разбиения для каждого временного интервала 15 сентября 2015 года. |
| 16/09 | integer | Общее количество уникальных абонентов в каждой ячейке зоны разбиения для каждого временного интервала 16 сентября 2015 года. |
| 17/09 | integer | Общее количество уникальных абонентов в каждой ячейке зоны разбиения для каждого временного интервала 17 сентября 2015 года. |
| 18/09 | integer | Общее количество уникальных абонентов в каждой ячейке зоны разбиения для каждого временного интервала 18 сентября 2015 года. |
| average\_weekdays | integer | Среднее количество уникальных абонентов в будние дни за 1 календарную неделю. |
| 19/09 | integer | Общее количество уникальных абонентов в каждой ячейке зоны разбиения для каждого временного интервала 19 сентября 2015 года. |
| 20/09 | integer | Общее количество уникальных абонентов в каждой ячейке зоны разбиения для каждого временного интервала 20 сентября 2015 года. |
| average\_weekend | integer | Среднее количество уникальных абонентов в выходные дни за 1 календарную неделю. |
| average\_week | integer | Среднее количество уникальных абонентов за 1 календарную неделю. |

Требованиями к решению поставленной задачи являются:

* + 1. Расчет справочника по сетке деления г. Нижний Новгород по квадратам 500 на 500 метров за сентябрь.
    2. Измерение ZID должно соответствовать справочнику по сетке деления г. Нижний Новгород на квадраты 500 на 500 метров и содержать все зоны разбиения.
    3. Показатели total, average\_weekdays, average\_weekend, average\_weekдолжны быть рассчитаны согласно алгоритму и содержать целое неотрицательное число не равное нулю.

Следующим пунктом является решение задачи машинного обучения по определению пола абонента. В настоящее время нет достоверных сведений о поле абонента, пользующегося телефонным номером. Для витрины в рамках отдельного проекта Data Eye необходимо сформировать показатель «пол» для каждого абонента активной абоненткой базы (ААБ) автоматически раз в месяц.

Система должна включать в себя:

* + 1. Модель определения пола абонента по набору признаков;
    2. Расчет точности определения пола для каждого абонента;
    3. Автоматизацию определения пола ААБ по расписанию: раз в месяц (1-5 числа каждого месяца);
    4. Результат разметки должен сохраняться как /user/tech\_arsspp\_bgd\_ms/spp/gender/merged\_gender.csv,
    5. Предыдущий расчет нужно удалять.

1. Анализ задачи


5. 1. Актуальность выбора технологий

В 1995 году исследовательская компания Gartner предложила hype cycle – кривую зрелости технологии, графически представляющую стадии, через которые проходит технологическое новшество в ходе своего становления [7].

Данный феномен наблюдается при появлении любой новой техники, будь то появление планшетов на рынке или внедрение новой CRM системы на предприятии. Про то, как эта кривая работает в части электроники, написано много статей.

Как можно видеть, кривая состоит из пяти фаз:

* + 1. «Запуск технологии» – первая фаза цикла: технологический прорыв, запуск проекта внедрения, который обещает желанные цели и решение многих проблем (хорошо если не всех)
    2. «Пик завышенных ожиданий» – общественный ажиотаж приводит к чрезмерному энтузиазму и нереалистичным ожиданиям. Успешное применение технологии возможно, но обычно неудач больше, чем успехов.
    3. «Нижняя точка разочарования» – технология не в состоянии соответствовать ожиданиям и быстро гасит энтузиазм. Начинают появляться разные «уважительные» причины, которые препятствуют ходу проекта.
    4. «Склон просвещения» – тут начинаются встречи, пересмотры некоторых идей или задач, корректировки хода проекта, иногда многие задачи, которые казались важными и нужными в начале, тут отметаются, но появляются смежные задачи, которые обнаруживаются в ходе проекта и решение которых дает больше преимущества для организации.
    5. «Плато производительности» – преимущества технологии становятся очевидными и признаются всеми. Технология стабильна и эволюционирует во второе и третье поколение. Окончательная высота плато зависит от того, насколько широко технология применяется.



Рисунок 2 – Цикл зрелости технологии (Hype cycle) Gartner

На самом деле данная кривая и ее разные варианты взяты из классической теории управления. График схож с обычной функцией мотивации [8].

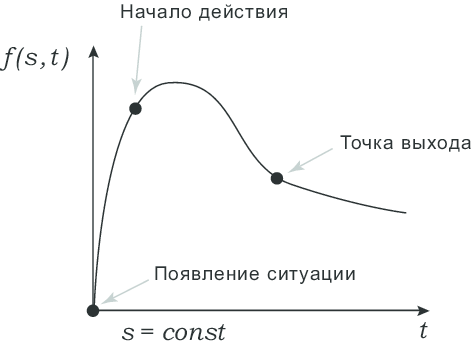


Рисунок 3 – График функции мотивации при *s* = *const*

Соответственно любой процесс или даже свойство всей системы (отдела, компании, отрасли) можно рассматривать как объект управления с управляющими воздействиями, планируемым значением (уровень продаж, степень мотивации), отклонениями и возмущающими воздействиями (вывод нового продукта конкурентами, увольнение сотрудника).

По результатам исследований аналитического агентства Gartner термин Big Data находился на пике популярности в 2013 году, о чем свидетельствует рисунок 4 [9].



Рисунок 4 – Кривая «зрелости» технологий по данным за 2013 год (Hype Cycle for Emerging Technologies, 2013)

В 2014 совокупность технологий Big Data уже находилась «склоне разочарований» [10] (рисунок 5).

Наконец, в 2015 года агентство Gartner заменило Big Data, и теперь там появился термин Machine Learning [11] (рисунок 6). Тем не менее, технология продолжает пользоваться популярностью и значимостью.



Рисунок 5 – Кривая «зрелости» технологий по данным за 2014 год (Hype Cycle for Emerging Technologies, 2014)



Рисунок 6 – Кривая «зрелости» технологий по данным за 2015 год (Hype Cycle for Emerging Technologies, 2015)

* 1. Анализ требований и подходов к проекту

У проекта Hadoop for BigData, предназначенного для решения определенного существующего круга задач, имеются три главные цели:

* + 1. поиск лучших решений для бизнеса;
    2. создание новых типов продуктов;
    3. совместное развитие экспертизы.

Отдельно стоит упомянуть три крупных цикла производства программного продукта, свойственных проекту:

1. «пилотная» фаза;
2. уточнение параметров;
3. автоматизация.

На стадии «пилота» проверяются основные предположения относительно цели продукта, закладываются базовые параметры алгоритма. В дальнейшем получившиеся данные становятся близки к реальным, но еще не в полной мере отражают желания бизнес-заказчика. На этой фазе допустимы различные эксперименты с данными и появление новых критериев.

В рамках фазы уточнения параметров по выделенным критериям проводятся дополнительные проверки, в ходе чего финализируется алгоритм. Полученные значения на данной ступени близки к совершенным, другими словами к 100% попаданию в цель. Все проверки осуществляются по заранее определенным наборам критериев.

На фазе автоматизации полученный в ходе первых двух стадий алгоритм включается в релиз и автоматизируется для бизнес-целей заказчика. Алгоритм может улучшаться в ходе продолжения исследований алгоритма.

* 1. Анализ существующих решений. Подсистемы Hadoop

Материалов по теории больших данных в специализированных журналах и на сайтах сегодня публикуется довольно много, но из теоретических публикаций далеко не всегда ясно, как можно использовать соответствующие технологии для решения конкретных практических задач.

Одним из самых известных и обсуждаемых проектов в области Big Data является Hadoop, разрабатываемый фондом Apache Software Foundation. Он представляет собой набор из свободно распространяемых утилит и библиотек для разработки и выполнения программ распределенных вычислений.

В результате уникальной стоимости бизнеса спрос на навыки в области больших данных растет быстрыми темпами. Об этой платформе и ее широком практическом применении пойдет речь в последующих главах настоящей работы.

Непосредственно сам проект Apache Hadoop, развиваемый сообществом разработчиков, обеспечивает и дополняет существующие технологии как модель платформы для консолидации, обработки и анализа больших объемов сложных данных. В состав проекта Hadoop входят следующие компоненты:

* + 1. Common – набор компонентов и интерфейсов для распределенных файловых систем и общего ввода-вывода;
    2. MapReduce – модель распределеных вычислений, предназначенная для параллельных вычислений над очень большими (до нескольких петабайт) объемами данных. Платформа ресурсов нового поколения называется YARN.
    3. HDFS – распределенная файловая система, работающая на больших кластерах типовых машин. В свою очередь включает в себя сервер имен (Name Node), вторичный сервер имен для репликации данных (Secondary Name Node), а также узел данных.

Классическая конфигурация кластера Hadoop состоит из одного сервера имён (Name Node), одного планировщика MapReduce-заданий, так называемого JobTracker, а также набора рабочих машин, на каждой из которых одновременно находятся сервер данных (DataNode) и сервер заданий (Task Tracker).

Узел NameNode представляет собой программный код, обычно выполняющийся на выделенной машине экземпляра HDFS. Этот узел отвечает за управление пространством имен файловой системы и за управление доступом со стороны внешних клиентов. Узел NameNode определяет соответствие между файлами и реплицированными на узлах DataNode блоками. При общепринятой тройной репликации вторая реплика хранится на отдельном узле, расположенном в той же стойке, а третья – на узле, расположенном в другой стойке. Обратите внимание на то, что для понимания этого материала вы должны обладать знаниями о кластерной архитектуре [15].

Реальные операции ввода/вывода не обращаются к узлу NameNode – через этот узел передаются только метаданные о сопоставлениях между узлами типа DataNode и файловыми блоками. Когда внешний клиент посылает запрос на создание файла, узел NameNode отвечает ему, посылая в ответ идентификационные данные файлового блока и IP-адрес узла DataNode, который будет хранить первую копию этого блока. Также узел NameNode информирует те узлы DataNode, которые будут получать копии данного файлового блока.

Узел NameNode хранит всю информацию о пространстве имен файловой системы в файле с именем FsImage. Этот файл вместе с журналом всех транзакций (файл с именем EditLog) хранится в локальной файловой системе узла NameNode. Файлы FsImage и EditLog также реплицируются с целью их защиты от повреждения или утраты в случае выхода из строя самого узла NameNode.

Узел DataNode также представляет собой программный код, обычно выполняющийся на выделенной машине экземпляра HDFS. Кластер Hadoop содержит один узел типа NameNode и сотни или тысячи узлов типа DataNode. Узлы DataNode обычно устанавливаются в стойки, в которых все они подключены к коммутатору. При использовании Hadoop предполагается, что сетевая пропускная способность между узлами одной стойки выше, чем между узлами, установленными в разных стойках.

Узлы DataNode обрабатывают запросы на чтение и запись, поступающие от клиентов файловой системы HDFS. Также они отвечают на команды создания, удаления и реплицирования блоков, полученные от узла NameNode. Узел NameNode получает периодические сообщения о состоянии (т. н. heartbeat-сообщения) от каждого узла DataNode. Каждое такое сообщение содержит отчет, из которого узел NameNode может получить информацию о сопоставлении блоков и о других метаданных файловой системы. Если узлу DataNode не удается отправить сообщение о состоянии, узел NameNode может предпринять корректирующие действия для реплицирования блоков, располагавшихся на отказавшем узле, на другие узлы кластера.

