Оглавление

[1.1 Какой характер роста данных в технологиях BigData? 1](#_Toc61866376)

[1.2 Развитие каких технологий и устройств привело к появлению термина BigData? 1](#_Toc61866377)

[1.3 Что является основным глобальным источником роста объемов данных? 2](#_Toc61866378)

[1.4 Каковы причины актуальности технологий BigData? 2](#_Toc61866379)

[1.5 Какие этапы включает в себя парадигма BigData? 3](#_Toc61866380)

[1.6 Какие факторы тормозят внедрение технологий BigData? 3](#_Toc61866381)

[1.7 Что представляют собой процедуры ETL? 4](#_Toc61866382)

[1.8 Какой основной недостаток полной автоматизации аналитических процедур? 4](#_Toc61866383)

[1.9 Опишите основные характеристики BigData? 5](#_Toc61866384)

[1.10 Опишите основные методы, используемые для решения задач BigData? 5](#_Toc61866385)

[1.11 Подходы NoSQL. 7](#_Toc61866386)

[1.12 Технология Hadoop. 8](#_Toc61866387)

[1.13 Парадигма MapReduce. 9](#_Toc61866388)

[1.14 Файловая система HDFS. 10](#_Toc61866389)

[1.15 Утилиты Hadoop. 11](#_Toc61866390)

[1.16 Состав модуля YARN. 13](#_Toc61866391)

[1.17 Принципы работы с BigData. 14](#_Toc61866392)

[1.18 Платформа Oracle для BigData. 15](#_Toc61866393)

[1.19 Кластер Oracle Big data Appliance 16](#_Toc61866394)

[1.20 Кластер Oracle Exadata. 17](#_Toc61866395)

[1.21 Кластер Oracle Exalytics. 18](#_Toc61866396)

[1.22 Java Hotspot VM. 19](#_Toc61866397)

[1.23 Oracle Data Mining Algorithms. 20](#_Toc61866398)

[1.24 Oracle Endeca Information Discovery 21](#_Toc61866399)

[1.25 SAP HANA Data Platform для Big Data. 22](#_Toc61866400)

[1.26 Технологии OLTP BI и OLAP в хранилищах SAP HANA. 23](#_Toc61866401)

[1.27 Задачи аналитики в SAP HANA. 26](#_Toc61866402)

[1.28 Аналитика на уровне СУБД в SAP HANA. 26](#_Toc61866403)

[1.29 Интеграционная модель SAP HANA. 27](#_Toc61866404)

[1.30 Типовые задачи для SAP ESP. 27](#_Toc61866405)

[1.31 Принципы построения модели данных в HBase. 28](#_Toc61866406)

[1.32 Архитектура HBase. 29](#_Toc61866407)

[1.33 Опишите основные процессы HBase. 31](#_Toc61866408)

[1.34 Архитектура Hive. 32](#_Toc61866409)

[1.35 Основные объекты Hive. 32](#_Toc61866410)

[1.36 Опишите процедуры партицирования и бакетирования в Hive. 34](#_Toc61866411)

[1.37 Организация обработки запросов в Impala. 34](#_Toc61866412)

[1.38 Опишите основные функции Impala. 35](#_Toc61866413)

[1.39 Опишите операторы Pig Latin. 35](#_Toc61866414)

[1.40 Основные части скрипта на Pig Latin. 36](#_Toc61866415)

[1.41 Опишите состав библиотеки Mahout. 37](#_Toc61866416)

[1.42 Опишите концепцию Spark. 38](#_Toc61866417)

[1.43 Какие типы задач выполняет Oozie? 39](#_Toc61866418)

[1.44 Опишите состав потока работ Oozie. 39](#_Toc61866419)

[1.45 Архитектура Yarn. 40](#_Toc61866420)

[1.46 Из каких 2-х основных частей состоит утилита Pig Latin? Каковы их функции? 41](#_Toc61866421)

[1.47 Какой программный компонент является корнем иерархии модуля YARN? 41](#_Toc61866422)

[1.48 Какое предназначение утилиты Sqoop? 42](#_Toc61866423)

[1.49 Опишите основные функции файлового браузера HUE. 43](#_Toc61866424)

[1.50 Опишите механизм распределенных запросов в Impala. 44](#_Toc61866425)

[1.51 Таблицы каких 2-х видов используются в Hive? Опишите особенности их применения 44](#_Toc61866426)

[1.52 Назовите основные Утилиты Hadoop. Каково их назначение? 46](#_Toc61866427)

[1.53 Какие основные методы анализа данных применяются в технологиях Big Data? 48](#_Toc61866428)

[1.54 54 Опишите процедуру создания базы данных в Hive. 50](#_Toc61866429)

[1.55 Какой сервис используется в HBase для управления конфигурациями? 50](#_Toc61866430)

[1.56 Для чего в HBase используется BlockCache? 51](#_Toc61866431)

[1.57 Какая операция применяется в HBase для управления размером регионов? 51](#_Toc61866432)

[1.58 В каком формате хранятся данные в хранилище HBase Persistent Storage? 52](#_Toc61866433)

[1.59 Какие БД (строковые или колоночные) более эффективны для решения задач BigData? 52](#_Toc61866434)

[1.60 Для чего необходим лог-файл WAL в HBase? 53](#_Toc61866435)

## Какой характер роста данных в технологиях BigData?

Big Data в разном контексте могут подразумеваться данные большого объема, технология их обработки, проекты, рынок и даже компании, активно использующие эту технологию.

Экспоненциальный характер ростом вычислительных средств, приложений и пользователей — от миллионов в эпоху мэйнфреймов до сотен миллионов в эпоху ПК и миллиардов пользователей в эпоху мобильных устройств, мобильного Интернета, социальных сетей, «облачных» технологий и построения всевозможных решений «умной» экономики.

## Развитие каких технологий и устройств привело к появлению термина BigData?

Сегодня поток данных растет с невероятной скоростью. Данные поступают с разного рода устройств безопасности, проводится мониторинг износостойкости и прочности ответственных изделий. Огромные массивы данных генерируются в научных экспериментах, таких как регистрация метеорологических данных и астрономических наблюдений. Один только архив телескопа «Хаббл», накопленный за 15 лет, занимает около 50 Тбайт. В повседневной жизни мы сталкиваемся со всё возрастающим объемом данных — они поступают с экрана ПК, планшетника, мобильного телефона, приборных панелей автомобиля, радиочастотных идентификаторов, даже бытовые предметы, такие как кроссовки, и те могут генерировать данные. Множатся мобильные устройства и медицинские приложения, способные измерять пульс, давление и режим тренировки.

Отдельная тема — Интернет и мобильные технологии. Потоки информации генерируются всё новыми интернет-сервисами, социальными сетями, приложениями электронной торговли, приложениями о местонахождении абонентов сетей

Значительную часть информации создают не люди, а роботы, взаимодействующие как друг с другом, так и с другими сетями данных – такие, как, например, сенсоры и интеллектуальные устройства. При таких темпах роста количество данных в мире, по прогнозам исследователей, будет ежегодно удваиваться. Количество виртуальных и физических серверов в мире вырастет десятикратно за счет расширения и создания новых data-центров. В связи с этим растет потребность в эффективном использовании и монетизации этих данных. Поскольку использование Big Data в бизнесе требует немалых инвестиций, то надо ясно понимать ситуацию. А она, в сущности, проста: повысить эффективность бизнеса можно сокращая расходы или/и увеличивая объем продаж.

## Что является основным глобальным источником роста объемов данных?

При огромном объеме данных постоянно растущего числа вебстраниц это лишь вершина айсберга, большую же его часть составляют данные о данных. Аналитикам и интернет-маркетологам интересно не только содержание вебсайтов, но и обширная информация о пользователях: их привычки, история посещенных страниц, рекламные предпочтения, круг общения и знакомства. Именно это позволяет продвигать товары и рекламу в Интернете.

Мы можем только предполагать, насколько подробно можно проанализировать наше поведение по истории посещений в Сети. Очевидно, что для этого нужно обрабатывать действительно гигантские объемы данных (рис.).

## Каковы причины актуальности технологий BigData?

Анализируя данные, можно создавать новые сервисы и продукты, оптимизировать бизнес, а следовательно, зарабатывать деньги. По сути технология Big Data не есть что-то принципиально новое. Просто на современном этапе развития технологий появились средства, позволяющие хранить, обрабатывать и анализировать такие объемы данных, что стал меняться принцип подхода к их анализу. Аналитики рассуждают следующим образом: «Мы не знаем, нужна ли нам информация, а если нужна, то какая, до тех пор, пока не проанализируем, насколько она взаимосвязана». Стоимость хранения информации настолько снизилась, что появилась возможность собирать всё больше данных и анализировать, казалось бы, не связанные друг с другом факторы.

Данные собираются исходя из принципа «мы не знаем, чего мы не знаем». Человеческий мозг не может обнаружить такие закономерности, какие отмечает компьютер, выдавая совершенно неожиданные количественные взаимосвязи.

Например, он может количественно связать площадь того или иного цвета на обложке журнала с вероятностью его продаж в таком-то месяце года. Технология Big Data предоставляет услуги, помогающие раскрыть коммерческий потенциал мегамассивов данных за счет поиска ценных закономерностей и фактов путем объединения и анализа больших объемов данных

## Какие этапы включает в себя парадигма BigData?

Хранение и управление объемом данных в сотни петабайт, которые обычные реляционные базы данных не позволяют эффективно использовать;

Организация неструктурированной информации, состоящей из текстов, изображений, видео и других типов данных;

Анализ Big Data, который ставит вопрос о способах работы с неструктурированной информацией, генерацию аналитических отчетов, а также внедрение прогностических моделей.

## Какие факторы тормозят внедрение технологий BigData?

Сегодня анализируется только 0,5% накопленных цифровых данных несмотря на то, что объективно существуют общеотраслевые задачи, которые можно было бы решить с помощью аналитических решений класса Big Data. Развитые IT-рынки уже имеют результаты, по которым можно оценить ожидания, связанные с накоплением и обработкой больших данных

Одним из главных факторов, который тормозит внедрение Big Data - проектов, помимо высокой стоимости, считается проблема выбора обрабатываемых данных: то есть определение того, какие данные необходимо извлекать, хранить и анализировать, а какие – не принимать во внимание.

Многие представители бизнеса отмечают, что сложности при внедрении Big Data-проектов связаны с нехваткой специалистов – маркетологов и аналитиков. От качества работы сотрудников, занимающихся глубинной и предикативной аналитикой, напрямую зависит скорость возврата инвестиций в Big Data.

## Что представляют собой процедуры ETL?

В связи с большим объемом и высокой скоростью потока данных, процесс их сбора предполагает процедуры ETL в режиме реального времени. Для справки: ETL – от англ. Extract, Transform, Load — дословно «извлечение, преобразование, загрузка») — один из основных процессов в управлении хранилищами данных, который включает в себя: извлечение данных из внешних источников, их трансформацию и очистку с целью соответствия нуждам бизнес-модели и загрузка их в хранилище данных. ETL следует рассматривать не только как процесс переноса данных из одного приложения в другое, но и как инструмент подготовки данных к анализу.

