МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ Н. Г. ЧЕРНЫШЕВСКОГО»

УТВЕРЖДАЮ
Зав.кафедрой,
к. ф.-м. н.
_____ А. С. Иванов

ОТЧЕТ О ПРАКТИКЕ

студента 4 курса 411 группы факультета КНиИТ
Вязкова Андрея Андреевича

вид практики: преддипломная
кафедра: математической кибернетики и компьютерных наук
курс: 4
семестр: 8
продолжительность: 4 нед., с 30.04.2020 г. по 27.05.2020 г.

Руководитель практики от университета,

доцент, к. т. н. _____ В. М. Соловьев

Руководитель практики от организации (учреждения, предприятия),

к. ф.-м. н _____ С. В. Миронов

Тема практики: «Построение архитектуры ETL процесса и ее программная реализация»

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Большие данные являются популярным трендом в современном IT. С помощью них можно прогнозировать события, угадывать поведение пользователя, обнаруживать мошенническую активность. Большие данные используются в таких отраслях как: финансовые услуги, здравоохранение, физика.

Однако не всегда данные могут быть однородными и готовыми для дальнейшего анализа. Чтобы они были готовы их необходимо предварительно обработать. Подготовка данных для дальнейшего анализа называется ETL процессом от английского Extract — извлечение, Transform — трансформация, Load — загрузка. В данной работе показан пример построения архитектуры этого процесса с помощью современных инструментов в облаке Amazon Web Services.

1 Большие данные

1.1 Основные понятия

Большие данные — обозначение структурированных и неструктурированных данных огромных объёмов и значительного многообразия, эффективно обрабатываемых горизонтально масштабируемыми программными инструментами.

Термин «Большие Данные» **??** вызывает множество споров, многие полагают, что он означает лишь объем накопленной информации, но не стоит забывать и о технической стороне. К данной сфере именно относится обработка и хранение большого объема информации, для которого традиционные способы являются неэффективными, а также сервисные услуги. В таблице 1 представлена сравнительная характеристика традиционной базы данных и базы больших данных.

Таблица 1 – Сравнительный анализ традиционной базы данных и базы Больших данных

Характеристика	Традиционная БД	База больших данных
Объем информации	От гигабайт до терабайт	От петабайт до эксабайт
Способ хранения	Централизованный	Децентрализованный
Структурированность данных	Структурирована	Неструктурирована или полуструктурирована
Модель хранения и обработки данных	Вертикальная модель	Горизонтальная модель
Взаимосвязь данных	Сильная	Слабая

Большие данные характеризуются следующими признаками:

— **Volume** — объем, накопленная база данных представляет собой большой объем информации, который трудно хранить и обрабатывать традиционными способами;

— **Velocity** — скорость, данный признак указывает как на увеличивающуюся скорость накопления данных, так и на скорость обработки данных, в последнее время стали более востребованы технологии обработки данных в реальном времени.

— **Variety** — многообразие, возможность одновременной обработки струк-

турированной и неструктурированной разноформатной информации. Главное отличие структурированной информации – это то, что она может быть классифицирована. Неструктурированная информация включает в себя видео, аудио файлы, свободный текст, информацию, поступающую из социальных сетей. Данная информация нуждается в комплексной обработке для дальнейшего анализа.

— **Veracity** — достоверность, насколько точны полученные данные? **??**

1.2 Отрасли применения

Большие Данные получили широкое распространение во многих отраслях. Их используют в здравоохранении, телекоммуникациях, торговле, логистике, в финансовых компаниях, а также в государственном управлении. Ниже представлены примеры их применения:

— Финансовые услуги — большие данные дают возможность проанализировать кредитоспособность заемщика. Внедрение технологий Больших Данных позволит сократить время рассмотрения кредитных заявок. С помощью Больших Данных можно проанализировать операции конкретного клиента и предложить подходящие именно ему банковские услуги;

— Телекоммуникации — в этой отрасли большие данные получили популярность у сотовых операторов. Операторы сотовой связи наравне с финансовыми организациями имеют одни из самых объемных баз данных, что позволяет им проводить наиболее глубокий анализ накопленной информации. Главной целью анализа данных является удержание существующих клиентов и привлечение новых;