Ранее в Hadoop входили другие подпроекты, которые теперь являются самостоятельными продуктами Apache Software Foundation [16]:

* + 1. Avro – система сериализации для выполненных межъязыковых вызовов RPC и долгосрочного хранения данных;
    2. Pig – язык управления потоком данных и исполнительная среда для анализа больших объемов данных;
    3. Hive – распределенное хранилище данных; оно управляет данными, хранимыми в HDFS, и предоставляет язык запросов на базе SQL для работы с этими данными;
    4. HBase – нереляционная распределенная база данных;
    5. ZooKeeper – распределенный координационный сервис; предоставляет примитивы для построения распределенных приложений;
    6. Sqoop – инструмент для пересылки данных между структурированными хранилищами и HDFS;
    7. Oozie – сервис для записи и планировки заданий Hadoop.



Рисунок 7 – Набор компонентов, входящих в «экосистему» Hadoop

Сегодня Hadoop представляет собой сложную систему, состоящую из большого числа компонентов. Установить и настроить такую систему самостоятельно – весьма непростая задача. Поэтому многие компании сегодня предлагают готовые дистрибутивы Hadoop, включающие инструменты развертывания, администрирования и мониторинга.

Дистрибутивы Hadoop распространяются как под коммерческими (продукты таких компаний, как Intel, IBM, EMC, Oracle), так и под свободными (продукты компаний Cloudera, Hortonworks и MapR) лицензиями.

Cloudera Hadoop представляет собой полностью открытый дистрибутив. Он распространяется как в бесплатном, так и в платном варианте, известном под названием Cloudera Enterprise.

Cloudera Hadoop включает следующие основные компоненты:

* + 1. Cloudera Hadoop (CDH) – собственно дистрибутив Hadoop;
    2. Cloudera Manager – инструмент для развертывания, мониторинга и управления кластером Hadoop.



Рисунок 8 – Набор компонентов, предоставляемых дистрибутивами от Cloudera

Компоненты Cloudera Hadoop распространяются в виде бинарных пакетов, называемых парселами. По сравнению со стандартными пакетами и пакетными менеджерами парселы имеют следующие преимущества:

* + 1. Простота загрузки: каждый парсел представляет собой один файл, в котором объединены все нужные компоненты.
    2. Внутренняя согласованность: все компоненты внутри парсела тщательно протестированы, отлажены и согласованы между собой, поэтому вероятность возникновения проблем с несовместимостью компонентов очень мала.
    3. Разграничение распространения и активации: можно сначала установить парселы на все управляемые узлы, а затем активировать их одним действием; благодаря этому обновление системы осуществляется быстро и с минимальным простоем.
    4. Обновления «на ходу»: при обновлении минорной версии все новые процессы (задачи) будут автоматически запускаться под этой версией, уже запущенные задачи продолжат исполняться в старом окружении до своего завершения. Однако обновление до более новой мажорной версии возможно только посредством полного перезапуска всех сервисов кластера, и соответственно всех текущих задач.
    5. Простой откат изменений: при возникновении каких-либо проблем в работе с новой версией CDH ее можно легко откатить до предыдущей.

Начиная со второй версии Hadoop, исполняемую среду MapReduce, обеспечивающую управление ресурсами и обработку данных, заменили на Map Reduce v2 – YARN (Yet Another Negotiation) – еще один посредник между HDFS и инструментами для работы с данными. Различия между версиями приведены на рисунке:



Рисунок 9 – Различия версий Hadoop 1.0 и 2.0

* 1. Сравнение исполняемых сред

MapReduce – это модель распределенной обработки данных, предложенная компанией Google для обработки больших объёмов данных на компьютерных кластерах [17].

MapReduce предполагает, что данные организованы в виде некоторых записей. Обработка данных происходит в 3 стадии:

* + 1. Стадия Map. На этой стадии данные предобрабатываются при помощи функции map(), которую определяет пользователь. Работа этой стадии заключается в предобработке и фильтрации данных. Функция map() примененная к одной входной записи и выдаёт множество пар ключ-значение. Что будет находиться в ключе и в значении – решать пользователю, но ключ – очень важная вещь, так как данные с одним ключом в будущем попадут в один экземпляр функции reduce.
    2. Стадия Shuffle. Проходит незаметно для пользователя. В этой стадии вывод функции map «разбирается по корзинам» – каждая корзина соответствует одному ключу вывода стадии map. В дальнейшем эти корзины послужат входом для reduce.
    3. Стадия Reduce. Каждая «корзина» со значениями, сформированная на стадии shuffle, попадает на вход функции reduce(). Функция reduce задаётся пользователем и вычисляет финальный результат для отдельной «корзины». Множество всех значений, возвращённых функцией reduce(), является финальным результатом MapReduce-задачи.

Несколько дополнительных фактов про MapReduce:

* + 1. Все запуски функции map работают независимо и могут работать параллельно, в том числе на разных машинах кластера.
    2. Все запуски функции reduce работают независимо и могут работать параллельно, в том числе на разных машинах кластера.
    3. Shuffle внутри себя представляет параллельную сортировку, поэтому также может работать на разных машинах кластера. Пункты 1-3 позволяют выполнить принцип горизонтальной масштабируемости.
    4. Функция map, как правило, применяется на той же машине, на которой хранятся данные – это позволяет снизить передачу данных по сети (принцип локальности данных).
    5. MapReduce – это всегда полное сканирование данных, никаких индексов нет. Это означает, что MapReduce плохо применим, когда ответ требуется очень быстро.

Обработка данных подразделяется на следующие этапы (последовательность представлена на рисунке):

* + 1. Запуск приложения: передача кода приложения на главный (master) и подчиненные узлы (workers).
    2. Мастер назначает конкретные задачи map или reduce и распределяет части входных данных по вычислительным узлам (workers).
    3. Map-узлы читают назначенные им входные данные и начинают их обработку.
    4. Map-узлы локально сохраняют промежуточные результаты: каждый узел сохраняет полученный результат на локальные диски.
    5. Reduce-узлы читают промежуточные данные с Map-узлов и выполняют Reduce обработку данных.
    6. Reduce-узлы сохраняют итоговые результаты в выходные файлы, обычно в HDFS.

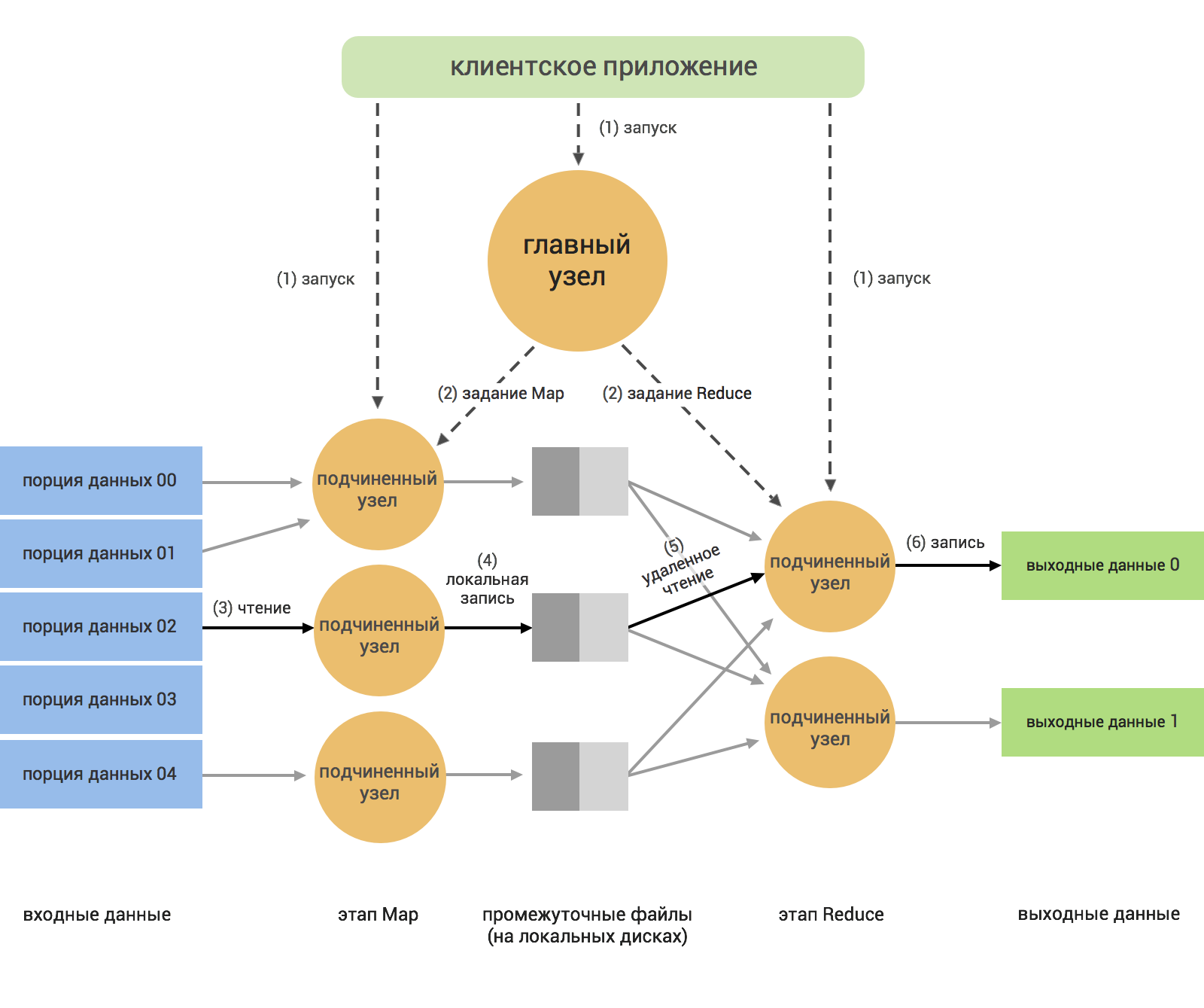


Рисунок 10 – Последовательность выполнения MapReduce-задания

Создание приложений для MapReduce достаточно трудоемко, написание всех функций, компилирование, тестирование и упаковка занимают много времени.

Apache Spark – это платформа параллельной обработки, которая поддерживает обработку в памяти и тем самым значительно повышает производительность аналитических приложений, работающих с большими данными. Spark работает для SQL, потока данных и машинного обучения.

Проект Apache Spark в отличие от MapReduce использует идею локальности данных, однако выносит большинство вычислений в память вместо диска [18]. Ключевым понятием в Spark является Resilient Distributed Dataset (далее RDD) – указатель на ленивую распределённую колекцию данных. Большинство операций над RDD не приводит к каким-либо вычислениям, а только создаёт очередную обёртку, обещая выполнить операции только тогда, когда они понадобятся.

Ниже приведён скрипт на Python (Spark поддерживает интерфейсы для Scala, Java и Python) для решения задачи фильтрации серверных сообщений из журналов [<https://habrahabr.ru/post/240405/>]:

Листинг 1. Пример обработки текстовых файлов

# создаём контекст (SparkContext)

sc = ...

# создаём указатель на данные

rdd = sc.textFile("/path/to/server\_logs")

# разбираем строки и переводим их в удобный формат.

# Далее фильтруем записи без ошибок и сохраняем результаты на диск

rdd.map(parse\_line).filter(contains\_error).saveAsTextFile("/path/to/result")

В этом примере реальные вычисления начинаются только на последней строке. Spark видит, что нужно материализовать результаты, и для этого начинает применять операции к данным. При этом здесь нет никаких промежуточных стадий, каждая строчка кладется в память, разбирается, проверяется на признак ошибки в сообщении и, если такой признак есть, тут же записывается на диск.   
Такая модель оказалась настолько эффективной и удобной, что проекты из экосистемы Hadoop начали один за другим переводить свои вычисления на Spark, а над самим проектом сейчас работает больше людей, чем над устаревшим MapReduce.