## Какой основной недостаток полной автоматизации аналитических процедур?

Вопросы обеспечения безопасности данных, поступающих из внешних источников, должны иметь решения, соответствующие объемам собираемой информации. Так как методы анализа Big Data развиваются пока только вслед за ростом объема данных, большую роль играет свойство аналитических платформ использовать новые методы подготовки и агрегирования данных. Это говорит о том, что, например, данные о потенциальных покупателях или массивное хранилище данных с историей кликов на сайтах online-магазинов могут быть интересны для решения разных задач.

## Опишите основные характеристики BigData?

Volume-объем Большие данные работают с объемами информации, которые нельзя обработать средствами традиционных СУБД. Как правило, это объемы, превышающие 1 петабайт (PB). Основной идеей работы с таким объемом информации является параллельная обработка.

Variety–разнообразие Технологии BigData работают с разнородными и распределенными данными. Это могут быть как структурированные отчеты, так и аудиовидео файлы, изображения, информация из социальных сетей (отзывы, комментарии) и т.д.

Velocity – скорость Когда говорят о скорости, подразумевают не только скорость роста количества информации, но и скорость реакции на ее изменение. Технологии Больших Данных должны быстро реагировать на изменения уже имеющихся данных и корректировать результат с поправкой на них. В идеале – в режиме реального времени.

Veracity – достоверность. При таком объеме данных, с которыми работают Большие Данные, особую ценность представляет именно отделение достоверной информации от недостоверной, так как от этой информации зависит правильность принятых решений.

Value– ценность Собираемые данные должны оправдывать затраты на их обработку и анализ. Информация должна быть ценной и полезной.

## Опишите основные методы, используемые для решения задач BigData?

Методы класса Data Mining

Основная особенность Data Mining – это сочетание широкого математического инструментария и последних достижений в сфере информационных технологий.

К методам и алгоритмам Data Mining относятся следующие: искусственные нейронные сети, деревья решений, символьные правила, методы ближайшего соседа и k ближайшего соседа, метод опорных векторов, байесовские сети, линейная регрессия, корреляционно-регрессионный анализ; иерархические методы кластерного анализа, неиерархические методы кластерного анализа, в том числе алгоритмы k-средних и k- медианы; методы поиска ассоциативных правил, в том числе алгоритм Apriori; метод ограниченного перебора, эволюционное программирование и генетические алгоритмы, разнообразные методы визуализации данных и множество других методов.

**A/B-тестирование** (A/Btesting, Splittesting) Метод исследования, суть которого заключается в том, что контрольная группа элементов сравнивается с набором тестовых групп, в которых один или несколько показателей были изменены, для того, чтобы выяснить, какие из изменений улучшают целевой показатель. Большие данные позволяют провести огромное количество итераций и таким образом получить статистически достоверный результат.

**Краудсорсинг** Методика сбора данных из большого количества источников.

**Машинное обучение** Направление в информатике (исторически за ним закрепилось название “искусственный интеллект”), которое преследует цель создания алгоритмов самообучения на основе анализа эмпирических данных.

**Сетевой анализ** Набор методик анализа связей между узлами в сетях. Применительно к социальным сетям позволяет анализировать взаимосвязи между отдельными пользователями, компаниями, сообществами и т.п.

**Прогнозная аналитика** Класс методов анализа данных, концентрирующийся на прогнозировании будущего поведения объектов и субъектов с целью принятия оптимальных решений.

**Имитационное моделирование** Метод, позволяющий строить модели, описывающие процессы так, как они проходили бы в действительности. Такую модель можно «проиграть» во времени как для одного испытания, так и заданного их множества. При этом результаты будут определяться случайным характером процессов. По этим данным можно получить достаточно устойчивую статистику.

**Распознавание образов** Раздел информатики и смежных дисциплин, развивающий основы и методы классификации и идентификации предметов, явлений, процессов, объектов и т.д., которые характеризуются конечным набором некоторых свойств и признаков.

## Подходы NoSQL.

NoSQL (not only SQL) Термин, обозначающий ряд подходов, направленных на реализацию хранилищ баз данных, имеющих существенные отличия от моделей, используемых в традиционных реляционных СУБД с доступом к данным средствами языка SQL.

Традиционные СУБД ориентируются на требования ACID к транзакционной системе:

* атомарность (atomicity)
* согласованность (consistency),
* изолированность (isolation),
* надёжность (durability),

тогда как в NoSQL вместо ACID может рассматриваться набор свойств BASE:

* базовая доступность (basic availability) – каждый запрос гарантированно завершается (успешно или безуспешно).
* гибкое состояние (soft state) – состояние системы может изменяться со временем, даже без ввода новых данных, для достижения согласования данных.
* согласованность в конечном счёте(eventual consistency) – данные могут быть некоторое время рассогласованы, но приходят к согласованию через некоторое время.

Решения NoSQL отличаются не только проектированием с учётом масштабирования. Другими характерными чертами NoSQL-решений являются:

* Применение различных типов хранилищ.
* Возможность разработки базы данных без задания схемы.
* Использование многопроцессорности.
* Линейная масштабируемость (добавление процессоров увеличивает производительность). Инновационность: «не только SQL» открывает много возможностей для хранения и обработки данных.
* Сокращение времени разработки.
* Скорость: даже при небольшом количестве данных конечные пользователи могут оценить снижение времени отклика системы с сотен миллисекунд до миллисекунд.

## Технология Hadoop.

Hadoop – это свободно распространяемый набор утилит, библиотек и фреймворк для разработки и выполнения распределённых программ, работающих на кластерах из сотен и тысяч узлов. Эта основополагающая технология хранения и обработки больших данных ([Big Data](https://www.bigdataschool.ru/wiki/big-data" \t "_blank)) является проектом верхнего уровня фонда Apache Software Foundation.

Проект состоит из основных 4-х модулей:

**Hadoop Common** – набор инфраструктурных программных библиотек и утилит, которые используются в других решениях и родственных проектах, в частности, для управления распределенными файлами и создания необходимой инфраструктуры [1];

[**HDFS**](https://www.bigdataschool.ru/wiki/hdfs) – распределённая файловая система, Hadoop Distributed File System – технология хранения файлов на различных серверах данных (узлах, DataNodes), адреса которых находятся на специальном сервере имен (мастере, NameNode) [2]. За счет дублирования (репликации) информационных блоков, [HDFS](https://www.bigdataschool.ru/wiki/hdfs) обеспечивает надежное хранение файлов больших размеров, поблочно распределённых между узлами вычислительного кластера [1];

**YARN** – система планирования заданий и управления кластером (Yet Another Resource Negotiator), которую также называют [MapReduce](https://www.bigdataschool.ru/wiki/mapreduce" \t "_blank) 2.0 (MRv2) – набор системных программ (демонов), обеспечивающих совместное использование, масштабирование и надежность работы распределенных приложений [3]. Фактически, YARN является интерфейсом между аппаратными ресурсами кластера и приложениями, использующих его мощности для вычислений и обработки данных [1];

**Hadoop [MapReduce](https://www.bigdataschool.ru/wiki/mapreduce" \t "_blank)** – платформа программирования и выполнения распределённых [MapReduce](https://www.bigdataschool.ru/wiki/mapreduce" \t "_blank)-вычислений, с использованием большого количества компьютеров (узлов, nodes), образующих кластер.

## Парадигма MapReduce.

MapReduce – это фреймворк для вычисления некоторых наборов распределенных задач с использованием большого количества компьютеров (называемых «нодами»), образующих кластер.

Работа MapReduce состоит из двух шагов: Map и Reduce.

На Map шаге происходит предварительная обработка входных данных. Для этого один из компьютеров (называемый главным узлом – masternode) получает входные данные задачи, разделяет их на части и передает другим компьютерам (рабочим узлам – workernode) для предварительной обработки. Название данный шаг получил от одноименной функции высшего порядка.

На Reduce-шаге происходит свёртка предварительно обработанных данных. Главный узел получает ответы от рабочих узлов и на их основе формирует результат

## Файловая система HDFS.

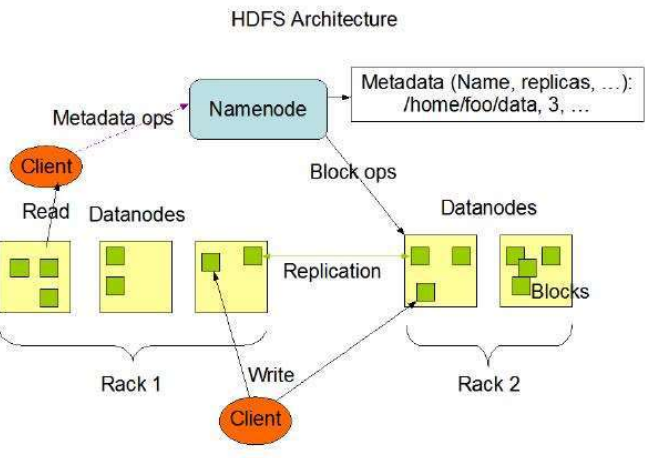
В основе технологии лежит распределённая файловая система HDFS (Hadoop Distributed File System) - распределенная файловая система, созданная для хранения очень большого объема информации (терабайт или даже петабайт) и обеспечения высокой скорости доступа к этой информации. Файлы хранятся в избыточной форме на нескольких машинах для обеспечения их устойчивости при возможных ошибках и высокой доступности параллельным приложениям.

Таким образом, если один или несколько узлов кластера выходят из строя, то риск потери информации сводится к минимуму и кластер продолжает работу в штатном режиме.

Кластер HDFS строится из NameNode и DataNode серверов. Как правило, это один или два NameNode–сервера и множество DataNode–серверов. NameNode–сервер хранит метаданные и используется для быстрого обращения к файлам.

Чтобы открыть файл, клиент обращается к NameNode и получает список расположения блоков, составляющих файл. Эти адреса идентифицируют DataNode, который хранит каждый блок. Затем клиенты считывают информацию напрямую с серверов DataNode, возможно, параллельно. NameNode не принимает непосредственного участия в этом обмене объемами данных, сводя его накладные расходы к минимуму.

Кластер HDFS является горизонтально масштабируемым, что позволяет изменять вычислительные мощности путем добавления новых серверов в кластер.



## Утилиты Hadoop.

**Hbase** — это распределенная, колоночноориентированная, мультиверсионная база типа «ключзначение». Данные организованы в таблицы, проиндексированные первичным ключом, который в Hbase называется RowKey. Для каждого RowKey ключа может храниться неограниченный набор атрибутов (или колонок).