— Горнодобывающая и нефтяная промышленности — большие данные используются как при добыче полезных ископаемых, так и при их переработке и сбыте. Предприятия могут на основании поступившей информации делать выводы об эффективности разработки месторождения, отслеживать график капитального ремонта и состояния оборудования, прогнозировать спрос на продукцию и цены;

1.3 Технологии Больших данных

Для сбора и обработки больших данных используются как известные технологии и концепции, так и достаточно новые. Вот некоторые из них:

— **SQL** (Structured Query Language) — структурированный язык запросов для работы с базами данных. С его помощью можно манипулировать данными, а за управление данными отвечает движок базы данных и система управления базами данных;
— **NoSQL** (Not Only SQL) — совокупность подходов, направленных на создание базы данных, отличной от реляционной. Такой подход удобно использовать при меняющейся структуре данных;
— **MapReduce** — модель программирования распределенных вычислений над большими объемами данных, представленная компанией Google в 2004 году. Весь процесс делится на две фазы: map и reduce. Все вычисления производятся на кластере из главного узла (master node) и рабочих узлов (worker nodes) Каждая из фаз принимает на вход и отдает список пар ключ/значение. Фаза map принимает список, главный узел разбивает этот список на части отправляет на каждый рабочий узел. Каждый рабочий узел применяет заранее написанную функцию к каждому элементу такого списка. Фаза reduce группирует значений по ключам и отправку результатов на главный узел. Также между этими фазами есть промежуточная фаза, называемая shuffle. На этой фазе данные перераспределяются между рабочими узлами на основе ключей, таким образом, чтобы все данные по одному ключу лежали на одном рабочем узле.
— **Apache Spark** — фреймворк для быстрых кластерных вычислений, разработанный в Университете Беркли и переданный в фонд Apache. Распространяется по открытой лицензии. Spark использует парадигму резидентных вычислений. Все данные хранятся и обрабатываются в оперативной памяти. Использует ленивые вычисления. Основной концепцией в Spark является RDD (Resilient Distributed Dataset)**??** — неизменяемая коллекция объектов, которую можно обрабатывать параллельно. RDD создается путем загрузки внешнего набора данных или распределения коллекции из основной программы. RDD поддерживает 2 типа операций**??**: трансформации — операции над RDD, результатом которых является новый RDD и действия — операции, которые возвращают значения вычислений над RDD. Ядро Spark состоит из высокоуровневых библиотек, таких как: Spark SQL — обертка над SQL, позволяю-

щаяя делать запросы в БД из кода, Spark Streaming — библиотека для потоковой обработки данных в реальном времени, MLlib — библиотека машинного обучения, GraphX — это библиотека для манипуляций над графами и выполнения с ними параллельных операций. Spark имеет программные интерфейсы на языках Scala, Java, Python, R.

Также существуют сервисы которые предоставляют готовые услуги по предоставлению баз данных и вычислительных ресурсов. Такие компании как Amazon, Google и Microsoft предлагают подобные решения. Подробнее об этих решениях будет описано в разделе 2.

1.4 Открытые данные

2 ETL процесс. Его задачи и технологии

ETL (Extract, Transform, Load) — одна из главных процедур копирования данных из одного или нескольких источников в конечную систему. Данные в конечной системе имеют общий структурированный вид. Термин набрал популярность в 1970-х годах и преимущественно используется при построении хранилищ данных.**??**.

ETL процесс состоит из трех фаз. Ниже представлено описание каждой из них**??**:

— Extract — извлечение данных из различных источников. Источниками могут выступать: результаты работы программ, логи этих программ, копии таблиц базы данных, любой внешний набор данных;

— Transform — выполнение преобразований над данными, их фильтрация, группировка и агрегация. На этом этапе сырые данные превращаются в готовый для анализа датасет;

— Load — загрузка обработанных данных в место конечного использования, например в хранилище данных. Эти данные могут быть использованы конечными пользователями или их можно подать на вход другому ETL процессу.

ETL придает данным значительную ценность. Это не просто копирование данных из исходного источника в хранилище данных. ETL решает такие проблемы как:

— Удаляет ошибки и исправляет недостающие данные;

— Настраивает данные из нескольких источников для совместного использования:

— Структурирует данные для использования конечными инструментами:

2.1 Технологии для решения задач ETL

Так как ETL — весьма комплексная структура, то к ней выдвигаются определенные требования.