Компания Hortonworks решила сделать упор на альтернативный движок Tez. Он представляет задачу в виде направленного ациклического графа (DAG) компонентов-обработчиков. Планировщик запускает вычисление графа и при необходимости динамически переконфигурирует его, оптимизируя под данные. Это очень естественная модель для выполнения сложных запросов к данным, таких как SQL-подобные скрипты в Hive, где Tez приносит ускорение до 100 раз. Впрочем, кроме как в связке с Hive, Tez пока мало где используется, поэтому сказать, насколько он пригоден для более простых и распространённых задач, довольно сложно.

* 1. Языки запросов к данным

Hive – самая первая и до сих пор одна из самых популярных СУБД на платформе Hadoop. В качестве языка запросов используется HiveQL –диалект SQL, который, тем не менее, позволяет выполнять довольно сложные запросы над данными, хранимыми в HDFS. Кроме того, как Hortonworks, так и Cloudera предоставляют ODBC-драйвера, позволяя подключить к Hive Microsoft Excel.

Impala – продукт компании Cloudera и основной конкурент Hive. В отличие от последнего, Impala никогда не использовала классический MapReduce, а изначально исполняла запросы на своём собственном движке, написанном на C++. Кроме того, в последнее время Impala активно использует кеширование часто используемых блоков данных и колоночные форматы хранения, что очень хорошо сказывается на производительности аналитических запросов. Так же, как и для Hive, Cloudera предлагает к своему продукту вполне эффективный ODBC-драйвер.

Spark SQL – новая ветвь развития SQL на базе Spark. В Spark SQL нет отдельной консоли и своего хранилища метаданных, SQL-парсер пока довольно слабый, а партиции вовсе не поддерживаются. На данный момент его основная цель — уметь читать данные из сложных форматов, таких как Parquet, и выражать логику в виде моделей данных, а не программного кода. Очень часто конвеер обработки состоит из чередующихся SQL-запросов и программного кода. Spark SQL позволяет связать эти стадии.

* 1. Анализ организации хранения данных в системе

В основе инфраструктуры Hadoop расположена файловая система Hadoop Distributed File System (далее HDFS). Обычная файловая система, по большому счёту, состоит из таблицы файловых дескрипторов и области данных. В HDFS вместо таблицы используется специальный сервер – сервер имён (далее Name Node), а данные хранятся на сервере данных (далее Data Node).

В остальном отличий не так много: данные разбиты на блоки (обычно по 64Мб или 128Мб), для каждого файла сервер имён хранит его путь, список блоков и их реплик. HDFS имеет классическую unix-подобную древовидную структуру директорий, пользователей с набором прав, а также схожий набор консольных команд:

Листинг 2. Использование команд Hadoop для работы с файловой системой

# просмотреть корневую директорию: локально и на HDFS

ls /

hadoop fs -ls /

# оценить размер директории

du -sh mydata

hadoop fs -du -s -h mydata

# вывести на экран содержимое всех файлов в директории

cat mydata/\*

hadoop fs -cat mydata/\*

Основным хранилищем данных для будущей системы является распределенная файловая система HDFS. Источники, располагающиеся в ней, представляют собой сырой массив данных, которые извлекаются, преобразовываются, загружаются и хранятся в одном месте, на промышленном кластере. Это могут быть события сети (вызовы, обрывы, удержания звонков и т.п.), геолокационные данные, данные Customer Relationship Management (далее CRM), данные биллинга, данные о пополнении счета и так далее. Схема промышленного кластера приведена на рисунке:



Рисунок 11 – Структура промышленного кластера для продуктивных задач

* 1. Анализ требований к отчетным формам задачи геоаналитики

Для успешного решения поставленной задачи необходимо предварительно провести подробный анализ отчетных форм и определить, какие данные они должны содержать.

Разработку конечного программного продукта можно разделить на несколько взаимосвязанных этапов по формированию отдельных вспомогательных справочников:

1. базовых станций;
2. справочник покрытия БС;
3. соседних базовых станций;
4. сетки административного деления 500 на 500.

Входными источниками являются:

* + 1. справочник с координатами полигонов для Нижнего Новгорода;
    2. версионный справочник по секторам базовых станций DIM\_CRAMER\_OSS\_SECTORS по регионам;
    3. версионный справочник позиций базовых станций DIM\_BS\_POSITION по регионам;
    4. справочник с сетками городов. Размер ячейки в сетке – 500 на 500.
  1. Справочник базовых станций

Базовая станция (БС) – комплекс радиопередающей аппаратуры (ретрансляторы, приемопередатчики), осуществляющий связь с конечным абонентским устройством – сотовым телефоном.

Location Area Code (далее LAC) – код зоны местоположения группы базовых станций.

Cell ID – идентификатор конкретной соты внутри Location Area.

IMSI – уникальный идентификационный номер sim-карты.

БС может быть описана при помощи комбинации LAC и Cell ID.

Справочник базовых станций рассчитывается для каждого города из списка: Волгоград, Москва, Московская область, Воронеж, Казань, Красноярск, Нижний Новгород, Новосибирск, Омск, Ростов, Санкт-Петербург, Уфа, Самара. Для этого необходимо объединить файлы DIM\_CRAMER\_OSS\_SECTORS со справочником DIM\_BS\_POSITION по значению атрибута POSITION\_CODE:

Таблица 2 – Соответствие значения поля POSITION\_CODE регионам

| № | GEOREGION\_NAME | POSITION\_CODE | Наименование латиницей |
| --- | --- | --- | --- |
| 1 | Волгоград | 7264 | volgograd |
| 2 | Воронеж | 7171 | voronezh |
| 3 | Казань | 8911 | kazan |
| 4 | Красноярск | 8668,307733, 307735 | krasnoyarsk |
| 5 | Нижний Новгород | 5971 | nizhniy\_novgorod |
| 6 | Новосибирск | 5837 | novosibirsk |
| 7 | Омск | 5802 | omsk |
| 8 | Ростов-на-Дону | 5593 | rostov\_na\_donu |
| 9 | Санкт-Петербург | 599,194844 | saint\_petersburg |
| 10 | Уфа | 11152 | ufa |
| 11 | Самара | 5497 | samara |
| 12 | Москва | 601,186351,  6512168,186347,  367230,186349 | moscow |
| 13 | Москва и МО | 12 | moscow\_mo |

Таблица 3 – Атрибуты выходных данных справочника БС

| Показатель | Описание измерения | Формат данных | Алгоритм расчета |
| --- | --- | --- | --- |
| POSITION\_LATITUDE | Географическая широта | Вещественное число | Определяется из справочника позиций базовых станций |
| POSITION\_LONGITUDE | Географическая долгота | Вещественное число | Определяется из справочника позиций базовых станций |
| CELL | ID соты (номер вышки). | Целое неотрицательно число | Определяется из справочника свечения по секторам |
| AZIMUTH | Направление свечения вышки | Вещественное число | Определяется из справочника свечения по секторам |
| SECTOR\_ANGLE | Угол свечения вышки | Вещественное число | Определяется из справочника свечения по секторам |
| RANGE | Cтандарт связи. Диапазон (900, 1800, 2100 МГц) | Целое неотрицательное число | Определяется из справочника свечения по секторам |

Получаем справочники БС с названием CITY\_DIM\_BASE\_STATIONS.csv, где CITY – название города, по которому произведен расчет. Разделитель полей – символ табуляции с кодом «\t».

* 1. Справочник покрытия БС

Определяем «высоту» для БС из справочника CITY\_DIM\_BASE\_STATIONS.csv:

* + 1. Если стандарт связи составляет 900 МГц, то радиус равен 1 км.
    2. Если стандарт связи – 1800 МГц, то радиус – 0,75 км. Если стандарт связи – 2100, то радиус – 0,25 км.
    3. Если стандарт связи принимает пустое значение или -99, то радиус равен 1 км.

На следующем этапе необходимо определить площадь свечения БС. Наносим координаты каждой БС на карту. Координаты БС (значения полей POSITION\_LATITUDE*,* POSITION\_LONGITUDE) берутся из справочника CITY\_DIM\_BASE\_STATIONS.csv.

От получившейся точки откладываем прямую по направлению свечения БС (значение поля AZIMUTH из справочника CITY\_DIM\_BASE\_STATIONS.csv). Затем от получившейся прямой откладываем угол, равный углу свечения БС (значение поля SECTOR\_ANGLE из справочника CITY\_DIM\_BASE\_STATIONS.csv). В получившийся угол вписываем окружность с радиусом равным значению, соответствующему стандарту связи.

Если для БС угол свечения строго больше 180, то около вышки описываем окружность радиусом 1 км, затем данную окружность вписываем в треугольник. По получившейся окружности определяем основание треугольника.

Определяем дальность свечения БС. Дальность свечения БС равна высоте треугольника. Если в справочнике по секторам БС – DIM\_CRAMER\_OSS\_SECTORS – отсутствуют азимут и угол свечения, то вместо площади свечения будет использоваться круг. Направление свечения БС равно 360, угол свечения БС равен 360 градусам.

Получаем справочники секторов с названием CITY\_DIM\_TRIANGLES\_WKT.csv, где CITY – название города (кроме Москвы и МО), по которому необходим расчет (в нашем случае – это Нижний Новгород). Разделитель в справочнике – символ «;».

Таблица 4 – Атрибуты выходных данных справочника покрытия БС

| Показатель | Описание показателя | Формат данных | Алгоритм расчета |
| --- | --- | --- | --- |
| LAC | Номер группы вышек. | Целое неотрицательное число | Определяется из справочника по секторам базовых станций |
| CELL | ID соты (номер вышки). | Целое неотрицательное число | Определяется из справочника по секторам базовых станций |
| LATITUDE\_LEFT | Географическая широта | Вещественное неотрицательное число | – |
| LONGITUDE\_LEFT | Географическая долгота | Вещественное неотрицательное число | – |
| LATITUDE\_RIGHT | Географическая широта | Вещественное неотрицательное число | – |
| LONGITUDE\_RIGHT | Географическая долгота | Вещественное неотрицательное число | – |
| LATITUDE\_BS | Координата (географическая широта) вершины треугольника, где установлена БС | Вещественное неотрицательное число | – |
| LONGITUDE\_BS | Координата (географическая долгота) вершины треугольника, где установлена БС | Вещественное неотрицательное число | – |
| LATITUDE\_CENTER | Координата (географическая широта) центр треугольника, где установлена БС | Вещественное неотрицательное число | – |
| LONGITUDE\_CENTER | Координата (географическая широта) центр треугольника, где установлена БС | Вещественное неотрицательное число | – |
| AZIMUTH | Направление свечения вышки | Вещественное неотрицательное число | Определяется из справочника свечения по секторам |
| ANGLE | Угол свечения вышки | Вещественное неотрицательное число | Определяется из справочника свечения по секторам |
| RANGE | Диапазон рабочих частот | Целое неотрицательное число | Определяется из справочника свечения по секторам |
| HEIGHT | Высота треугольника | Вещественное неотрицательное число | – |
| the\_geom\_EPSG:3576 | Точки треугольника в формате WKT и системе координат EPSG:3576 | – | – |

* 1. Справочник соседних БС

На сетку из справочника с сетками городов наносим покрытие БС (в виде треугольника), получившееся в результате расчета предыдущих вычислений DIM\_TRIANGLE\_WKT.csv. Если площади треугольников вышек пересекаются, то они являются соседними.

Строим справочник вида:

Листинг 3. Вид справочника соседних БС

Lac#cell\_id <разделитель> Lac1#cell\_id1, Lac2#cell\_id2, Lac3#cell\_id3, …

//Lac#cell\_id – вышка, для которой мы определяем соседей; //Lac1#cell\_id1, Lac2#cell\_id2, Lac3#cell\_id3 – соседние БС, //для вышки Lac#cell\_id.

Название справочника – CITY\_DIM\_NEIGHBORS.csv, где CITY – название города, по которому возможен расчет.

Ниже приведена таблица с описанием атрибутов выходных данных справочника соседних БС.