Колонки организованны в группы колонок, называемые Column Family. Как правило в одну Column Family объединяют колонки, для которых одинаковый паттерн использования и хранения.

Для каждого атрибута может храниться несколько различных версий.Разные версии имеют разный timestamp.

Записи физически хранятся в отсортированном по RowKey порядке. При этом данные соответствующие разным Column Family хранятся отдельно, что позволяет при необходимости читать данные только из нужного семейства колонок.

**Hive** Cамая первая и до сих пор одна из самых популярных СУБД на этой платформе. В качестве языка запросов использует HiveQL — урезанный диалект SQL, который, тем не менее, позволяет выполнять довольно сложные запросы над данными, хранимыми в HDFS.

Hive представляет из себя движок, который превращает SQLзапросы в цепочки map-reduce задач. Движок включает в себя такие компоненты, как Parser (разбирает входящие SQL-запрсоы), Optimimer (оптимизирует запрос для достижения большей эффективности), Planner (планирует задачи на выполнение) Executor (запускает задачи на фреймворке MapReduce).

**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. 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.

**Apache Impala** - это открытая исходная база данных для Apache Hadoop . Impala поставляется компанией Cloudera.

Impala обеспечивает быстрые интерактивные SQLзапросы непосредственно на ваших данных Apache Hadoop, хранящихся в HDFS, HBase или Amazon Simple Storage Service (S3).

Помимо использования одной и той же унифицированной платформы хранения, Impala также использует те же метаданные, тот же синтаксис SQL (Hive SQL), драйвер ODBC и пользовательский интерфейс (интерфейс запроса Impala в Hue) как и Apache Hive. Это обеспечивает привычную и унифицированную платформу для запросов в реальном времени или пакетных запросов.

**Pig** Надстройка для Hadoop, позволяющая создавать программы для обрабоки данных на высокоуровневом языке

Pig состоит из двух частей: • язык для описания потоков Pig Latin; • исполнительная среда для запуска сценариев Pig Latin (доступны два варианта: запуск на локальной JVM или исполнение в кластере Hadoop)

Pig упрощает использование Hadoop, позволяя выполнять SQL-подобные запросы к распределенным наборам данных. Характерным свойством программ Pig является то, что их структура поддается существенному распараллеливанию,

**Mahout** Первая большая библиотека, реализовавшая многие популярные алгоритмы средствами MapReduce. Включает в себя алгоритмы для кластеризации, коллаборативной фильтрации, случайных деревьев, а также несколько примитивов для факторизации матриц.

**Rhadoop** Надстройка над Hadoop для выполнения программ, написанных на языке R. Как и Mahout позволяет использовать алгоритмы DataMining.

**Tez** представляет задачу в виде ориентированного ациклического графа (DAG) компонентовобработчиков. Планировщик запускает вычисление графа и при необходимости динамически переконфигурирует его, оптимизируя под данные.

**Oozie** Утилита позволяет создавать workflow из программ MapReduce, запросов Hive и Pig. Oozie — это проект с открытым исходным кодом на основе технологии Java™, упрощающий процесс создания потоков работ и координацию заданий. Oozie предоставляет принципиальную возможность объединения нескольких последовательно выполняемых заданий в одну логическую единицу работы.

**Hue** Web интерфейс к Hadoop для мониторинга заданий, выполняющихся на кластере, и облегчения обслуживания кластера. ϖHUE (Hadoop User Experience) – это веб интерфейс для анализа данных на Hadoop.

**Sqoop** Утилита для загрузки данных из реляционных баз данных в кластер Hadoop и для выгрузки данных обратно БД.

Инструмент для пересылки данных между структурированными хранилищами и HDFS

## Состав модуля YARN.

Корнем иерархии YARN является ResourceManager. Этот демон управляет всем кластером и назначением приложений базовым вычислительным ресурсам. ResourceManager распределяет ресурсы (вычислительные ресурсы, память, пропускная способность и т.д.) для базовых NodeManager (агент YARN на узле). Кроме того, ResourceManager взаимодействует с ApplicationMaster при распределении ресурсов и с NodeManager при запуске и мониторинге базовых приложений.

Каждым экземпляром приложения, которое выполняется в YARN, управляет демон ApplicationMaster. Он отвечает за получение ресурсов из ResourceManager и мониторинг (посредством NodeManager) исполнения и потребления ресурсов контейнеров (распределение ресурсов процессора, памяти и т.д.). Хотя в настоящее время ресурсы являются традиционными (ядра процессора, память), завтра могут появиться новые ресурсы, специфичные для конкретной задачи (например, графические процессоры или специализированные устройства обработки).

NodeManager управляет всеми узлами в кластере YARN. NodeManager предоставляет сервисы на каждом узле кластера, начиная с контроля управления контейнером в течение его жизненного цикла и заканчивая мониторингом ресурсов и отслеживанием состояния узла. В отличие от MRv1, управлявшей задачами map и reduce через слоты, NodeManager управляет абстрактными контейнерами, которые представляют собой ресурсы на узле, доступные для конкретного приложения. YARN по-прежнему использует слой HDFS с главными узлами NameNode для сервисов метаданных и DataNode для сервисов хранения, реплицированных по всему кластеру.

## Принципы работы с BigData.

Горизонтальная масштабируемость: так как данных может быть много, то и система, в которой они хранятся должна быть расширяемой. Если объем данных вырос в 2 раза, то и количество кластеров увеличивается в 2 раза.

Отказоустойчивость: горизонтальная масштабируемость подразумевает тот факт, что машин в кластере большое количество. И естественно эти машины будут по тем или иным причинам выходить из строя. К примеру, Hadoopкластер Yahoo насчитывает более 42000 машин. Методы работы с BigData должны учитывать этот фактор и продолжать работать без видимых потерь.

Локальность данных: в больших системах данные распределены на большом количестве машин. Если данные находятся на одной машине, а обрабатываются на другой, то расходы на передачу этих данных могут и вовсе превысить расходы на обработку. Поэтому важным вопросом в проектировании BigData стоит принцип локальности данных, обработке информации там же, где она хранится.

## Платформа Oracle для BigData.

Oracle официально представила систему Big Data Appliance в ходе конференции OpenWorld в СанФранциско в 2013 году. Это комплексное решение, которое пополнило новейшую линейку корпорации, созданную на стыке программных продуктов Oracle и серверных платформ, которые достались ей после покупки Sun.

Полный список ПО следующий

* Oracle Linux 5.6
* Hotspot Java VM
* Cloudera’s Distribution including Apache Hadoop with:
* Hadoop Core
* HDFS
* Hive
* HBase
* Zookeeper
* Oozie
* Mahout
* Sqoop
* Cloudera Manager
* Oracle R open source
* MySQL Standard Edition (restricted use) to support metadata and data for Hadoop tools and for ODI
* Oracle NoSQL Database Community Edition

## Кластер Oracle Big data Appliance

Oracle Big Data Appliance представляет собой оптимизированный комплекс, объединяющий аппаратное и программные продукты, включаяCloudera’s Distribution с Apache Hadoop и Cloudera Manager, а также дистрибутив среды программирования R с открытым исходным кодом.

Hadoop Distributed File System Oracle NoSQL Database



Из этого рисунка можно видеть, что структурированная информация накапливается (вводится) в обычной СУБД Oracle. Эта часть обычно есть в любой компании.

Есть менее структурированная – она имеет вид ключ-значение. Ключ – это идентификатор, по которому осуществляется доступ, а значение – все что угодно: учетная запись пользователя, документ, скалярное значение и т.д. Такую информацию мы храним в базе данных Oracle NoSQL.

Есть также полностью неструктурированная информация абсолютно любого вида складируется в файловой системе. Она хранится в “обычной”, но распределенной файловой системе, что дает возможность использовать эти данные с любого узла в кластере Big Data. Эта файловая система – HDFS.

Все это, с точки зрения ETL, — источники данных: плоские файлы, базы данных и т.д. Фактически в этой части ничего нового в Big Data не появляется.

Наибольший интерес, с точки зрения обработки информации, представляет уровень организации данных. Вот здесь, как раз, Hadoop играет ключевую роль. Его задача опустить предварительную обработку данных на уровень узла, хранящего эту информацию.

предварительная обработка данных ложится на плечи Hadoop, а задачи по анализу данных решает СУБД Oracle, которую мы используем в качестве хранилища уже подготовленных данных. Oracle предоставляет следующим набор инструментов для анализа:

* Oracle R Enterprise
* In-Database Data Mining
* In-Database Text Mining
* In-Database Semantic Analysis
* In-Database Spatial

Есть также возможность перенести анализ данных с уровня СУБД Oracle на уровень Hadoop. Делается это за счет модуля In-Database MapReduce (на рисунке обозначен Oracle MR).

## Кластер Oracle Exadata.

Exadata и SuperCluster как машины для СУБД Oracle позволяют строить огромные по объему хранилища данных. Существует множество примеров построения хранилищ на сотни терабайт и даже петабайт данных. Big Data Appliance (BDA), который тоже нужен для хранения и обработки очень больших объемов данных с использованием таких технологий, как Hadoop и Oracle NoSQL Database.

Зачастую обработка “больших данных” является довольно простым, с точки зрения алгоритмов, процессом, но проблема именно в том, что данных очень много, а на BDA эти данные обработать будет гораздо дешевле. Почему? СУБД Oracle развивается в течение более чем 35 лет и содержит тысячи самых разных возможностей. Но, если ваша задача заключается, например, в хранении, фильтрации и агрегации логов вебсерверов (типичная задача из области Big Data), то для такой задачи, возможно, не будет использоваться даже сотая доля возможностей СУБД Oracle, несмотря на то, что данных может быть многие петабайты. Такие технологии, как Hadoop, не имеют очень многих возможностей реляционных баз, но позволяют осуществить обработку в десятки раз дешевле.

Поэтому Oracle рекомендует первичную, грубую обработку данных производить при помощи Hadoop, а уже более сложный анализ очищенных и агрегированных данных – делать в СУБД Oracle, в том числе с использованием Exadata и SuperCluster.