Хранилища данных должны иметь копию исходных данных, подтверждение транзакций, которые изменили данные, а также подтверждение безопасности копий данных с течением времени. Исходные данные должны быть качественными. Над некачественными данными придется проводить слишком много манипуляций, чтобы они были пригодны для дальнейшего исполь-

зования в ETL. Поэтому необходимо производить профилирование данных. Профилирование данных - это систематическое исследование качества, объема и контекста источника данных, позволяющее построить ETL процесс. Если источник данных содержит некачественные данные,то для его пригодности может потребоваться:

— Удаление некоторых полей полностью;
— Автоматическая замена поврежденных значений;
— Разработка нормализованного представления данных;

Управление и мониторинг процесса ETL — очень важно понимать каким образом будут строиться ETL пайплайны (от англ. pipeline — трубопровод). Будет ли разработчик конструировать их через пользовательский интерфейс или же писать код? Как хорошо хорошо инструмент реализует мониторинг и оповещение об ошибках? Реализован ли процесс перезапуска в случае ошибки? Существует ли возможность заново обработать данные, которые уже были обработаны?

Обработка данных — будет ли использован JVM-ориентированный язык (например Java или Scala) или будет использован SQL (например Hive). Плюсы первого подхода заключаются в использовании парадигмы MapReduce, а также в том, что с помощью этого подхода удобно писать пользовательские функции, минусы подхода в том, что придется изучать один из языков. Во втором подходе весь процесс происходит вокруг SQL таблиц, что весьма удобно. Минусы подхода заключаются в том, что для того чтобы написать пользовательские функции придется задействовать сторонний язык программирования.

В данном разделе будут описаны технологии и инструменты для решения и реализации той или иной задачи.

2.1.1 Хранилища данных

Amazon Redshift?? — сервис управления хранилищами данных в облаке, представленный компанией Amazon в 2012 году. Представляет собой кластер из узлов, который состоит из узла@=лидера и одного или нескольких вычислительных узлов. Тип и количество вычислительных узлов, которые нужны, зависят от размера данных, количества запросов, которые будут выполняться, и требуемой производительности выполнения запроса. Позволяет добавляет и удалять узлы без остановки кластера. Может хранить объ-

емы данных от сотен гигабайт до нескольких сотен петабайт. Работает на измененном движке PostgreSQL. Amazon Redshift позволяет делать резервные копии, как в ручном так и в автоматическом режиме. Также имеется шифрование данных и защищенный доступ по SSL. Данные хранятся на каждом узле в блоках, которые называются срезами, используется колоночное хранение. Redshift **??** использует архитектуру MPP (Massively Parallel Processing), разбивая большие наборы данных на куски, которые назначаются срезами в каждом узле. Быстрая производительность достигается параллельной обработкой каждого среза на узле. Главный узел объединяет результаты и возвращает их клиентскому приложению.

Apache Hive?? — хранилище данных создан компанией Facebook в 2010 году, затем передан фонду Apahe под открытой лицензией. Входит в экосистему Hadoop. Имеет собственный SQL диалект, называемый HiveQL. HiveQL имеет SQL запросы, выполняемые в Hive конвертируются в MapReduce программы. Есть возможность создания пользовательских функций. Данные таблиц хранятся в HDFS. Поддерживаются следующие форматы:

1. Текстовый файл — данные располагаются в строках, каждая строка является записью. Строки заканчиваются символом новой строки в UNIX формате.
2. Последовательные файлы — кодируются как ключ и значение для каждой записи. Записи хранятся в двоичном формате и, следовательно, занимают меньше места, чем текстовый формат:
3. Колоночный формат (RCFile, Apache Parquet) — позволяет хранить значения столбцов рядом друг с другом и все метаданные об этой таблице:
4. Avro файлы — формат кодирует схему своего содержимого непосредственно в файле, что позволяет пользователю самостоятельно сохранять сложные объекты. Avro не только формат файла, но и структура сериализации и десериализации;
5. Файлы ORC - это формат файла столбцов Optimized Row, который обеспечивает высокоэффективный способ хранения данных Hive и преодолевает ограничения других форматов файлов Hive.
6. Пользовательские форматы файлов;

Google BigQuery?? — хранилище данных, разработанное компанией Google в 2011 году. Архитектура BigQuery не требует настройки серверов.