Таблица 5 – Атрибуты выходных данных справочника соседних БС

| Показатель | Описание показателя | Формат данных | Алгоритм расчета |
| --- | --- | --- | --- |
| LAC | Номер группы вышек. | Вещественное число | Определяется из справочника по секторам базовых станций |
| CELL | ID соты (номер вышки). | Вещественное число | Определяется из справочника по секторам базовых станций |
| LAC\_NEIGHBOUR | Номер группы соседних вышек. | Вещественное число | Определяется из справочника с площадью секторов, построенного по алгоритму определения площади свечения вышки на ячейки сетки |
| CELL\_NEIGHBOUR | ID соседней соты (номер вышки). | Вещественное число | Определяется из справочника с площадью секторов, построенного по алгоритму определения площади свечения вышки на ячейки сетки |

* 1. Справочник сетки 500 на 500

Используем заготовленные сетки по 12 городам 500х500 в виде файлов с расширением *.*shp – CITY\_GRID\_500.shp, где CITY – название города.

На сетку из справочника города наносим покрытие БС в виде треугольника, получившееся в результате расчета DIM\_TRIANGLE\_WKT.csv. Далее определяем на какие ячейки сетки светит данная БС.

На рисунке 2 приведен пример расположения БС и ее свечения на соседние квадраты сетки города.



Рисунок 12 – Расположение БС и засветы на квадраты

Рассчитываем площадь свечения вышки в каждой ячейке сетки (2, 3, 4, 6, 7, 8, 10, 11, 12) и заносим в справочник DIM\_GRID\_500.csv.

Таблица 6 – Выходной формат данных справочника DIM\_GRID\_500.csv

|  |  |  |  |
| --- | --- | --- | --- |
| Показатель | Описание показателя | Формат данных | Алгоритм расчета |
| ID\_SQUARE | Идентификатор квадрата, на которую светит данная вышка | Целое неотрицательно число | Определяется из справочника с координатами полигонов |
| LAC | Номер группы вышек. | Целое неотрицательно число | Определяется из справочника по секторам базовых станций |
| CELL\_ID | ID соты (номер вышки). | Целое неотрицательно число | Определяется из справочника по секторам базовых станций |
| AREA\_GLARE | Площадь засвета квадрата, получившегося в результате пересечения квадрата и сектора вышки | Вещественное число | – |
| AREA\_COVERAGE | Площадь фигуры полученной в результате пересечения треугольника (покрытия БС) и квадрата (ячейки сетки) | Вещественное число | – |

* 1. Анализ источников входных данных

Наиболее важными при решении поставленных задач являются такие таблицы, как:

* + 1. Трафик Anritsu. Содержит информацию о разобранных событиях сигнальной сети, включая следующие данные: звонки, смс, геолокацию БС. Представляет собой сжатые файлы в формате Sequence File для хранения в HDFS [1]. Источник содержит информацию по транзакциям в сетях 2G и 3G. Периодичность и время появления данных – каждые 15 мин. Средний объем поступающих данных в день – 55 Гб.
    2. Трафик Polystar. Схож по содержанию с Anritsu за исключением того, что включает в себя больше событий, больше регионов, а также ошибки сети. Средний объем поступающих данных в день – 12 Гб.

Ниже приведена таблица 7 с указанием номеров полей в каждом из источников.

Таблица 7 – Индексы полей источников Anritsu и Polystar

| Название поля | Номер поля в источнике Anritsu/ADR | Номер поля в источнике Anritsu/TDR |
| --- | --- | --- |
| Datetime | 2 | 2 |
| Imeisv | 53 | 70 |
| Imsi | 16 | 68 |
| Tmsi | 17 | 71 |
| Lac | 18 | 25 |
| Cell\_id | 19 | 84 |
| Event | 1 | 1 |
| Causes | 33 | 32 |
| Standart | 2G | 3G |
| Num2 | 06 или 07 | 73 или 74 |
| OPC/DPC | 4 или 5 | 8 или 9 |
| Сauses\_calls | 37 | 54 |
| LOC\_UPD\_TYPE | 56 | 43 |
| WAIT\_TIME | 24 | 47 |
| CALL\_TYPE | 6 | 51 |
| DISCONNECTION\_CAUSE | 49 | 52 |
| IMEI | 15 | 69 |
| TMSI\_OLD | 54 | 72 |
| CONV\_TIME | 25 | 77 |
| LAC\_OLD | 46 | 82 |
| RANAP\_MESSAGE\_FLAG1 | – | 86 |

* 1. Анализ архитектуры приложения



Рисунок 13 – Диаграмма развертывания на dev-кластере

Файл содержащий текст в формате сsv. Разделитель – символ «;». Данный выбор обусловлен дальнейшим использованием файлов редактором электронных таблиц Microsoft Excel, а также текстовым редактором EmEditor, поддерживающим открытие и редактирование файлов очень больших объемов, которые не удается открыть с помощью Excel.

3. Аппаратные и программные требования

В блоге Cloudera приводится следующий список аппаратных конфигураций для различных вариантов загрузки [19]:

* + 1. легкая конфигурация (1U) – 2 шестиядерных процесссора, 24-64 Гб памяти, 8 жестких дисков емкостью 1-2 Тб;
    2. рациональная конфигурация (1U) – 2 шестиядерных процессора, 48-128 Гб памяти, 12-16 жестких дисков (1 или 2 Тб), подключенных напрямую через контроллер материнской платы;
    3. тяжелая конфигурация для хранилищ (2U): 2 шестиядерных процессора, 48-96 Гб памяти, 16-24 жестких дисках. При множественных сбоях в работе узлов в данной конфигурации происходит резкое увеличение сетевого трафика;
    4. конфигурация для интенсивных вычислений: 2 шестиядерных процессора, 64-512 Гб памяти, 4-8 жестких дисков емкостью 1-2 Тб.

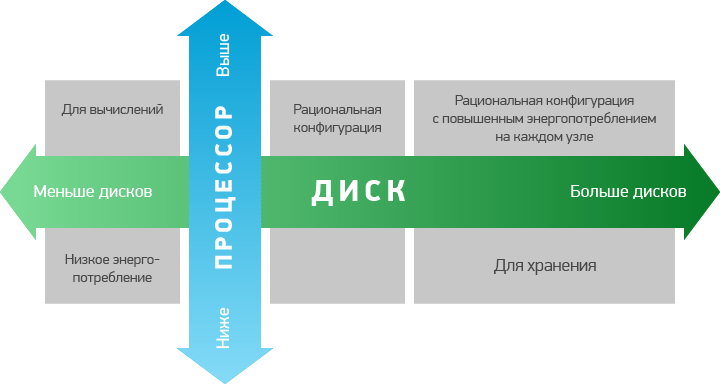


Рисунок 14 – Выбор конфигурации в зависимости от аппаратного оснащения

Классическая конфигурация кластера Hadoop состоит из одного сервера имён (Name Node), одного мастера MapReduce, так называемого JobTracker, а также набора рабочих машин, на каждой из которых одновременно крутится сервер данных (DataNode) и сервер заданий (TaskTracker).

В рамках данной работы возможно использование QuickStart VM от Cloudera с Single-Node Apache Hadoop Cluster с примерами для упражнений, программными скриптами, а также Cloudera Manager для управления кластером. Обязательным условием является наличие всех необходимых справочников, источников и таблиц, предоставляемых на вход создаваемым программам, анализ которых приведен в главе «Анализ задачи».

К аппаратному оснащению, а именно процессору, оперативной памяти (ОЗУ) и ПЗУ с учетом использования технологии виртуализации предъявляется следующее требования:

* + 1. двухъядерный либо четырехъядерный 64-битный процессор на базе Intel с поддержкой аппаратной виртуализации VT (Virtualization Technology) или AMD-V.
    2. не менее 20 Гб (гигабайт) свободного места на жестком диске.

Минимальная рекомендуемая оперативная память для работы с помощью средств виртуализации в зависимости от версии CDH отражена в таблице 8.

Таблица 8 – Аппаратные требования дистрибутивов CDH

|  |  |
| --- | --- |
| Версия CDH и Cloudera Manager | Требуемое количество оперативной памяти основной ОС для обеспечения виртуализации |
| CDH 5 (по умолчанию) | 4 Гб и более |
| Cloudera Express | 8 Гб и более |
| Cloudera Enterprise (пробный период) | 10 Гб и более |

1. 1. Программные средства для тестового окружения

Для обеспечения стабильной работы используется только 64-разрядная основная операционная система с поддержкой средств виртуализации [20] [21]. Это могут быть:

1. Microsoft Windows 7/8/10;
2. Mac OS X 10.10 или выше;
3. Ubuntu 14.04 и выше;
4. CentOS 6 и выше.

В качестве виртуальной машины возможно использование одного из нижеперечисленных программных продуктов:

1. VirtualBox – программный продукт виртуализации для операционных систем Microsoft Windows, Linux, Mac OS X и других.
2. VMware WorkStation версии 8 (и выше) с предустановленными VMware Player версии 4 (и выше) для Windows и Linux, а также VMware Fusion 4 (и выше) для Mac OS X. Нужно учесть, что VMware Fusion работает лишь под управлением Intel-архитекутры, таким образом ранние компьютеры с Mac OS с процессорами PowerPC не запустят предлагаемые дистрибутивы QuickStart VM.
3. KVM – программное решение, обеспечивающее виртуализацию в среде Linux на платформе x86, которая поддерживает аппаратную виртуализацию на базе Intel VT (Virtualization Technology) либо AMD SVM (Secure Virtual Machine).

Исходя из вышеперечисленных требований, предъявляемых со стороны средств виртуализации и имеющейся установленной операционной системы Microsoft Windows 7 в рамках данной работы для тестов была использована виртуальная машина от Oracle – VirtualBox, а вместе с ней файл VMDK-образа Cloudera QuickStart VM версии 5.4.2.

Было замечено, что иногда при работе с VirtualBox у некоторых пользователей наблюдается «паника» ядра при запуске файла виртуальной ОС CentOS 6.4. В таком случае рекомендуется открыть пункт основного меню «Настройки» (Settings), далее в списке выбрать «Система» (System) и переместиться на вкладку «Материнская плата» (Motherboard). Отметить пункт ICH9 вместо PIIX3 для чипсета; также стоит упомянуть про необходимость установки флажка напротив «Включить I/O APIC» на этой же вкладке.

Для загрузки большого количества файлов или файлов большого объема предпочтительнее копировать файлы в HDFS с помощью утилиты hadoop. Эта операция осуществляется при помощи следующей команды (листинг 4), выполняемой с любого сервера, входящего в HDFS-кластер.

Листинг 4. Команда put для копирования файлов HDFS

hadoop fs -put file\_for\_hadoop /path/to/put/file/in/HDFS/

Самый простой способ переноса данных в кластер — это копирование файлов через веб-интерфейс файлового менеджера в панели управления Hue.

Hue – это набор веб-приложений, используемых для взаимодействия с кластером. Только кластеры Linux.

Веб-интерфейс расположен по адресу <http://[HueNode]:8888/filebrowser/>

Вместо [HueNode] указывается адрес узла, на котором развернут Hue. Он интуитивно понятен и не требует дополнительных пояснений.



Рисунок 15 – Веб-интерфейс Hue для доступа к ресурсам кластера

* 1. Программные средства реализации

С учетом всех факторов, в качестве средств реализации были выбраны следующие программные продукты и технологии:

1. Язык программирования Java (JDK 1.7);
2. Apache Maven – фреймворк для сборки проектов;
3. HiveQL – диалект SQL, который позволяет выполнять запросы над данными, хранимыми в HDFS.
4. Apache Oozie – планировщик потоков задач. Спроектирован для объединения отдельных MapReduce задач в единый конвейер и запуска их по расписанию. Дополнительно может выполнять Hive, Java и консольные действия.
5. Dell Toad for Apache Hadoop – инструмент для настройки и получения доступа к кластеру.
6. Реализация

В ходе анализа была изучена предметная область, рассмотрена архитектура будущей системы, были установлены все требования, предъявляемые к разрабатываемым модулям, а также выбраны способы и средства их реализации. Данный раздел описывает основные особенности их непосредственной разработки.