Аппаратноеобеспечение

* Кластер из 18 узлов (18 Sun X4270 M2 )
* На каждом узле ⎥ 64 GB RAM (всего 1152GB RAM) ⎥ 16 ядер Intel (всего 288 ядер) ⎥ 48 ТБ дисков (всего 864TB)
* 40 Gb p/sec InfiniBand 1
* 10 Gb p/sec Ethernet

## Кластер Oracle Exalytics.

Exalytics — аппаратно-программный комплекс аналитической обработки в оперативной памяти,

Является узлом архитектуры x86-64 с предустановленными специализированными версиями реляционной [системы управления базами данных в оперативной памяти](https://ru.wikipedia.org/w/index.php?title=%D0%91%D0%B0%D0%B7%D0%B0_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85_%D0%B2_%D0%BE%D0%BF%D0%B5%D1%80%D0%B0%D1%82%D0%B8%D0%B2%D0%BD%D0%BE%D0%B9_%D0%BF%D0%B0%D0%BC%D1%8F%D1%82%D0%B8&action=edit&redlink=1) [Timesten](https://ru.wikipedia.org/w/index.php?title=Timesten&action=edit&redlink=1" \o "Timesten (страница отсутствует)) и многомерной системы управления базами данных Essbase, а также с аналитическим пакетом линейки [Oracle BI](https://ru.wikipedia.org/w/index.php?title=Oracle_Business_Intelligence&action=edit&redlink=1" \o "Oracle Business Intelligence (страница отсутствует))

В отличие от программно-аппаратных комплексов для хранилищ данных и средств бизнес-анализа в оперативной памяти, Oracle Exalytics представляет собой первый интегрированный программно-аппаратный комплекс, который включает в себя программное обеспечение и оборудование для анализа данных в оперативной памяти, а также оптимизированную [BI](http://www.tadviser.ru/index.php/BI)-платформу с расширенными возможностями визуализации.

Oracle Exalytics спроектирован как единое решение одного поставщика на основе широко распространенной платформы бизнес-анализа Oracle BI и системы управления базами данных в оперативной памяти, сконфигурированных для аппаратных средств, которые разрабатываются, изготавливаются и поддерживаются корпорацией Oracle.

Oracle Exalytics предлагает инновационные возможности:

* Интерактивную визуализацию «со скоростью мысли», которая позволяет исследовать и анализировать большие наборы данных, легко выявляя устойчивые закономерности, тренды и исключения;
* Возможность анализировать данные для многих тысяч пользователей мобильных устройств Apple iPad, благодаря которой анализ становится доступным любому человеку независимо от его местонахождения;
* Адаптивная технология обработки данных в оперативной памяти, оптимизирующая производительность для обеспечения минимального времени отклика даже в тех случаях, когда объем данных превышает доступную комплексу Exalytics память.

## Java Hotspot VM.

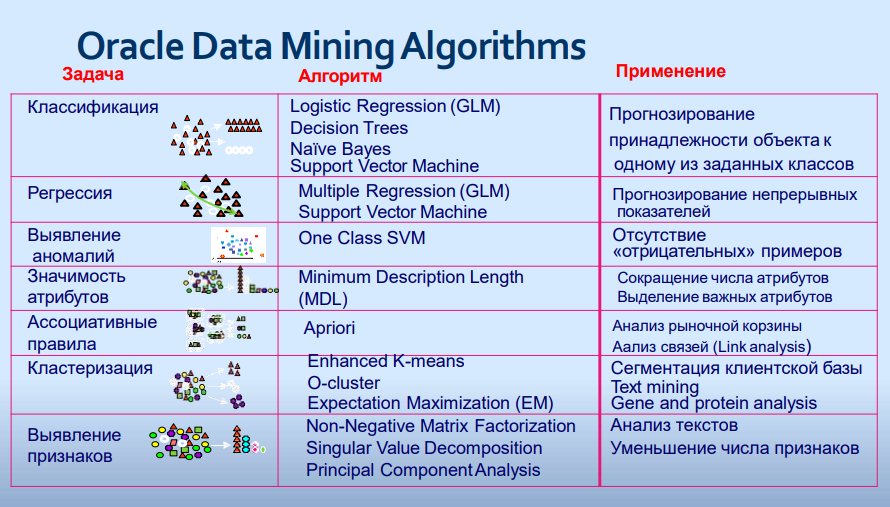
«HotSpot» — это основная виртуальная машина Java (JVM) как для клиентских, так и для серверных компьютеров, выпускаемая корпорацией «Oracle». Для повышения производительности обладает технологиями динамической компиляции JIT и адаптивной оптимизации.

Виртуальная машина Java HotSpot — это основной компонент платформы Java SE. Он реализует спецификацию виртуальной машины Java и поставляется как разделяемая библиотека в среде выполнения Java. В качестве механизма выполнения байт-кода Java он предоставляет средства среды выполнения Java, такие как синхронизация потоков и объектов, в различных операционных системах и архитектурах. Он включает динамические компиляторы, которые адаптивно компилируют байт-коды Java в оптимизированные машинные инструкции и эффективно управляют кучей Java с помощью сборщиков мусора, оптимизированных как для малого времени паузы, так и для пропускной способности. Он предоставляет данные и информацию инструментам и приложениям профилирования, мониторинга и отладки.

Он включает

* Загрузчик Java-классов;
* Интерпретатор [байткода](https://ru.wikipedia.org/wiki/%D0%91%D0%B0%D0%B9%D1%82%D0%BA%D0%BE%D0%B4" \o "Байткод);
* Две версии виртуальной машины — Client и Server;
* Несколько сборщиков мусора;
* Набор вспомогательных библиотек времени выполнения.
* Client-версия виртуальной машины характеризуется меньшим временем запуска приложений и меньшим потреблением памяти по сравнению с Server-версией, уступая при этом последней в производительности.

## Oracle Data Mining Algorithms.



## Oracle Endeca Information Discovery

Oracle Endeca Information Discovery - это комплексное решение для гибкого обнаружения данных в масштабах всего предприятия, позволяющее бизнес-пользователям внедрять инновации в сочетании с управлением и контролем ИТ. Это позволяет принимать более быстрые и надежные решения, в то же время облегчая нагрузку на ИТ.

Endeca Information Discovery состоит из трех основных компонентов:

* [Endeca Information Discovery Studio](https://docs.oracle.com/cd/E49933_01/index_studio_tab.htm) - это[лучшая](https://docs.oracle.com/cd/E49933_01/index_studio_tab.htm) в отрасли среда для компоновки приложений и возможностей обнаружения, которая позволяет бизнес-пользователям легко загружать и объединять несколько различных источников данных, а затем быстро настраивать приложения для обнаружения - все в контексте корпоративной инфраструктуры, которая поддерживает существующее управление и определения предприятия.

Studio включает поиск мирового класса, управляемую навигацию и фильтрацию, а также предлагает множество мощных интерактивных визуализаций для быстрого интуитивного анализа, не требующего никакого обучения.

* [Endeca Server](https://docs.oracle.com/cd/E49933_01/index_server_tab.htm) - это гибридная поисково-аналитическая машина, которая организует сложные и разнообразные данные из разрозненных источников.

В основе Endeca Information Discovery уникальная модель данных, подобная NoSQL, и архитектура в памяти Endeca Server создают чрезвычайно гибкую структуру для обработки сложных комбинаций данных, устраняя необходимость в сложном предварительном моделировании и предлагая исключительную производительность при масштабировании. Endeca Server также поддерживает 35 различных языков.

[Endeca Information Discovery Integrator](https://docs.oracle.com/cd/E49933_01/index_integrator_tab.htm) - это мощная среда интеграции визуальных данных, которая включает систему сбора информации (IAS) для сбора контента из файловых систем, систем управления контентом и веб-сайтов; и готовый к работе ETL, специально созданный для включения данных из широкого спектра источников, включая Oracle BI Server.

Кроме того, Oracle Endeca Web Acquisition Toolkit - это графический веб-инструмент ETL, продаваемый как дополнительный модуль.

## SAP HANA Data Platform для Big Data.

SAP HANA (High-performance ANalytic Appliance) – это программноаппаратный комплекс, в основе которого лежит технология хранения и обработки данных in-memory

In-Memory Software поставляется в комплекте с оборудованием Hardware-партнера (HP, IBM, Huawei, CISCO, Fujitsu, Hitachi, Dell,)

Данные хранятся и обрабатываются непосредственно в оперативной памяти

Поколоночное хранение данных позволяют оптимизировать выборку по запросу и существенно увеличить производительность системы

Инструменты моделирования, управления жизненным циклом информации, обеспечения безопасности, поддержки масштабируемости и управляемости

Отсутствие агрегатов БД позволяет получать доступ к детальным данным в режиме реального времени

Сжатие данных за счет хранения и обработки только значимого содержимого Технологическая платформа для любых типов приложений

## Технологии OLTP BI и OLAP в хранилищах SAP HANA.

OLTP (англ. Online Transaction Processing) или реляционные БД- это базы данных, которые используются везде и повсеместно. Их основная цель - ввод/редактирование/удаление данных в режиме онлайн. Примеры использования: мессенджеры, социальные сети, 1С: Бухгалтерия и т.д.

OLAP (англ. Online Analytical Processing) или многомерные БД - это базы данных, которые служат непосредственно для проведения быстрого анализа больших объемов данных. Обычно такие БД используются на больших предприятиях для построения аналитической отчетности за большой промежуток времени (месяц, квартал, год). Такая информация в основном используется для анализа прошедшего периода и планирования будущего. Основные пользователи аналитических данных - руководители.

Причина использования OLAP (интерактивная аналитическая обработка) для обработки запросов — скорость. [Реляционные базы данных](https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D0%BB%D1%8F%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D1%8B%D0%B5_%D0%B1%D0%B0%D0%B7%D1%8B_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85) хранят сущности в отдельных таблицах, которые обычно хорошо нормализованы. Эта структура удобна для операционных баз данных (системы [OLTP](https://ru.wikipedia.org/wiki/OLTP)), но сложные многотабличные запросы в ней выполняются относительно медленно.

OLAP-структура, созданная из рабочих данных, называется [OLAP-куб](https://ru.wikipedia.org/wiki/OLAP-%D0%BA%D1%83%D0%B1). Куб создаётся из соединения таблиц с применением [схемы звезды](https://ru.wikipedia.org/wiki/%D0%A1%D1%85%D0%B5%D0%BC%D0%B0_%D0%B7%D0%B2%D0%B5%D0%B7%D0%B4%D1%8B) или [схемы снежинки](https://ru.wikipedia.org/wiki/%D0%A1%D1%85%D0%B5%D0%BC%D0%B0_%D1%81%D0%BD%D0%B5%D0%B6%D0%B8%D0%BD%D0%BA%D0%B8). В центре схемы звезды находится [таблица фактов](https://ru.wikipedia.org/wiki/%D0%A2%D0%B0%D0%B1%D0%BB%D0%B8%D1%86%D0%B0_%D1%84%D0%B0%D0%BA%D1%82%D0%BE%D0%B2), которая содержит ключевые факты, по которым делаются запросы.

* многомерная OLAP (Multidimensional OLAP — MOLAP);
* реляционная OLAP (Relational OLAP — ROLAP);
* гибридная OLAP (Hybrid OLAP — HOLAP).

С точки зрения реализации, делятся на «физическую OLAP» и «виртуальную» (реляционную, [англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) Relational OLAP, ROLAP). «Физическая», в свою очередь, в зависимости от реализации подразделяется на многомерную ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) Multidimensional OLAP, MOLAP) и гибридную — ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) Hybrid OLAP, HOLAP).