В качестве движка для выполнения запросов используется Dremel — разработанный в Google и позволяющий сканировать миллиарды строк в секунду. Dremel использует архитектуру массивных параллельных запросов для сканирования данных в базовой системе управления файлами Colossus**??**. Colossus распределяет файлы на блоки по 64Мб среди множества вычислительных ресурсов, называемых узлами, которые сгруппированы в кластеры. Dremel использует колоночную структуру данных, аналогичную Redshift. Древовидная архитектура отправляет запросы тысячам машин за считанные секунды.

2.2 Планировщики ETL процессов

AWS Glue?? — бессерверный сервис для построения задач ETL, представленный компанией AWS в 2017 году, состоит из:

— централизованное хранилище метаданных хранилищ данных (AWS Data Catalog);

— подсистему ETL, автоматически генерирующую код на Python или Scala, кото- рый описывает трансформацию данных;

— планировщик заданий;

— подсистему мониторинга и перезапуска.

Основное назначение сервиса — построение системы доставки данных из различных источников в централизованное хранилище данных.

Архитектура AWS Glue представлена на рисунке 1 и состоит из следующих компонентов:

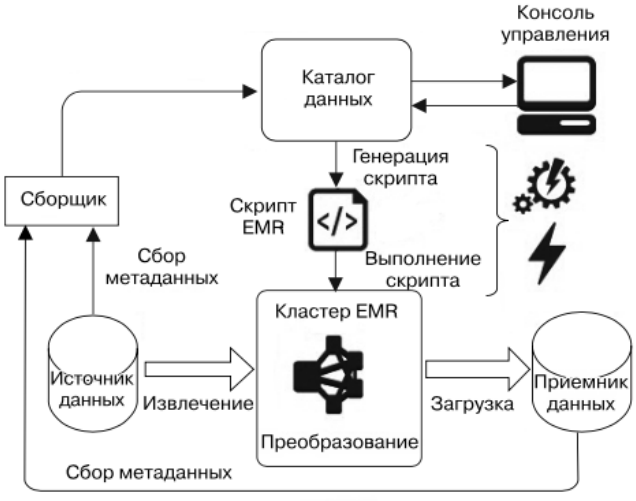

Рисунок 1 – Архитектура AWS Glue

— Выполняемые задания (jobs) — непосредственно рабочие процессы, производящие требуемое извлечение, трансформацию и загрузку данных из источников в назначения;
— Сборщик, или краулер (crawler) — программа, подключаемая к хранилищам данных, которая используется для наполнения каталога данных метаданными источников. В каталоге сборщику указывается хранилище данных, и он, определив схему источника, создает в каталоге соответствующий объект, называемый таблицей, поскольку все источники данных так или иначе могут быть описаны как таблицы. Вдобавок к описанию таблицы сборщик извлекает дополнительные метаданные, необходимые для построения сценария выполняемого задания ETL;
— Классификатор (classifier) — часть сборщика, определяющая схему данных источника. Поддерживает источники типа JSON, CSV, AVRO и XML. Кроме того, имеет возможность работать с реляционными базами данных, поддерживающими протокол JDBC;
— Сценарий трансформации данных — сценарий, выполняющий задачу трансформации и копирования, генерируемый AWS Glue или предоставляемый пользователем. Для описания задач поддерживаются языки Python и Scala.
— Триггер (trigger) — встроенный механизм запуска задания, работающий по расписанию или на основе перехваченного события. При срабатывании триггера происходит основной процесс ETL: на внутренних вычислительных ресурсах, содержащих фреймворк Apache Spark, выполняется сценарий трансформации данных, сгенерированный AWS Glue или предоставляемый пользователем.
— Сервер Notebook — интерактивный редактор кода, позволяющий писать программы на PySpark;
— Каталог данных (data catalog) — централизованное хранилище данных, которое содержит описания таблиц, сгруппированные в базы данных, выполняемых заданий и пр.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