1. 1. Реализация модуля формирования справочника уникальных базовых станций

Разработку модуля условно можно разделить на 3 этапа (по количеству stage-пакетов):

1. фильтрация данных источника Anritsu (пакет stage1);
2. фильтрация данных источника Polystar (пакет stage2);
3. соединение справочников DIM\_BS\_POSITION и DIM\_CRAMER\_OSS\_SECTORS (stage3).

Для первого этапа разработаны классы MImsiTmsi, RImsiTmsi, DImsiTmsi. Диаграмма классов представлена на рисунке 16:

Рисунок 16 – Диаграмма классов пакета stage1 модуля unique-bs-polystar-imsitmsi

Класс-mapper – MImsiTmsi – наследует класс Mapper из библиотеки org.apache.hadoop.mapreduce.Mapper. Класс-reducer – RImsiTmsi – наследует класс Reducer из библиотеки org.apache.hadoop.mapreduce.Reducer.

На втором этапе задействованы классы MPolystar, RPolystar, DPolystar, представленные на рисунке 17.



Рисунок 17 – Диаграмма классов пакета stage2 модуля unique-bs-polystar-imsitmsi

На третьем этапе созданы классы BsPosition, CramerOssSectors (описывающие спецификацию источников), а также 2 mapper-класса – MDimBsPosition, MDimCramerOssSectors, и класс-reducer – RJoinDim. Разработанные классы приведены в соответствии со спецификацией UML на следующем рисунке 18:



Рисунок 18 – Диаграмма классов пакета stage3 модуля unique-bs-polystar-imsitmsi

* 1. Реализация модуля формирования справочников покрытия, соседних базовых станций и отчета по сетке 500 на 500

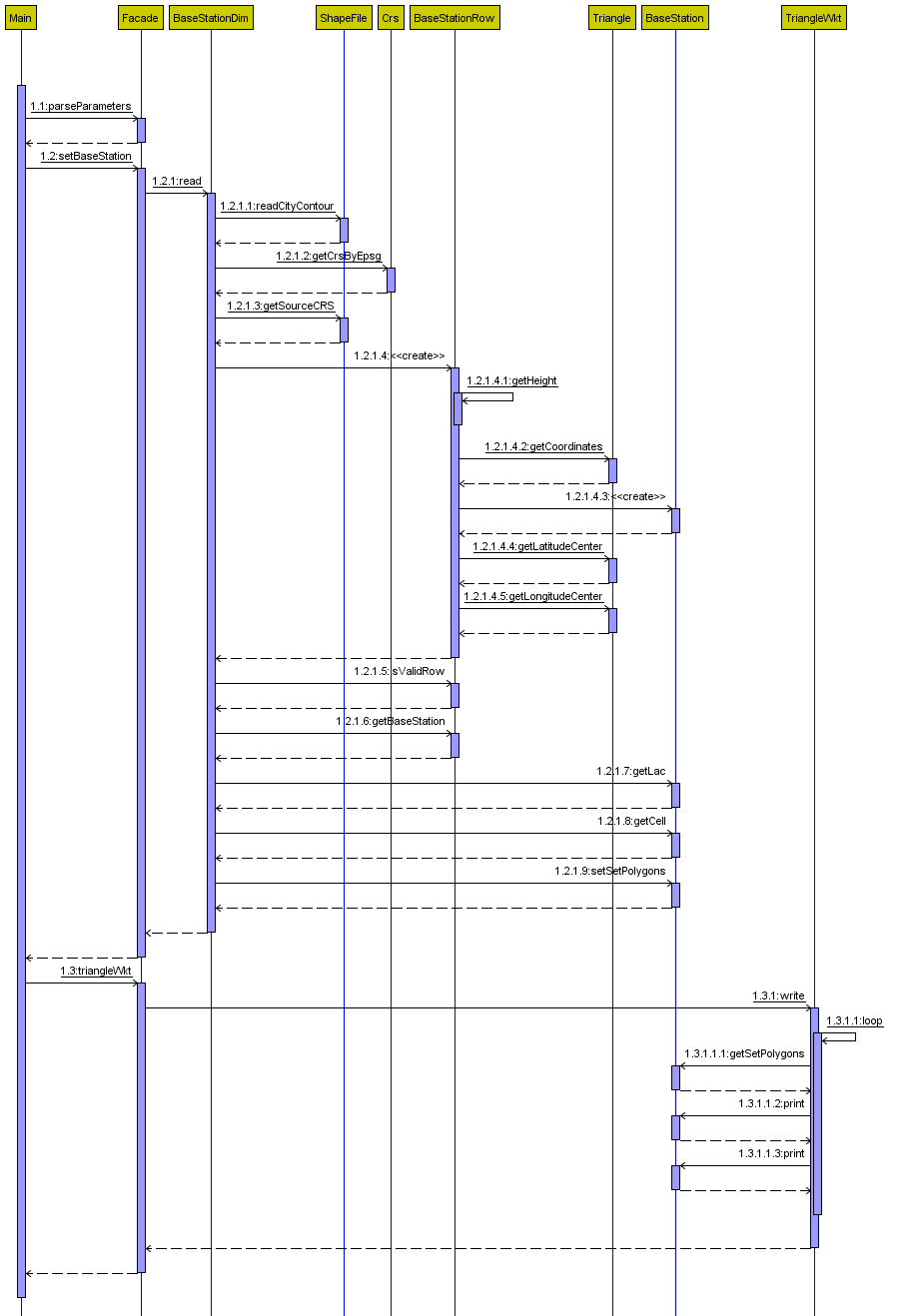
Далее приведены диаграммы основных классов реализуемого модуля на рисунке 19.

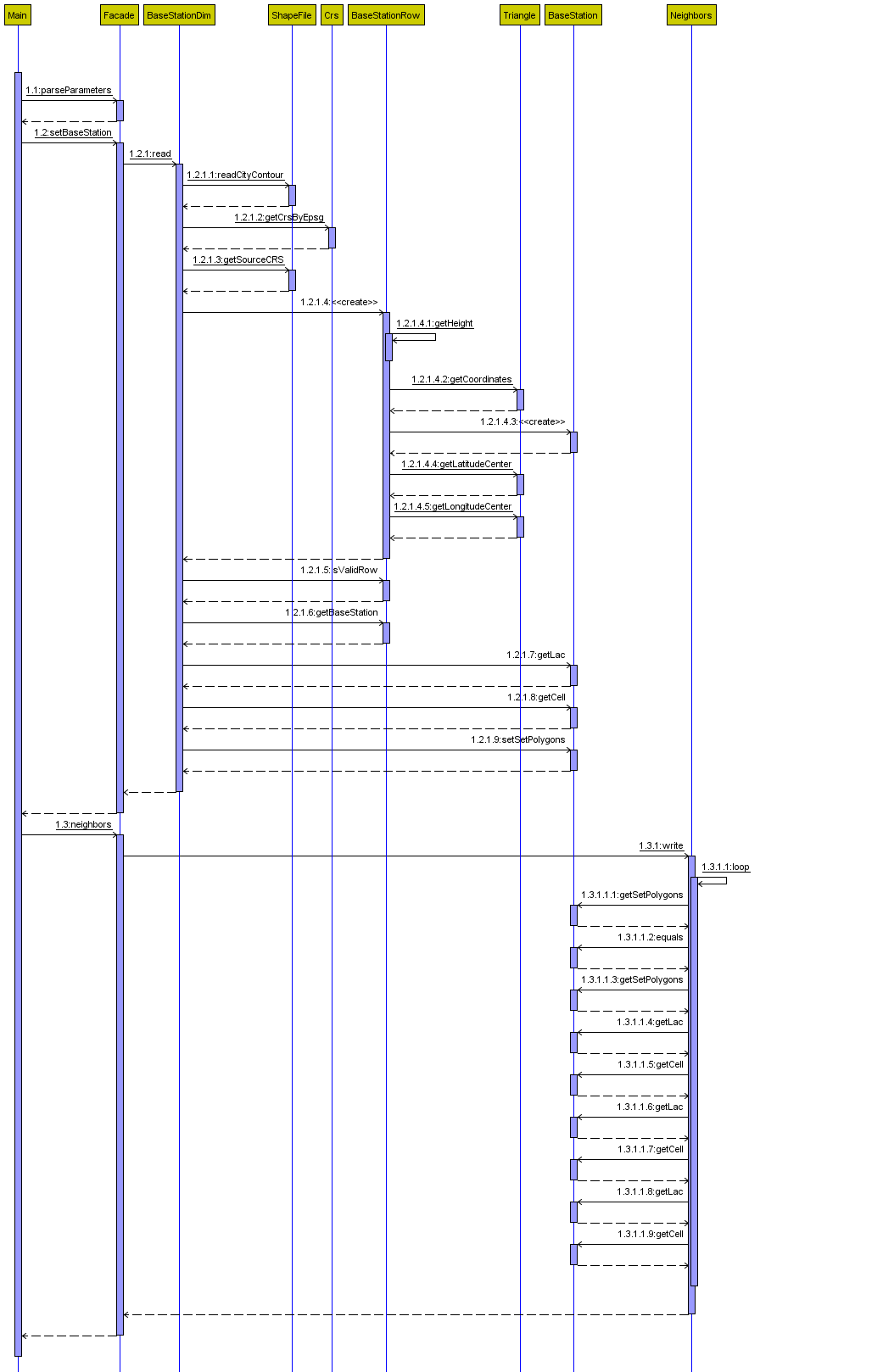
Рисунок 19 – Диаграмма классов модуля формирования справочников покрытия, соседних БС и отчета по сетке 500 на 500



Рисунок 20 – Диаграмма последовательности формирования вспомогательных справочников

 Рисунок 21 – Диаграмма последовательности формирования справочника БС

Рисунок 22 – Диаграмма последовательности формирования справочника покрытий БС

 Рисунок 23 – Диаграмма последовательности формирования справочника покрытий соседних БС