В первом случае наличествует программа, выполняющая на этапе предварительной загрузки данных в OLAP предварительный расчёт агрегатов (вычислений по нескольким исходным значениям, например «итог за месяц»), которые затем сохраняются в специальную многомерную базу данных, обеспечивающую быстрое извлечение и экономичное хранение.

Гибридная реализация является комбинацией: сами данные хранятся в реляционной базе данных, а агрегаты — в многомерной.

В ROLAP-реализациях все данные хранятся и обрабатываются в реляционных системах управления базами данных, а агрегаты могут не существовать вообще или создаваться по первому запросу к базе данных или кэше аналитического программного обеспечения.

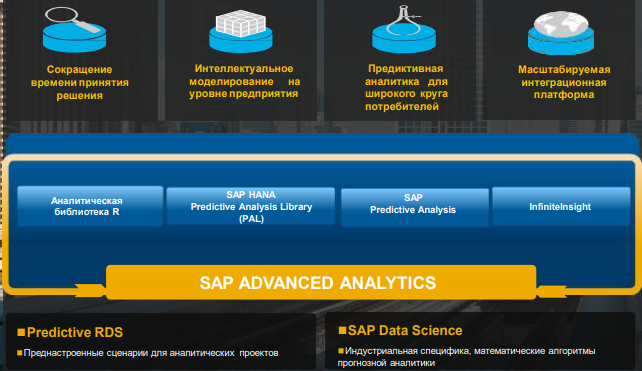


## Задачи аналитики в SAP HANA.

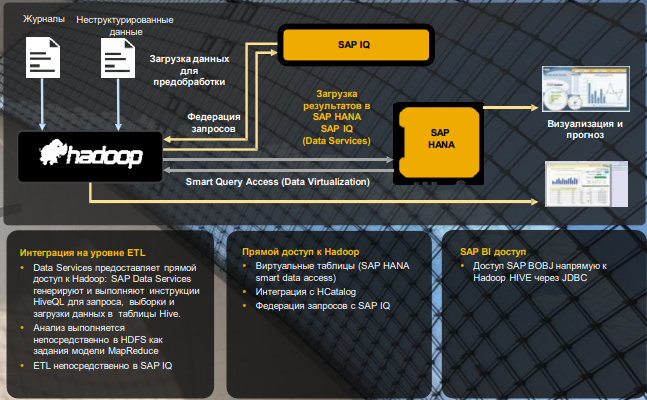


## Аналитика на уровне СУБД в SAP HANA.

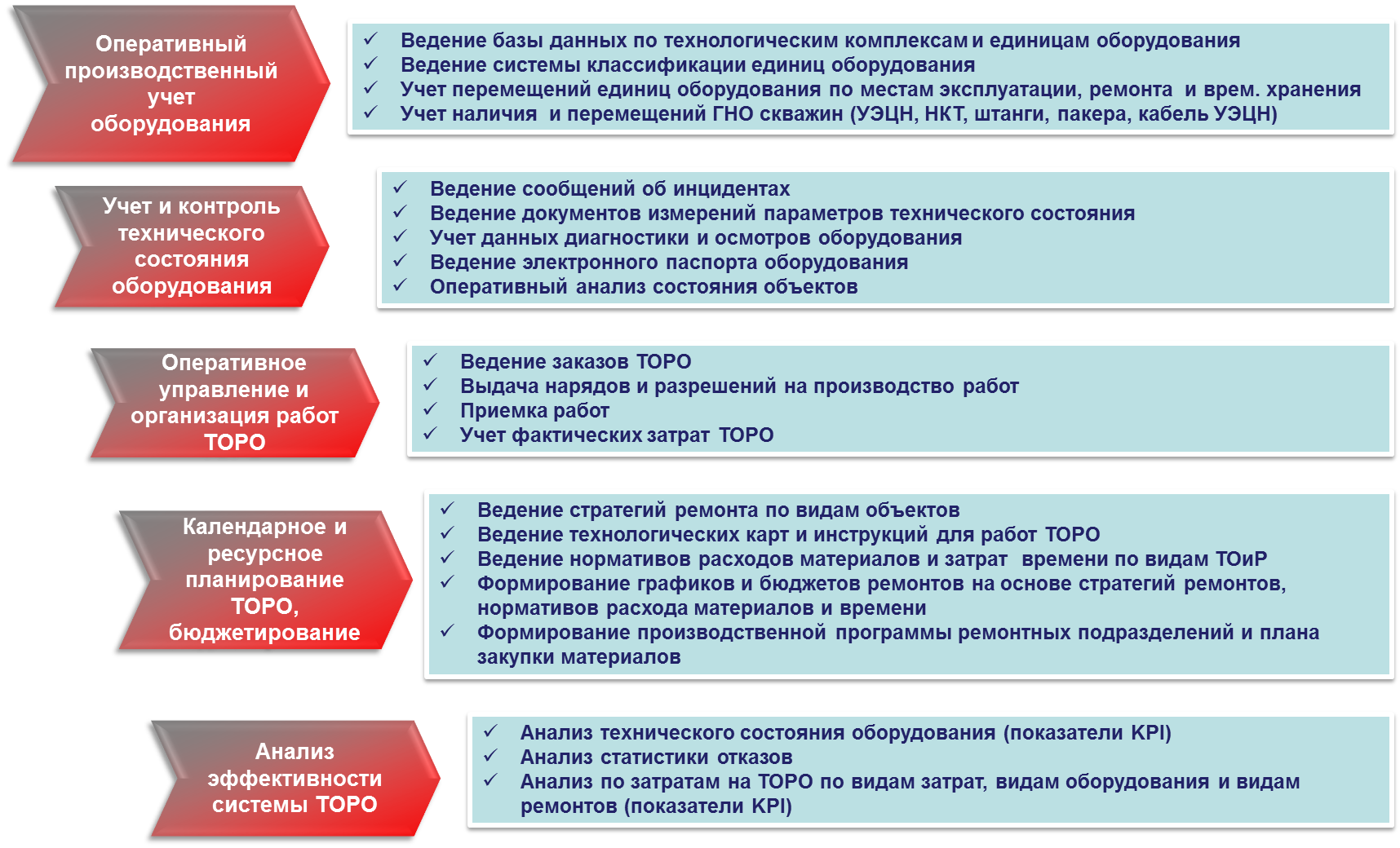
О том, зачем нужна бизнес-аналитика как таковая, написано много – общее состоит в том, что она необходима для снижения рисков принятия ошибочных решений в быстро меняющемся мире. Важно знать, где аналитик встретится с Большими Данными, — сегодня, как правило, это комплексы, отслеживающие состояние и тренды поведения рукотворных и нерукотворных больших, взаимосвязанных стохастических систем.



## Интеграционная модель SAP HANA.



## Типовые задачи для SAP ESP.



## Принципы построения модели данных в HBase.

Hbase — это распределенная, колоночноориентированная, мультиверсионная база типа «ключзначение». Данные организованы в таблицы, проиндексированные первичным ключом, который в Hbase называется RowKey. Для каждого RowKey ключа может храниться неограниченный набор атрибутов (или колонок).

Колонки организованны в группы колонок, называемые Column Family. Как правило в одну Column Family объединяют колонки, для которых одинаковый паттерн использования и хранения.

Для каждого атрибута может храниться несколько различных версий. Разные версии имеют разный timestamp.

Записи физически хранятся в отсортированном по RowKey порядке. При этом данные, соответствующие разным Column Family хранятся отдельно, что позволяет при необходимости читать данные только из нужного семейства колонок.

При удалении определённого атрибута физически он сразу не удаляется, а лишь маркируется специальным флажком tombstone. Физическое удаление данных произойдет позже, при выполнении операции Major Compaction.

Атрибуты, принадлежащие одной группе колонок и соответствующие одному ключу, физически хранятся как отсортированный список. Любой атрибут может отсутствовать или присутствовать для каждого ключа, при этом если атрибут отсутствует — это не вызывает накладных расходов на хранение пустых значений.

Список и названия групп колонок фиксирован и имеет четкую схему. На уровне группы колонок задаются такие параметры как time to live (TTL) и максимальное количество хранимых версий. Если разница между timestamp для определенной версии и текущим временем больше TTL — запись помечается к удалению. Если количество версий для определённого атрибута превысило максимальное количество версий — запись также помечается к удалению.

## Архитектура HBase.

HBase - это система с открытым исходным кодом, распределенная система хранения данных с ключом-значением и ориентированная на столбцы база данных с высокой скоростью записи и низкой скоростью произвольного чтения. Используя HBase, мы можем проводить онлайн аналитику в реальном времени. Архитектура HBase обладает сильной случайной читабельностью. В HBase данные физически переносятся в так называемые регионы. Каждый регион размещается на одном сервере региона, и один или несколько регионов отвечают за каждый сервер региона.

Архитектура HBase состоит из главных и подчиненных серверов. Кластер HBase имеет один главный узел, называемый HMaster, и несколько серверов регионов, называемых HRegion Server (HRegion Server). Существует несколько регионов - регионов на каждом региональном сервере.

Узел **HMaster** является легковесным и используется для назначения региона региону сервера.

Есть несколько основных обязанностей Hmaster:

Выполнение некоторых задач администрирования, включая загрузку, балансировку, создание данных, обновление, удаление и т. Д.

Отвечает за изменения в схеме или изменения в данных META в соответствии с указаниями клиентского приложения.

Большая часть работы DDL над таблицами HBase выполняется HMaster.

**Региональные серверы** - это рабочие узлы, которые обрабатывают запросы клиентов на чтение, запись, обновление и удаление. Регион-сервер является легковесным, он работает на всех узлах кластера Hadoop. Основная задача сервера региона - сохранять данные по областям и выполнять запросы клиентов.

**HDFS** расшифровывается как распределенная файловая система Hadoop. Он сохраняет каждый файл в нескольких блоках и реплицирует блоки в кластере Hadoop для обеспечения отказоустойчивости. HDFS обеспечивает высокую отказоустойчивость и работает с недорогими материалами.

**Zookeeper** - проект с открытым исходным кодом. HMaster и HRegionServers регистрируются в ZooKeeper.

Он предоставляет различные услуги, такие как поддержание информации о конфигурации, присвоение имен, обеспечение распределенной синхронизации и т. Д.

**Преимущества архитектуры HBase**

* Они могут хранить большие наборы данных
* Мы можем поделиться базой данных
* Гигабайт в петабайт экономически эффективным
* Высокая доступность благодаря репликации и сбоям

**Недостатки архитектуры HBase**

* Структура SQL не поддерживает
* Не поддерживает транзакцию
* Только с ключом отсортировано
* Проблемы с памятью кластера

## Опишите основные процессы HBase.

Region Server — обслуживает один или несколько регионов. Регион — это диапазон записей соответствующих определенному диапазону подряд идущих RowKey. Каждый регион содержит:

Persistent Storage — основное хранилище данных в Hbase. Данные физически хранятся на HDFS, в специальном формате HFile. Данные в HFile хранятся в отсортированном по RowKey порядке. Одной паре (регион, column family) соответствует как минимум один HFIle.