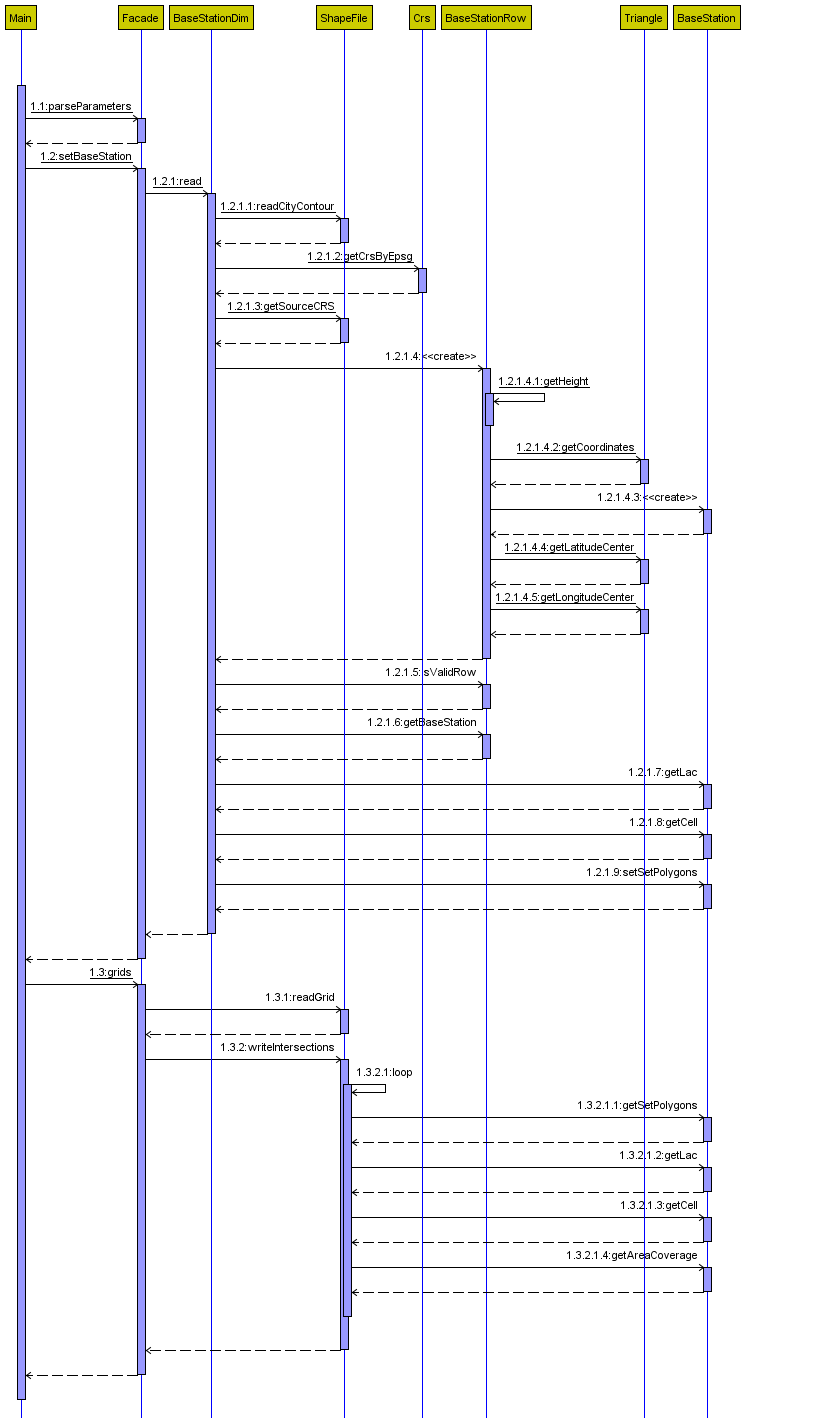


Рисунок 24 – Диаграмма последовательности формирования справочника сетки 500 на 500

* 1. Реализация результирующего отчета «Монобренд»

В данном разделе приведены диаграммы классов основных пакетов модуля системы на рисунках 25, 26, 27, 28.



Рисунок 25 – Диаграмма классов пакета polyimsi модуля формирования результирующего отчета «Монобренд»



Рисунок 26 – Диаграмма классов пакета grid модуля формирования результирующего отчета «Монобренд»



Рисунок 27 – Диаграмма классов пакета specs модуля формирования результирующего отчета «Монобренд»



Рисунок 28 – Диаграмма вспомогательных классов пакета specs модуля формирования результирующего отчета «Монобренд»

* 1. Реализация модуля по определению пола абонента

Таблица 9 – Описание источников для задачи определения пола

|  |  |
| --- | --- |
| Источник | Путь до источника |
| BIIS\_FCT.FCT\_USAGE\_PREP\_CHA\_N – существующий интерфейсный объект BIIS, содержащий информацию о трафике абонентов Prepaid. | – |
| BIIS\_FCT.FCT\_USAGE\_PREP\_PSA\_N – существующий интерфейсный объект BIIS, содержащий информацию об SMS-трафике абонентов Prepaid. | – |
| BIIS\_FCT.FCT\_USAGE\_POSTPAID – существующий интерфейсный объект BIIS, содержащий информацию о трафике абонентов Postpaid. | – |
| BIIS.DIM\_BAN – существующий интерфейсный объект BIIS, содержащий информацию об абонентах. | – |
| SPP.AGG\_SUBS\_GENDER – существующий агрегат, содержащий информацию о предполагаемом поле абонентов, на основе смс-сообщений | /user/tech\_arsspp\_bgd\_ms/AGG\_SUBS\_GENDER |
| DWH.FCT\_PDE\_SUBS\_ATTR\_DAILY – существующий объект BIIS, содержащий информацию об абонентах. | Источник будет подключен в HDFS |
| BIIS\_FCT.FCT\_RTC\_MONTHLY – ежемесячный агрегат по активной абонентской базе (ААБ) | – |

Таблица 10 – Описание полей отчета по определению пола абонента

| Показатель | Формат данных | Алгоритм расчета |
| --- | --- | --- |
| CTN абонента | Строка | – |
| Пол абонента из анкеты | Символьный | Пол абонента из источника FCT\_PDE\_SUBS\_ATTR\_DAILY |
| Пол абонента из справочника договоров абонентов | Целое положительное число | Пол абонента из источника DIM\_BAN |
| Количество исходящих звонков 23 февраля 2015 | Целое положительное число | Количество исходящих звонков по данным голосового трафика (fct\_usage\_prep\_cha\_n, fct\_usage\_postpaid) за 23 февраля текущего года |
| Количество исходящих звонков 8 марта 2015 | Целое положительное число | Количество исходящих звонков по данным голосового трафика (fct\_usage\_prep\_cha\_n, fct\_usage\_postpaid) за 8 марта текущего года |
| Количество исходящих смс 23 февраля 2015 | Целое положительное число | Количество исходящих SMS по данным SMS-трафика (fct\_usage\_prep\_psa\_n, fct\_usage\_postpaid) за 23 февраля текущего года |
| Количество исходящих смс 8 марта 2015 | Целое положительное число | Количество исходящих SMS по данным SMS-голосового трафика (fct\_usage\_prep\_psa\_n, fct\_usage\_postpaid) за 8 марта текущего года |
| Количество входящих звонков 23 февраля 2015 | Целое положительное число | Количество входящих звонков по данным голосового трафика (fct\_usage\_prep\_cha\_n, fct\_usage\_postpaid) за 23 февраля текущего года |
| Количество входящих звонков 8 марта 2015 | Целое положительное число | Количество входящих звонков по данным голосового трафика (fct\_usage\_prep\_cha\_n, fct\_usage\_postpaid) за 8 марта текущего года |
| Количество входящих смс 23 февраля 2015 | Целое положительное число | Количество входящих SMS по данным SMS-трафика (fct\_usage\_prep\_psa\_n, fct\_usage\_postpaid) за 23 февраля текущего года |
| Количество входящих смс 8 марта 2015 | Целое положительное число | Количество входящих SMS по данным SMS-трафика (fct\_usage\_prep\_psa\_n, fct\_usage\_postpaid) за 8 марта текущего года |
| Среднее количество исходящих звонков за все субботы февраля 2015 | Целое положительное число | Суммарное количество исходящих звонков за субботы февраля 2015 по данным голосового трафика (fct\_usage\_prep\_cha\_n, fct\_usage\_postpaid) деленное на число суббот |
| Среднее количество исходящих звонков за все субботы марта 2015 | Целое положительное число | Суммарное количество исходящих звонков за субботы марта 2015 по данным голосового трафика (fct\_usage\_prep\_cha\_n, fct\_usage\_postpaid) деленное на число суббот |
| Среднее количество входящих звонков за все субботы февраля 2015 | Целое положительное число | Суммарное количество входящих звонков за субботы февраля по данным трафика (fct\_usage\_prep\_cha\_n, fct\_usage\_postpaid) деленное на число суббот |
| Среднее количество входящих звонков за все субботы марта 2015 | Целое положительное число | Суммарное количество входящих звонков за субботы марта 2015 по голосового трафика (fct\_usage\_prep\_cha\_n, fct\_usage\_postpaid) деленное на число суббот |
| Среднее количество входящих смс за все субботы февраля 2015 | Целое положительное число | Суммарное количество входящих SMS за субботы февраля по данным трафика (fct\_usage\_prep\_psa\_n, fct\_usage\_postpaid) деленное на число суббот |
| Среднее количество входящих смс за все субботы марта 2015 | Целое положительное число | Суммарное количество входящих SMS за субботы марта 2015 по данным трафика (fct\_usage\_prep\_psa\_n, fct\_usage\_postpaid) деленное на число суббот |
| Объем Data-трафика | Целое положительное число | Суммарный data-трафик абонента за предыдущий месяц по данным интернет трафика (fct\_usage\_prep\_ogprs\_n) |



Рисунок 29 – Диаграмма классов модуля gender-model



Рисунок 30 – Диаграмма классов пакета prepare модуля gender-prediction



Рисунок 31 – Диаграмма классов пакета features модуля gender-prediction

* 1. Автоматизация модулей по формированию отчетов геоаналитики

Скрипты автоматизации Oozie-компонентов приведены в приложениях «Б», «В», «Г».

* 1. Анализ полученных результатов

Для последующего нанесения полученных данных из выгрузки потребовалось программное обеспечение QGIS Desktop 2.8.4. [32]. Содержимое требуемых файлов для создания проекта в QGIS приведены в приложениях «Д» и «Е».

На основе имеющихся .shp-файлов строится карта города. Затем результирующий отчет привязывается к топологии по имеющимся координатам. Таким образом, мы получаем нечто похожее на тепловую карту с указанием распределения полученных количественных показателей. Параметры цветовой гаммы настраиваются внутри программы QGIS по процентилям [32].

На представленных ниже рисунках 32, 33, 34, 35 отражена ситуация за 14 сентября 2015 года, а именно количество активных абонентов в каждые из 4 отрезков времени. Для обозначения точек продаж разных мобильных операторов, включая компании конкурентов, приняты следующие условные обозначения:

* + 1. Билайн – красные треугольники;
    2. Мегафон – желтые квадраты;
    3. МТС – белые пятиугольники;
    4. Теле2 – зеленые ромбы.



Рисунок 32 – Количество активных абонентов в городе Нижний Новгород в промежуток с 0 до 6 часов до полудня



Рисунок 33 – Количество активных абонентов в городе Нижний Новгород в промежуток с 6 до 9 часов до полудня



Рисунок 34 – Количество активных абонентов в городе Нижний Новгород в промежуток с 9 до 21 часов после полудня



Рисунок 35 – Количество активных абонентов в городе Нижний Новгород в промежуток времени с 21 до 00

В результате анализа полученных сведенийзаказчикомпринималось решение о наиболее оптимальном расположении точек обслуживания клиентов мобильного оператора.

Заключение

В ходе работы над задачей геоаналитики были разработаны следующие модули:

1. модуль формирования справочника уникальных базовых станций;
2. модуль формирования справочников покрытия, соседних базовых станций и отчета по сетке 500 на 500;
3. модуль формирования результирующего отчета «Монобренд».

Полученные результаты, а именно результирующий отчет и модули формирования вышеперечисленных справочников соответствуют критериям качества и приняты как продуктивные задачи для автоматизированной работы на промышленном кластере.

В ходе работы над задачей машинного обучения были получены следующие результаты:

* + 1. создана модель определения пола абонента на основе метода
    2. опорных векторов (SVM);
    3. измерена точность обученной модели, показатель которой составляет быть не менее 70%;
    4. автоматизировано определение пола абонента на кластере (период обновления данных – 1 месяц);
    5. осуществлена проверка корректности предсказания на основе расчета автоматизированного кода с ранее предсказанными значениями у избранных CTN.

В дальнейшем предполагается увеличить точность модели. Для это

требуется подключить к автоматизации определения пола абонентов источники:

* + 1. SMSProof – рассчитанные признаки (вектор длиной 6 из 0 и 1) есть в spp (даст точность до 89% на подмножестве приблизительно 7 миллиона ААБ),
    2. PDE – подключить источник в HDFS, использовать данные по полу для разметки (100% точность на подмножестве, размеченном PDE около 1 миллиона);
    3. BIIS gender – использовать пол из BIIS (dim\_ban) для модели (повысит точность до 70%);
    4. добавить несколько признаков (количество СМС).

Наука о больших данных ещё только оформляется, но уже сейчас она очень востребована – и в будущем будет востребована только больше. С её помощью можно решать невероятные задачи: предсказывать зарплату по описанию вакансии [22] предлагать пользователю музыку на основании его анкеты в интернете.

Для того чтобы извлечь полезную информацию из больших объемов данных, возможность автоматического хранения, систематизации, обзора и анализа данных имеет решающее значение. Технологии больших данных имеют потенциал, который позволяет раскрыть практическую ценность в массивных объемах данных, позволяя значительно улучшить бизнес-процессы клиентской аналитики.

Список использованных источников

1. http://habrahabr.ru/company/dca/blog/267361/
2. https://www.google.com/trends/explore
3. http://arxiv.org/abs/1309.5821
4. habrahabr.ru/post/267827/
5. http://habrahabr.ru/company/yandex/blog/271725/
6. <https://habrahabr.ru/post/267827/BigDatavsDataMining/>
7. <https://habrahabr.ru/post/198506/>
8. http://theoryofmotivation.info/ru/general/function.html
9. https://www.gartner.com/newsroom/id/2575515
10. http://www.gartner.com/newsroom/id/2819918
11. http://www.gartner.com/newsroom/id/3114217
12. http://gsl.azurewebsites.net/Portals/0/Users/dewitt/Papers/Hadoop/cacm2010.pdf
13. http://habrahabr.ru/company/beeline/blog/218669/].
14. http://rusbase.com/longread/how-data/
15. http://www.ibm.com/developerworks/ru/library/l-hadoop/
16. https://azure.microsoft.com/ru-ru/documentation/articles/hdinsight-hadoop-introduction/
17. http://research.google.com/archive/mapreduce.html
18. https://www.dezyre.com/article/hadoop-mapreduce-vs-apache-spark-who-wins-the-battle/83
19. https://blog.selectel.ru/hadoop-chast-1-razvertyvanie-klastera/
20. http://www.cloudera.com/downloads/quickstart\_vms/5-4.html?src=Coursera
21. http://www.cloudera.com/documentation/enterprise/latest/topics/cloudera\_quickstart\_vm.html
22. http://habrahabr.ru/post/271407/],
23. Hadoop in Action
24. Hadoop: The Definitive Guide, 2nd Edition
25. Hadoop Operations
26. MapReduce Design Pattern
27. Programming Hive
28. Hadoop Real World solutions CookBook
29. Professional Hadoop Solutions
30. Pro Hadoop
31. White, T. Hadoop: The Definitive Guide, 2nd Edition / T. White. – Sebastopol : O’Reilly Media, 2010. – 628 p.
32. Документация QGIS 2.0 [Электронный ресурс] : оф. сайт. – Режим доступа : http://docs.qgis.org/2.0/ru/docs/index.html

Зарубежные статьи

1. <http://www.ibm.com/developerworks/opensource/library/l-hadoop-1/l-hadoop-1-pdf.pdf>
2. <http://www.ibm.com/developerworks/opensource/library/l-hadoop-2/l-hadoop-2-pdf.pdf>
3. <https://developer.yahoo.com/hadoop/tutorial/module2.html>
4. <https://developer.yahoo.com/hadoop/tutorial/module3.html>
5. <https://developer.yahoo.com/hadoop/tutorial/module4.html>
6. <https://developer.yahoo.com/hadoop/tutorial/module7.html>
7. <http://blog.cloudera.com/blog/2013/01/how-to-schedule-recurring-hadoop-jobs-with-apache-oozie/>

Приложение А. HQL-скрипт для выгрузки эталонных данных

-- важно: если этого не сделать, будут ошибки при соединении с полями-массивами

set hive.mapjoin.optimized.hashtable=false;

-- загружаем справочник сеток

create table dim\_grid\_500 (

ZID int,

LAC int,

CELL\_ID int,

AREA\_GLARE float,

AREA\_COVERAGE float

) row format delimited fields terminated by '\;';

-- сворачиваем справочник до БС (массив zid)

create table dim\_grid\_500\_grp stored as orc as

select lac, cell\_id, collect\_set(zid) zids

from dim\_grid\_500 t

group by lac, cell\_id;

-- приемник xdr - соединяем все события из ADR, TDR и Polystar

create table nn\_xdr\_20150914\_20 stored as orc as

select imsi, to\_date(datetime) day\_dt, case

when hour(datetime) >= 0 and hour(datetime) <= 5 then 1

when hour(datetime) >= 6 and hour(datetime) <= 8 then 2

when hour(datetime) >= 9 and hour(datetime) <= 20 then 3

when hour(datetime) >= 21 and hour(datetime) <= 23 then 4

end part, lac, cell\_id from anritsu.anritsu\_adr a

where a.region = 'VLG' and a.city = 'nino' and a.date >= 20150914 and a.date <= 20150920

union all

select imsi, to\_date(datetime) day\_dt, case

when hour(datetime) >= 0 and hour(datetime) <= 5 then 1

when hour(datetime) >= 6 and hour(datetime) <= 8 then 2

when hour(datetime) >= 9 and hour(datetime) <= 20 then 3

when hour(datetime) >= 21 and hour(datetime) <= 23 then 4

end part, lac, cell\_id

from anritsu.anritsu\_tdr t

where t.region = 'VLG' and t.city = 'nino' and t.date >= 20150914 and t.date <= 20150920

union all

select imsi, to\_date(datetime\_val) day\_dt, case

when hour(datetime\_val) >= 0 and hour(datetime\_val) <= 5 then 1

when hour(datetime\_val) >= 6 and hour(datetime\_val) <= 8 then 2

when hour(datetime\_val) >= 9 and hour(datetime\_val) <= 20 then 3

when hour(datetime\_val) >= 21 and hour(datetime\_val) <= 23 then 4

end part, lac, cellid

from polystar.polystar\_data t

where t.region = 'VLG' and t.city = 'nino' and t.date >= 20150914 and t.date <= 20150920;

-- сворачиваем, чтобы убрать дубли

create table nn\_xdr\_20150914\_20\_rnd stored as orc as

select imsi, day\_dt, part, lac, cell\_id

from nn\_xdr\_20150914\_20 t

group by imsi, day\_dt, part, lac, cell\_id;

create table nn\_semifinal (

imsi\_cnt bigint,

day\_dt string,

day\_part int,

zids array<int>,

rnd\_zid int

) stored as orc;

with im as (

select count(distinct imsi) imsi\_cnt, day\_dt, part, lac, cell\_id

from nn\_xdr\_20150914\_20\_rnd

group by day\_dt, part, lac, cell\_id)

insert into table nn\_semifinal

select im.imsi\_cnt, im.day\_dt, im.part, gr.zids, gr.zids[cast ((rand() \* size(zids)) as int)] rnd\_zid

from im left join dim\_grid\_500\_grp gr on (gr.lac = im.lac and gr.cell\_id = im.cell\_id);

-- собираем данные по уникальным сочетаниям imsi - БС - день - интервал + пересечение со справочником

-- сетки. На этом этапе случайным образом выбираем zid для каждой БС.

-- Есть БС которых нет в справочнике - это БС области, в справочнике БС только из города.

create table NN\_XXXX stored as ORC as

with im as (

select distinct imsi, day\_dt, part, lac, cell\_id

from nn\_xdr\_20150914\_20\_rnd )

select im.imsi, im.day\_dt, im.part, gr.zids, gr.zids[cast ((rand() \* size(zids)) as int)] rnd\_zid

from im left join dim\_grid\_500\_grp gr on (gr.lac = im.lac and gr.cell\_id = im.cell\_id);

create table nn\_semifinal\_new (

imsi\_cnt bigint,

day\_dt string,

day\_part int,

rnd\_zid int

) stored as orc;

-- группируем данные для показателей по конкретным zid-ам

insert into table nn\_semifinal\_new

select count(imsi), day\_dt, part, rnd\_zid

from NN\_XXXX

group by day\_dt, part, rnd\_zid;

-- выгружаем данные в нужном нам формате

insert overwrite local directory '/export/home/agchursin/nino/results' row format delimited fields terminated by '\;'

select rnd\_zid,

case day\_part when 1 then '00:00:00-05:59:59' when 2 then '06:00:00-08:59:59' when 3 then '09:00:00-20:59:59' when 4 then '21:00:00-23:59:59' end as day\_interval,

sum(case when day\_dt = '2015-09-14' then imsi\_cnt else 0 end) as day20150914,

sum(case when day\_dt = '2015-09-15' then imsi\_cnt else 0 end) as day20150915,

sum(case when day\_dt = '2015-09-16' then imsi\_cnt else 0 end) as day20150916,

sum(case when day\_dt = '2015-09-17' then imsi\_cnt else 0 end) as day20150917,

sum(case when day\_dt = '2015-09-18' then imsi\_cnt else 0 end) as day20150918,

sum(case when day\_dt = '2015-09-19' then imsi\_cnt else 0 end) as day20150919,

sum(case when day\_dt = '2015-09-20' then imsi\_cnt else 0 end) as day20150920,

sum(case when day\_dt in ('2015-09-14', '2015-09-15', '2015-09-16', '2015-09-17', '2015-09-18') then imsi\_cnt else 0 end)/5 as average\_weekday,

sum(case when day\_dt in ('2015-09-19', '2015-09-20') then imsi\_cnt else 0 end)/2 as average\_weekend,

sum(case when day\_dt in ('2015-09-14', '2015-09-15', '2015-09-16', '2015-09-17', '2015-09-18', '2015-09-19', '2015-09-20') then imsi\_cnt else 0 end)/7 as average\_week

from nn\_semifinal\_new

--where zids is not null -- если надо отрезать БС из области

group by rnd\_zid, day\_part;

-- проверка общего количества по день/интервал чтобы оценить результаты.

select sum(imsi\_cnt), day\_dt, day\_part

from nn\_semifinal t

where zids is not null

group by day\_dt, day\_part

order by day\_dt, day\_part;

Приложение Б. Содержимое файла coordinator.properties

nameNode=hdfs://nameservice1

jobTracker=hd-name004.vimpelcom.ru:8050

oozie.use.system.libpath=true

coord\_application\_path=${nameNode}${workingDir}/coordinator.xml

oozie.coord.application.path=${coord\_application\_path}

wf\_application\_path=hdfs://nameservice1${workingDir}

queueName=${queueName}

startTime=2015-09-03T03:00Z

endTime=2015-10-03T03:00Z

Приложение В. Содержимое oozie-файла coordinator.xml

<coordinator-app name="bdest-194-biis-7965-grid-references" frequency="${coord:months(1)}" start="${startTime}" end="${endTime}" timezone="UTC" xmlns="uri:oozie:coordinator:0.2">

<action>

<workflow>

<app-path>${wf\_application\_path}</app-path>

<configuration>

<property>

<name>jobTracker</name>

<value>${jobTracker}</value>

</property>

<property>

<name>nameNode</name>

<value>${nameNode}</value>

</property>

<property>

<name>queueName</name>

<value>${queueName}</value>

</property>

<property>

<name>projectName</name>

<value>${projectName}</value>

</property>

<property>

<name>numReduceTasks</name>

<value>${numReduceTasks}</value>

</property>

<property>

<name>HCAT\_URI</name>

<value>${HCAT\_URI}</value>

</property>

<property>

<name>KRB\_PRINCIPAL</name>

<value>${KRB\_PRINCIPAL}</value>

</property>

<property>

<name>START\_DATE</name>

<value>${coord:formatTime(coord:dateOffset(coord:dateOffset(coord:nominalTime(), -1, 'MONTH'), -2, 'DAY'), 'yyyy-MM-dd')}</value>