MemStore — буфер на запись. Так как данные хранятся в HFile в отсортированном порядке — обновлять HFile на каждую запись довольно дорого. Вместо этого данные при записи попадают в специальную область памяти MemStore, где накапливаются некоторое время. При наполнении MemStore до некоторого критического значения данные записываются в новый HFile.

BlockCache — кэш на чтение. Позволяет существенно экономить время на данных которые читаются часто.

Write Ahead Log(WAL). Так как данные при записи попадают в memstore, существует некоторый риск потери данных из-за сбоя. Для того чтобы этого не произошло все операции перед собственно осуществление манипуляций попадают в специальный лог-файл. Это позволяет восстановить данные после любого сбоя.

Master Server — главный сервер в кластере hbase. Master управляет распределением регионов по Region Servers, ведет реестр регионов, управляет запусками регулярных задач и делает другую полезную работу.

При увеличении количества данных в регионе и достижении им определенного размера Hbase запускает split, операцию разбивающую регион на 2. Для того чтобы избежать постоянных делений регионов — можно заранее задать границы регионов и увеличить их максимальный размер.

Так как данные по одному региону могут храниться в нескольких HFile, для ускорения работы Hbase периодически их сливает воедино. Эта операция в Hbase называется compaction. Compaction бывают двух видов:

Minor Compaction. Запускается автоматически, выполняется в фоновом режиме. Имеет низкий приоритет по сравнению с другими операциями Hbase.

Major Compaction. Запускается руками или по наступлению срабатыванию определенных триггеров (например по таймеру). Имеет высокий приоритет и может существенно замедлить работу кластера.

## Архитектура Hive.

Hive представляет из себя движок, который превращает SQLзапросы в цепочки map-reduce задач. Движок включает в себя такие компоненты, как Parser (разбирает входящие SQL-запрсоы), Optimimer (оптимизирует запрос для достижения большей эффективности), Planner (планирует задачи на выполнение) Executor (запускает задачи на фреймворке MapReduce).

Для работы hive также необходимо хранилище метаданных. Дело в том что SQL предполагает работу с такими объектами как база данных, таблица, колонки, строчки, ячейки и тд. Поскольку сами данные, которые использует hive хранятся просто в виде файлов на hdfs — необходимо где-то хранить соответствие между объектами hive и реальными файлами.

В качестве metastorage используется обычная реляционная СУБД, такая как MySQL, PostgreSQL или Oracle.

## Основные объекты Hive.

При работе с hive можно выделить следующие объекты которыми оперирует hive:

* База данных
* Таблица
* Партиция (partition)
* Бакет (bucket) Разберем каждый из них подробнее:

База данных представляет аналог базы данных в реляционных СУБД. База данных представляет собой пространство имён, содержащее таблицы.

Команда создания новой базы данных выглядит следующим образом: CREATE DATABASE|SCHEMA [IF NOT EXISTS] ϖ Database и Schema в данном контексте это одно и тоже. Необязательная добавка IF NOT EXISTS создает базу данных только в том случае если она еще не существует.

Пример создания базы данных: CREATE DATABASE userdb;

Для переключения на соответствующую базу данных используем команду USE: USE userdb;

Таблица в hive представляет из себя аналог таблицы в классической реляционной БД. Основное отличие — что данные hive таблиц хранятся просто в виде обычных файлов на hdfs. Это могут быть обычные текстовые csv-файлы, бинарные sequenceфайлы, более сложные колоночные paruqet-файлы и другие форматы. Но в любом случае данные, над которыми настроена hive-таблица очень легко прочитать и не из hive

Так как hive представляет из себя движок для трансляции SQLзапросов в mapreduce-задачи, то обычно даже простейшие запросы к таблице приводят к полному сканированию данных в этой таблицы. Для того чтобы избежать полного сканирования данных по некоторым из колонок таблицы можно произвести партиционирование этой таблицы. Это означает, что данные относящиеся к разным значениям будут физически храниться в разных папках на HDFS

Партиционирование помогает сократить время обработки, если обычно при запросах известны ограничения на значения какоголибо столбца. Однако оно не всегда применимо. Например — если количество значений в столбце очень велико.

Например — это может быть ID пользователя в системе, содержащей несколько миллионов пользователей.

В этом случае на помощь нам придет разделение таблицы на бакеты. В один бакет попадают строчки таблицы, для которых значение совпадает со значением хэш-функции вычисленным по определенной колонке.

При любой работе с бакетированными таблицами необходимо не забывать включать поддержку бакетов в hive (иначе hive будет работать с ними как с обычными таблицами)

## Опишите процедуры партицирования и бакетирования в Hive.

Mapreduce-задачи, то обычно даже простейшие запросы к таблице приводят к полному сканированию данных в этой таблицы. Для того чтобы избежать полного сканирования данных по некоторым из колонок таблицы можно произвести партиционирование этой таблицы. Это означает, что данные относящиеся к разным значениям будут физически храниться в разных папках на HDFS

Партиционирование помогает сократить время обработки, если обычно при запросах известны ограничения на значения какоголибо столбца. Однако оно не всегда применимо. Например — если количество значений в столбце очень велико.

Например — это может быть ID пользователя в системе, содержащей несколько миллионов пользователей.

В этом случае на помощь нам придет разделение таблицы на бакеты. В один бакет попадают строчки таблицы, для которых значение совпадает со значением хэш-функции вычисленным по определенной колонке.

При любой работе с бакетированными таблицами необходимо не забывать включать поддержку бакетов в hive (иначе hive будет работать с ними как с обычными таблицами)

## Организация обработки запросов в Impala.

1. Пользовательские приложения отправляют SQL -запросы в Impala через ODBC или JDBC, которые предоставляют стандартизованные интерфейсы запросов. Пользовательское приложение может подключаться к любому impalad в кластере. Этот impalad становится координатором запроса.
2. Impala анализирует запрос, чтобы определить, какие задачи должны выполняться impalad экземплярами в кластере. Выполнение запланировано для оптимальной эффективности.
3. Доступ impalad к данным, таким как HDFS и HBase, осуществляется локальными экземплярами для предоставления данных.
4. Каждый impalad возвращает данные координирущей impalad, которая отправляет эти результаты клиенту.

## Опишите основные функции Impala.

Наиболее распространенные функции SQL-92 для языка запросов HiveQL (HiveQL), включая функции SELECT, joins и aggregate functions.

HDFS, HBase и Amazon Simple Storage System (S3) , в том числе:

1. Форматы файлов HDFS: delimited text files, Parquet, Avro, SequenceFile и RCFile.

2. Кодеки сжатия: Snappy, GZIP, Deflate, BZIP.

Общие интерфейсы доступа к данным:

1. Драйвер JDBC .

2. Драйвер ODBC .

3. Hue Beeswax и пользовательский интерфейс Impala Query.

Интерфейс командной строки impala-shell.

Проверка подлинности Kerberos (аутентификация).

## Опишите операторы Pig Latin.

Pig Latin – относительно простой язык, выполняющий операторы. Оператор – это процедура, принимающая входные данные (например, массив, состоящий из набора записей) и формирующая на выходе другой массив. Массив – это структура, подобная таблице реляционной базы данных, в которой записи аналогичны строкам таблицы и состоят из полей.

Операторы Pig Latin работают с отношениями (и называются реляционными операторами). Ниже приведены основные реляционные операторы Pig:

* FILTER - выбор набора записей из массива в соответствии с заданным условием.
* FOREACH - выполнение итерации по записям массива и преобразование данных.
* GROUP - группировка данных в один или несколько массивов.
* JOIN - соединение двух или нескольких массивов (внутреннее или внешнее соединение).
* LOAD - загрузка данных из файловой системы.
* ORDER - сортировка массива по одному или нескольким полям.
* SPLIT - разделение массива на два или более массива.
* STORE - сохранение данных в файловой системе.

## Основные части скрипта на Pig Latin.

Скрипт состоит из трех частей: загрузка данных, обработка и сохранение.

Такой порядок является общим для большинства задач.

В некоторых случаях решение задач может включать и дополнительные этапы — например, генерацию данных (например, структурированных искусственных данных для проверки алгоритма) или сохранение промежуточных результатов вычислений.

Подробное описание синтаксиса, типов данных и операторов вы можете найти в официальной документации.

Pig Latin результатом любого оператора является отношение, представляющее собой набор кортежей.

Оператор LOAD создает отношение records из файлов в HDFS из директории ’/log/flume/events/19-02-20/’, используя стандартный интерфейс PigStorage (также укажем, что разделителем в файлах является символ табуляции ‘\t’).

Каждая строка из файлов предстанет кортежем в отношении. Секция AS присваивает полям в кортеже типы и имена, по которым нам будет удобнее к ним обращаться.

В переводе с Pig Latin на естественный язык приведенный скрипт выглядит так: для каждой записи (FOREACH), из records, сгруппированных вместе (GROUP ALL), выполнить подсчет записей в records (GENERATE COUNT).

**Сохранение**

1. Этап Map: процессоры и диски на каждом узле заняты обработкой своих частей данных.
2. Этап Reduce: результаты, полученные на первом этапе, передаются по сети и объединяются.
3. На третьем этапе результаты сохраняются в файловой системе (на графике виден скачок записи в HDFS).

## Опишите состав библиотеки Mahout.

Mahout содержит ряд моделей и алгоритмов, многие из которых все еще в разработке или экспериментальной фазе

Рекомендательные системы – это наиболее узнаваемая модель машинного обучения из используемых сегодня. Вы видите сервисы или сайты которые пытаются рекомендовать книги или фильмы, или статьи, базируясь на ваших предыдущих действиях.

Кластеризация менее очевидна, но оказывается в не менее известных упоминаниях. Как следует из названия, методы кластеризации пытаются группировать большие числа предметов вместе в кластеры, которые имеют общее сходство. Таким образом устанавливают иерархию и порядок в больших или трудных для понимания множествах данных, и таким способом устанавливают интересные закономерности или делают набор данных более легким для понимания.

Классификация помогает решить соответствует ли новая часть вводных данных или предмет предыдущим рассмотренным шаблонам; и она часто используется для классификации поведения или шаблона. Это может быть использовано для обнаружения подозрительной сетевой активности или мошенничества

## Опишите концепцию Spark.

В Spark вводится концепция RDD (устойчивый распределенный набор данных) – неизменяемая отказоустойчивая распределенная коллекция объектов, которые можно обрабатывать параллельно.

В RDD могут содержаться объекты любых типов. RDD создается путем загрузки внешнего набора данных или распределения коллекции из основной программы (driver program). В RDD поддерживаются операции двух типов:

Трансформации – это операции (например, отображение, фильтрация, объединение и т.д.), совершаемые над RDD; результатом трансформации становится новый RDD, содержащий ее результат.

Действия – это операции (например, редукция, подсчет и т.д.), возвращающие значение, получаемое в результате некоторых вычислений в RDD.

Трансформации в Spark осуществляются в «ленивом» режиме — то есть, результат не вычисляется сразу после трансформации.

Вместо этого они просто «запоминают» операцию, которую следует произвести, и набор данных (напр., файл), над которым нужно совершить операцию. Вычисление трансформаций происходит только тогда, когда вызывается действие, и его результат возвращается основной программе.

Благодаря такому подходу повышается эффективность Spark. Например, если большой файл был преобразован различными способами и передан первому действию, то Spark обработает и вернет результат лишь для первой строки, а не станет прорабатывать таким образом весь файл.

## Какие типы задач выполняет Oozie?

Oozie Утилита позволяет создавать workflow из программ MapReduce, запросов Hive и Pig.

Oozie — это проект с открытым исходным кодом на основе технологии Java, упрощающий процесс создания потоков работ и координацию заданий. Oozie предоставляет принципиальную возможность объединения нескольких последовательно выполняемых заданий в одну логическую единицу работы.

На практике существует 3 различных типа заданий Oozie.

* Workflow – это Ориентированные Ацикличные Графы действий (DAGs action). Или, по-русски говоря, это просто какая-то задача (Map Reduce задача, YARN задача, Spark задача, задача по работе с HDFS и т.п.);
* Coordinator – это Workflow, с заданным временем/частотой запуска;
* Bundle – это высший уровень абстракции в Oozie. Представляет собой набор из Coordinators, не обязательно связанных между собой.

## Опишите состав потока работ Oozie.

**Oozie**

Поток работ Oozie представляет собой коллекцию действий, организованных в виде ориентированного ациклического графа (directed acyclic graph, DAG). Этот граф может содержать два типа узлов: узлы управления (control node) и узлы действия (action node).

Узлы управления, которые используются для определения временной последовательности заданий, предоставляют правила для начала и окончания потока работ, а также управляют последовательностью выполнения потока работ с помощью точек принятия возможных решений, в качестве которых используются узлы ветвления (fork node) и узлы соединения (join node).

Узлы действия используются для инициирования выполнения (включения) задач. В частности, узел действия может представлять собой MapReduce-задание, Pig-приложение, задачу файловой системы или Java-приложение.

## Архитектура Yarn.

Корнем иерархии YARN является ResourceManager. Этот демон управляет всем кластером и назначением приложений базовым вычислительным ресурсам. ResourceManager распределяет ресурсы (вычислительные ресурсы, память, пропускная способность и т.д.) для базовых NodeManager (агент YARN на узле). Кроме того, ResourceManager взаимодействует с ApplicationMaster при распределении ресурсов и с NodeManager при запуске и мониторинге базовых приложений.

Каждым экземпляром приложения, которое выполняется в YARN, управляет демон ApplicationMaster. Он отвечает за получение ресурсов из ResourceManager и мониторинг (посредством NodeManager) исполнения и потребления ресурсов контейнеров (распределение ресурсов процессора, памяти и т.д.). Хотя в настоящее время ресурсы являются традиционными (ядра процессора, память), завтра могут появиться новые ресурсы, специфичные для конкретной задачи (например, графические процессоры или специализированные устройства обработки).

NodeManager управляет всеми узлами в кластере YARN. NodeManager предоставляет сервисы на каждом узле кластера, начиная с контроля управления контейнером в течение его жизненного цикла и заканчивая мониторингом ресурсов и отслеживанием состояния узла. В отличие от MRv1, управлявшей задачами map и reduce через слоты, NodeManager управляет абстрактными контейнерами, которые представляют собой ресурсы на узле, доступные для конкретного приложения. YARN по-прежнему использует слой HDFS с главными узлами NameNode для сервисов метаданных и DataNode для сервисов хранения, реплицированных по всему кластеру.

Использование кластера YARN начинается с клиентского запроса, в котором указывается приложение. ResourceManager выделяет необходимые ресурсы для контейнера и запускает ApplicationMaster для обслуживания указанного приложения. Используя протокол запроса ресурсов, ApplicationMaster выделяет контейнеры ресурсов для приложения на каждом узле. ApplicationMaster контролирует контейнер до завершения приложения. После завершения приложения ApplicationMaster отменяет выделение контейнера в ResourceManager, и цикл завершается

## Из каких 2-х основных частей состоит утилита Pig Latin? Каковы их функции?

Pig состоит из двух частей:

* язык для описания потоков Pig Latin;
* исполнительная среда для запуска сценариев Pig Latin (доступны два варианта: запуск на локальной JVM или исполнение в кластере Hadoop).

Apache Pig – это высокоуровневый процедурный язык, предназначенный для выполнения запросов к большим слабо структурированным наборам данных с помощью платформ Hadoop и MapReduce.

## Какой программный компонент является корнем иерархии модуля YARN?

Корнем иерархии YARN является ResourceManager.

Этот демон управляет всем кластером и назначением приложений базовым вычислительным ресурсам.

ResourceManager распределяет ресурсы (вычислительные ресурсы, память, пропускная способность и т.д.) для базовых NodeManager (агент YARN на узле). Кроме того, ResourceManager взаимодействует с ApplicationMaster при распределении ресурсов и с NodeManager при запуске и мониторинге базовых приложений.

## Какое предназначение утилиты Sqoop?

**Sqoop**

Apache Sqoop - это инструмент, предназначенный для передачи данных между Hadoop и реляционными базами данных или мэйнфреймами. Вы можете использовать Sqoop для импорта данных из реляционных систем управления базами данных (РСУБД), таких как MySQL или Oracle или мэйнфреймов в Hadoop, преобразовывать данные в Hadoop MapReduce, а затем экспортировать данные обратно в СУБД.

Sqoop автоматизирует большую часть этого процесса, опираясь на базу данных для описания схемы данных, которые будут импортированы. Sqoop использует MapReduce для импорта и экспорта данных, что обеспечивает параллельную работу, а также отказоустойчивость.

Sqoop поддерживает добавочную загрузку отдельной таблицы или [SQL](https://ru.bmstu.wiki/SQL_(Structured_Query_Language))-запроса произвольной формы, а также сохраненные задания, которые можно запускать несколько раз для импорта обновлений, сделанных в базу данных с момента последнего импорта. Импорт также можно использовать для заполнения таблиц в [[Apache Hive|Hive] или HBase. Экспорт можно использовать для помещения данных из Hadoop в реляционную базу данных.

Apache Sqoop выполняет следующие действия для интеграции перемещения больших объемов данных между Hadoop и структурированными хранилищами данных:

Импортировать последовательные наборы данных из мэйнфрейма. Удовлетворяет растущую потребность переносить данные с мэйнфреймов на HDFS;

Быстрые копии данных. Из внешних систем в Hadoop;

Эффективный анализ данных. Повышает эффективность анализа данных путем объединения структурированных данных с неструктурированными данными в озере данных схемы при чтении;

Балансировки нагрузки. Смягчает чрезмерную нагрузку при хранении и обработке на другие системы.

## Опишите основные функции файлового браузера HUE.

HUE (Hadoop User Experience) – это веб интерфейс для анализа данных на Hadoopе.

Hadoop Hue - это пользовательский интерфейс с открытым исходным кодом или пользовательский интерфейс для компонентов Hadoop. Пользователь может получить доступ к Hue прямо из браузера, и это повышает продуктивность разработчиков Hadoop. Он разработан Cloudera и является проектом с открытым исходным кодом. Через Hue пользователь может взаимодействовать с приложениями HDFS и MapReduce. Пользователям не нужно использовать интерфейс командной строки для использования экосистемы Hadoop, если он будет использовать Hue.

Особенности Hue

В Hue доступно множество функций, помимо веб-интерфейса, который он предоставляет разработчикам Hadoop. Hue предоставляет следующие перечисленные функции, благодаря которым он становится популярным инструментом для разработчиков Hadoop:

* Доступ к Hadoop API
* Наличие файлового браузера HDFS
* Браузер и дизайнер заданий
* Пользовательский интерфейс администратора
* Редактор Hive Query
* Редактор Pig Query
* Доступ к оболочке Hadoop
* Рабочие процессы могут получить доступ к интерфейсу Oozie
* Поиск по SOLR может получить отдельный интерфейс

Вышеупомянутые причины делают Hue предпочтительным выбором для разработчиков Hadoop и используются при установке кластера Hadoop. Доступ ко всем основным функциям Hadoop можно получить через Hue, а люди, не знакомые с интерфейсом командной строки, могут использовать Hue и получить доступ ко всем его функциям. На следующем изображении показан пользовательский интерфейс Hue:

## Опишите механизм распределенных запросов в Impala.

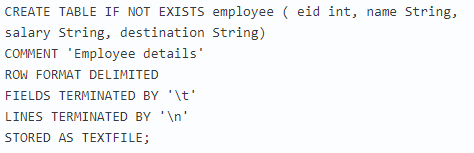
Запросы, выполняемые с помощью Impala, обрабатываются следующим образом:

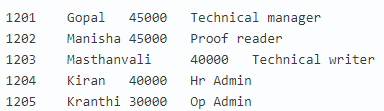
1. Пользовательские приложения отправляют SQL -запросы в Impala через ODBC или JDBC, которые предоставляют стандартизованные интерфейсы запросов. Пользовательское приложение может подключаться к любому impalad в кластере. Этот impalad становится координатором запроса.
2. Impala анализирует запрос, чтобы определить, какие задачи должны выполняться impalad экземплярами в кластере. Выполнение запланировано для оптимальной эффективности.
3. Доступ impalad к данным, таким как HDFS и HBase, осуществляется локальными экземплярами для предоставления данных.
4. Каждый impalad возвращает данные координирущей impalad, которая отправляет эти результаты клиенту.

## Таблицы каких 2-х видов используются в Hive? Опишите особенности их применения

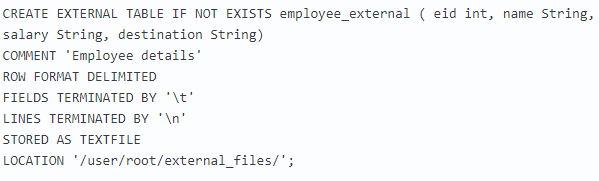
Таблица в hive представляет из себя аналог таблицы в классической реляционной БД. Основное отличие — что данные hive’овских таблиц хранятся просто в виде обычных файлов на hdfs. Это могут быть обычные текстовые csv-файлы, бинарные sequence-файлы, более сложные колоночные parquet-файлы и другие форматы. Но в любом случае данные, над которыми настроена hive-таблица очень легко прочитать и не из hive.ъ

Таблицы в hive бывают двух видов:

**Классическая таблица**, данные в которую добавляются при помощи hive. Вот пример создания такой таблицы ([источник примера](http://www.tutorialspoint.com/hive/hive_create_table.htm)):  


Тут мы создали таблицу, данные в которой будут храниться в виде обычных csv-файлов, колонки которой разделены символом табуляции.  После этого данные в таблицу можно загрузить. Пусть у нашего пользователя в домашней папке на hdfs есть (напоминаю, что загрузить файл можно при помощи hadoop fs -put) файл sample.txt вида:  


**Внешняя таблица**, данные в которую загружаются внешними системами, без участия hive. Для работы с  внешними таблицами при создании таблицы нужно указать ключевое слово EXTERNAL, а также указать путь до папки, по которому хранятся файлы:



После этого таблицей можно пользоваться точно так же как и обычными таблицами hive. Самое удобное в этом, что вы можете просто скопировать файл в нужную папочку в hdfs, а hive будет автоматом подхватывать новые файлы при запросах к соответствующей таблице. Это очень удобно при работе например с логами.