</property>

<property>

<name>END\_DATE</name>

<value>${coord:formatTime(coord:dateOffset(coord:nominalTime(), -3, 'DAY'), 'yyyy-MM-dd')}</value>

</property>

<property>

<name>OUTPUT\_PATH</name>

<value>${OUTPUT\_PATH}/${coord:formatTime(coord:dateOffset(coord:nominalTime(), -1, 'MONTH'), 'yyyyMM')}</value>

</property>

<property>

<name>IMSI\_TMSI</name>

<value>${IMSI\_TMSI}/${coord:formatTime(coord:dateOffset(coord:nominalTime(), -1, 'MONTH'), 'yyyyMM')}</value>

</property>

<property>

<name>POLYSTAR</name>

<value>${POLYSTAR}/${coord:formatTime(coord:dateOffset(coord:nominalTime(), -1, 'MONTH'), 'yyyyMM')}</value>

</property>

<property>

<name>COVERAGE\_PATH</name>

<value>${COVERAGE\_PATH}/${coord:formatTime(coord:dateOffset(coord:nominalTime(), -1, 'MONTH'), 'yyyyMM')}</value>

</property>

<property>

<name>SHAPE\_FILES\_PATH</name>

<value>${SHAPE\_FILES\_PATH}</value>

</property>

<property>

<name>DIM\_BS\_POSITION</name>

<value>${DIM\_BS\_POSITION}</value>

</property>

<property>

<name>DIM\_CRAMER\_OSS\_SECTORS</name>

<value>${DIM\_CRAMER\_OSS\_SECTORS}</value>

</property>

<property>

<name>SAINT\_PETERSBURG</name>

<value>${SAINT\_PETERSBURG}</value>

</property>

<property>

<name>NIZHNIY\_NOVGOROD</name>

<value>${NIZHNIY\_NOVGOROD}</value>

</property>

<property>

<name>ROSTOV\_NA\_DONU</name>

<value>${ROSTOV\_NA\_DONU}</value>

</property>

<property>

<name>KRASNOYARSK</name>

<value>${KRASNOYARSK}</value>

</property>

<property>

<name>NOVOSIBIRSK</name>

<value>${NOVOSIBIRSK}</value>

</property>

<property>

<name>VOLGOGRAD</name>

<value>${VOLGOGRAD}</value>

</property>

<property>

<name>VORONEZH</name>

<value>${VORONEZH}</value>

</property>

<property>

<name>MOSCOW\_MO</name>

<value>${MOSCOW\_MO}</value>

</property>

<property>

<name>MOSCOW</name>

<value>${MOSCOW}</value>

</property>

<property>

<name>SAMARA</name>

<value>${SAMARA}</value>

</property>

<property>

<name>KAZAN</name>

<value>${KAZAN}</value>

</property>

<property>

<name>OMSK</name>

<value>${OMSK}</value>

</property>

<property>

<name>UFA</name>

<value>${UFA}</value>

</property>

<property>

<name>SAINT\_PETERSBURG\_ID</name>

<value>${SAINT\_PETERSBURG\_ID}</value>

</property>

<property>

<name>NIZHNIY\_NOVGOROD\_ID</name>

<value>${NIZHNIY\_NOVGOROD\_ID}</value>

</property>

<property>

<name>ROSTOV\_NA\_DONU\_ID</name>

<value>${ROSTOV\_NA\_DONU\_ID}</value>

</property>

<property>

<name>KRASNOYARSK\_ID</name>

<value>${KRASNOYARSK\_ID}</value>

</property>

<property>

<name>NOVOSIBIRSK\_ID</name>

<value>${NOVOSIBIRSK\_ID}</value>

</property>

<property>

<name>VOLGOGRAD\_ID</name>

<value>${VOLGOGRAD\_ID}</value>

</property>

<property>

<name>VORONEZH\_ID</name>

<value>${VORONEZH\_ID}</value>

</property>

<property>

<name>MOSCOW\_MO\_ID</name>

<value>${MOSCOW\_MO\_ID}</value>

</property>

<property>

<name>MOSCOW\_ID</name>

<value>${MOSCOW\_ID}</value>

</property>

<property>

<name>SAMARA\_ID</name>

<value>${SAMARA\_ID}</value>

</property>

<property>

<name>KAZAN\_ID</name>

<value>${KAZAN\_ID}</value>

</property>

<property>

<name>OMSK\_ID</name>

<value>${OMSK\_ID}</value>

</property>

<property>

<name>UFA\_ID</name>

<value>${UFA\_ID}</value>

</property>

</configuration>

</workflow>

</action>

</coordinator-app>

Приложение Г. Содержимое oozie-файла coordinator.properties

#!/usr/bin/env bash

saint\_petersburg="saint\_petersburg"

nizhniy\_novgorod="nizhniy\_novgorod"

rostov\_na\_donu="rostov\_na\_donu"

krasnoyarsk="krasnoyarsk"

novosibirsk="novosibirsk"

volgograd="volgograd"

voronezh="voronezh"

moscow\_mo="moscow\_mo"

moscow="moscow"

samara="samara"

kazan="kazan"

omsk="omsk"

ufa="ufa"

saint\_petersburg\_id="599,194844"

nizhniy\_novgorod\_id=5971

rostov\_na\_donu\_id=5593

krasnoyarsk\_id="8668,307733,307735"

novosibirsk\_id=5837

volgograd\_id=7264

voronezh\_id=7171

moscow\_mo\_id=12

moscow\_id="601,186351,6512168,186347,367230,186349"

samara\_id=5497

kazan\_id=8911

omsk\_id=5802

ufa\_id=11152

OOZIE\_RR=http://hd-name003.vimpelcom.ru:11000/oozie

queueName="prod"

projectDir=/user/$USER/GRIDS

#Путь к покрытиям БС (Shape Files)

#coverage\_path=/user/$USER/bd-general/

coverage\_path=/user/achursin/adhoc/$USER/bd-general-shp/

dimCramerOssSectors=

/apps/hive/warehouse/biis.db/dim\_cramer\_oss\_sectors

dimBsPosition=/apps/hive/warehouse/biis.db/dim\_bs\_position

imsi\_tmsi\_msk\_path=/user/anritsu/data/\*DR/MSK/moscow/

polystar\_msk\_path=/user/polystar/data/MSK/moscow/\*/

numReduceTasks=100

projectName="bdest-194-biis-7965-grid-references"

projectPath=$HOME/Tasks/${projectName}

workingDir=${projectDir}/work

shapeFileDir=${projectDir}/shp

outputPath=${projectDir}

hadoop fs -rm -R $workingDir

hadoop fs -mkdir -p $workingDir

hadoop fs -mkdir -p $shapeFileDir

hadoop fs -put $HOME/Tasks/$projectName/\*.xml $workingDir

hadoop fs -put $HOME/Tasks/$projectName/\*.properties $workingDir

hadoop fs -put $HOME/Tasks/$projectName/shp/\* $shapeFileDir

hadoop fs -mkdir -p ${workingDir}/lib

hadoop fs -put ${projectPath}/lib/\*.jar ${workingDir}/lib

oozie job -auth KERBEROS \

-config /export/home/$USER/Tasks/$projectName/coordinator.properties \

-D projectName=${projectName} \

-D queueName=${queueName} \

-D numReduceTasks=${numReduceTasks} \

-D workingDir=${workingDir} \

-D KRB\_PRINCIPAL=hive/hd-name004.vimpelcom.ru@BEE.VIMPELCOM.RU \

-D HCAT\_URI=thrift://hd-name004.vimpelcom.ru:9083 \

-D DIM\_BS\_POSITION=${dimBsPosition} \

-D DIM\_CRAMER\_OSS\_SECTORS=${dimCramerOssSectors} \

-D COVERAGE\_PATH=${coverage\_path} \

-D IMSI\_TMSI=${imsi\_tmsi\_msk\_path} \

-D POLYSTAR=${polystar\_msk\_path} \

-D OUTPUT\_PATH=${outputPath} \

-D SHAPE\_FILES\_PATH=${shapeFileDir} \

-D SAINT\_PETERSBURG=${saint\_petersburg} \

-D NIZHNIY\_NOVGOROD=${nizh niy\_novgorod} \

-D ROSTOV\_NA\_DONU=${rostov\_na\_donu} \

-D KRASNOYARSK=${krasnoyarsk} \

-D NOVOSIBIRSK=${novosibirsk} \

-D VOLGOGRAD=${volgograd} \

-D VORONEZH=${voronezh} \

-D MOSCOW\_MO=${moscow\_mo} \

-D MOSCOW=${moscow} \

-D SAMARA=${samara} \

-D KAZAN=${kazan} \

-D OMSK=${omsk} \

-D UFA=${ufa} \

-D SAINT\_PETERSBURG\_ID=${saint\_petersburg\_id} \

-D NIZHNIY\_NOVGOROD\_ID=${nizhniy\_novgorod\_id} \

-D ROSTOV\_NA\_DONU\_ID=${rostov\_na\_donu\_id} \

-D KRASNOYARSK\_ID=${krasnoyarsk\_id} \

-D NOVOSIBIRSK\_ID=${novosibirsk\_id} \

-D VOLGOGRAD\_ID=${volgograd\_id} \

-D VORONEZH\_ID=${voronezh\_id} \

-D MOSCOW\_MO\_ID=${moscow\_mo\_id} \

-D MOSCOW\_ID=${moscow\_id} \

-D SAMARA\_ID=${samara\_id} \

-D KAZAN\_ID=${kazan\_id} \

-D OMSK\_ID=${omsk\_id} \

-D UFA\_ID=${ufa\_id} \

-run \

-oozie $OOZIE\_RR

Приложение Д. Содержимое QGIS-файла Nizhniy\_NovgorodGrid.prj

PROJCS["WGS\_84\_North\_Pole\_LAEA\_Russia",GEOGCS["GCS\_WGS\_1984",DATUM["D\_WGS\_1984",SPHEROID["WGS\_1984",6378137,298.257223563]],PRIMEM["Greenwich",0],UNIT["Degree",0.017453292519943295]],PROJECTION["Lambert\_Azimuthal\_Equal\_Area"],PARAMETER["latitude\_of\_origin",90],PARAMETER["central\_meridian",90],PARAMETER["false\_easting",0],PARAMETER["false\_northing",0],UNIT["Meter",1]]

Приложение Е. Содержимое файла QGIS-файла Nizhniy\_NovgorodGrid.qpj

PROJCS["WGS 84 / North Pole LAEA Russia",GEOGCS["WGS 84",DATUM["WGS\_1984",SPHEROID["WGS 84",6378137,298.257223563,AUTHORITY["EPSG","7030"]],AUTHORITY["EPSG","6326"]],PRIMEM["Greenwich",0,AUTHORITY["EPSG","8901"]],UNIT["degree",0.0174532925199433,AUTHORITY["EPSG","9122"]],AUTHORITY["EPSG","4326"]],PROJECTION["Lambert\_Azimuthal\_Equal\_Area"],PARAMETER["latitude\_of\_center",90],PARAMETER["longitude\_of\_center",90],PARAMETER["false\_easting",0],PARAMETER["false\_northing",0],UNIT["metre",1,AUTHORITY["EPSG","9001"]],AXIS["X",EAST],AXIS["Y",NORTH],AUTHORITY["EPSG","3576"]]

Приложение Ж. Содержимое файла модели gender\_model.model

solver\_type L1R\_LR

nr\_class 2

label 0 1

nr\_feature 7

bias 0.000000000000000

w

-0.09339977157324555

0.1325172774150834

0.001349625080676907

0.01743943478474989

-0.003943468510724177

-0.02552200094724406

0

0

w - это веса, которые мы нашли после обучения