## Назовите основные Утилиты Hadoop. Каково их назначение?

**Hbase** — это распределенная, колоночноориентированная, мультиверсионная база типа «ключзначение». Данные организованы в таблицы, проиндексированные первичным ключом, который в Hbase называется RowKey. Для каждого RowKey ключа может храниться неограниченный набор атрибутов (или колонок).

Колонки организованны в группы колонок, называемые Column Family. Как правило в одну Column Family объединяют колонки, для которых одинаковый паттерн использования и хранения.

Для каждого атрибута может храниться несколько различных версий.Разные версии имеют разный timestamp.

Записи физически хранятся в отсортированном по RowKey порядке. При этом данные соответствующие разным Column Family хранятся отдельно, что позволяет при необходимости читать данные только из нужного семейства колонок.

**Hive** Cамая первая и до сих пор одна из самых популярных СУБД на этой платформе. В качестве языка запросов использует HiveQL — урезанный диалект SQL, который, тем не менее, позволяет выполнять довольно сложные запросы над данными, хранимыми в HDFS.

Hive представляет из себя движок, который превращает SQLзапросы в цепочки map-reduce задач. Движок включает в себя такие компоненты, как Parser (разбирает входящие SQL-запрсоы), Optimimer (оптимизирует запрос для достижения большей эффективности), Planner (планирует задачи на выполнение) Executor (запускает задачи на фреймворке MapReduce).

**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. 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.

**Apache Impala** - это открытая исходная база данных для Apache Hadoop . Impala поставляется компанией Cloudera.

Impala обеспечивает быстрые интерактивные SQLзапросы непосредственно на ваших данных Apache Hadoop, хранящихся в HDFS, HBase или Amazon Simple Storage Service (S3).

Помимо использования одной и той же унифицированной платформы хранения, Impala также использует те же метаданные, тот же синтаксис SQL (Hive SQL), драйвер ODBC и пользовательский интерфейс (интерфейс запроса Impala в Hue) как и Apache Hive. Это обеспечивает привычную и унифицированную платформу для запросов в реальном времени или пакетных запросов.

**Pig** Надстройка для Hadoop, позволяющая создавать программы для обрабоки данных на высокоуровневом языке

Pig состоит из двух частей: • язык для описания потоков Pig Latin; • исполнительная среда для запуска сценариев Pig Latin (доступны два варианта: запуск на локальной JVM или исполнение в кластере Hadoop)

Pig упрощает использование Hadoop, позволяя выполнять SQL-подобные запросы к распределенным наборам данных. Характерным свойством программ Pig является то, что их структура поддается существенному распараллеливанию,

**Mahout** Первая большая библиотека, реализовавшая многие популярные алгоритмы средствами MapReduce. Включает в себя алгоритмы для кластеризации, коллаборативной фильтрации, случайных деревьев, а также несколько примитивов для факторизации матриц.

**Rhadoop** Надстройка над Hadoop для выполнения программ, написанных на языке R. Как и Mahout позволяет использовать алгоритмы DataMining.

**Tez** представляет задачу в виде ориентированного ациклического графа (DAG) компонентовобработчиков. Планировщик запускает вычисление графа и при необходимости динамически переконфигурирует его, оптимизируя под данные.

**Oozie** Утилита позволяет создавать workflow из программ MapReduce, запросов Hive и Pig. Oozie — это проект с открытым исходным кодом на основе технологии Java™, упрощающий процесс создания потоков работ и координацию заданий. Oozie предоставляет принципиальную возможность объединения нескольких последовательно выполняемых заданий в одну логическую единицу работы.

**Hue** Web интерфейс к Hadoop для мониторинга заданий, выполняющихся на кластере, и облегчения обслуживания кластера. ϖHUE (Hadoop User Experience) – это веб интерфейс для анализа данных на Hadoop.

**Sqoop** Утилита для загрузки данных из реляционных баз данных в кластер Hadoop и для выгрузки данных обратно БД.

Инструмент для пересылки данных между структурированными хранилищами и HDFS

## Какие основные методы анализа данных применяются в технологиях Big Data?

**Методы и техники анализа больших данных**

Международная консалтинговая компания McKinsey, специализирующаяся на решении задач, связанных со стратегическим управлением, выделяет 11 методов и техник анализа, применимых к большим данным.

Методы класса Data Mining (добыча данных, интеллектуальный анализ данных, глубинный анализ данных) — совокупность методов обнаружения в данных ранее неизвестных, нетривиальных, практически полезных знаний, необходимых для принятия решений. К таким методам, в частности, относятся обучение ассоциативным правилам (association rule learning), классификация (разбиение на категории), кластерный анализ, регрессионный анализ, обнаружение и анализ отклонений и др.

Краудсорсинг — классификация и обогащение данных силами широкого, неопределённого круга лиц, выполняющих эту работу без вступления в трудовые отношения

Смешение и интеграция данных (data fusion and integration) — набор техник, позволяющих интегрировать разнородные данные из разнообразных источников с целью проведения глубинного анализа (например, цифровая обработка сигналов, обработка естественного языка, включая тональный анализ, и др.)

Машинное обучение, включая обучение с учителем и без учителя — использование моделей, построенных на базе статистического анализа или машинного обучения для получения комплексных прогнозов на основе базовых моделей

Искусственные нейронные сети, сетевой анализ, оптимизация, в том числе генетические алгоритмы (genetic algorithm — эвристические алгоритмы поиска, используемые для решения задач оптимизации и моделирования путём случайного подбора, комбинирования и вариации искомых параметров с использованием механизмов, аналогичных естественному отбору в природе)

Распознавание образов

Прогнозная аналитика

Имитационное моделирование (simulation) — метод, позволяющий строить модели, описывающие процессы так, как они проходили бы в действительности. Имитационное моделирование можно рассматривать как разновидность экспериментальных испытаний

Пространственный анализ (spatial analysis) — класс методов, использующих топологическую, геометрическую и географическую информацию, извлекаемую из данных

Статистический анализ — анализ временных рядов, A/B-тестирование (A/B testing, split testing — метод маркетингового исследования; при его использовании контрольная группа элементов сравнивается с набором тестовых групп, в которых один или несколько показателей были изменены, для того чтобы выяснить, какие из изменений улучшают целевой показатель)

Визуализация аналитических данных — представление информации в виде рисунков, диаграмм, с использованием интерактивных возможностей и анимации как для получения результатов, так и для использования в качестве исходных данных для дальнейшего анализа. Очень важный этап анализа больших данных, позволяющий представить самые важные результаты анализа в наиболее удобном для восприятия виде.7

## 54 Опишите процедуру создания базы данных в Hive.

База данных представляет аналог базы данных в реляционных СУБД. База данных представляет собой пространство имён, содержащее таблицы. Команда создания новой базы данных выглядит следующим образом: CREATE DATABASE|SCHEMA [IF NOT EXISTS]

Database и Schema в данном контексте это одно и тоже. Необязательная добавка IF NOT EXISTS создает базу данных только в том случае если она еще не существует.

Пример создания базы данных: CREATE DATABASE userdb;

Для переключения на соответствующую базу данных используем команду USE: USE userdb;

## Какой сервис используется в HBase для управления конфигурациями?

1. Master Server — главный сервер в кластере hbase. Master управляет распределением регионов по Region Servers, ведет реестр регионов, управляет запусками регулярных задач и делает другую полезную работу.

Для координации действий между сервисами Hbase использует Apache ZooKeeper, специальный сервис предназначенный для управления конфигурациями и синхронизацией сервисов.

ZooKeeper - это централизованная служба для хранения информации о конфигурации, присвоения имен, обеспечения распределенной синхронизации и предоставления групповых услуг. Все эти виды услуг в той или иной форме используются распределенными приложениями. Каждый раз, когда они внедряются, приходится много работать над исправлением ошибок и условий гонки, которые неизбежны.

При увеличении количества данных в регионе и достижении им определенного размера Hbase запускает split, операцию разбивающую регион на 2. Для того чтобы избежать постоянных делений регионов — можно заранее задать границы регионов и увеличить их максимальный размер.

## Для чего в HBase используется BlockCache?

BlockCache — кэш на чтение. Позволяет существенно экономить время на данных которые читаются часто.

## Какая операция применяется в HBase для управления размером регионов?

При увеличении количества данных в регионе и достижении им определенного размера Hbase запускает split, операцию, разбивающую регион на 2. Для того чтобы избежать постоянных делений регионов — можно заранее задать границы регионов и увеличить их максимальный размер.

Так как данные по одному региону могут храниться в нескольких HFile, для ускорения работы Hbase периодически их сливает воедино. Эта операция в Hbase называется compaction. Compaction

Minor Compaction. Запускается автоматически, выполняется в фоновом режиме. Имеет низкий приоритет по сравнению с другими операциями Hbase.

Major Compaction. Запускается руками или по наступлению срабатыванию определенных триггеров (например по таймеру). Имеет высокий приоритет и может существенно замедлить работу кластера.

## В каком формате хранятся данные в хранилище HBase Persistent Storage?

Persistent Storage — основное хранилище данных в Hbase. Данные физически хранятся на HDFS, в специальном формате HFile. Данные в HFile хранятся в отсортированном по RowKey порядке. Одной паре (регион, column family) соответствует как минимум один HFIle.

MemStore — буфер на запись. Так как данные хранятся в HFile в отсортированном порядке — обновлять HFile на каждую запись довольно дорого. Вместо этого данные при записи попадают в специальную область памяти MemStore, где накапливаются некоторое время. При наполнении

MemStore до некоторого критического значения данные записываются в новый HFile.

## Какие БД (строковые или колоночные) более эффективны для решения задач BigData?

**Колоночные**

Используются для

Событийные таблицы, над которыми выполняются сложные выборки (агрегации, фильтры, сортировки). Например, добавление товаров в корзину в Интернет-магазине. Метрики рекламы, нагрузок.

Агрегатные таблицы с большим количеством данных для аналитических выборок. Это часто таблицы, которые строятся из событийных таблиц. Это могут быть таблицы со статистикой добавления товаров в корзину по дням, категориям, ценам и другим параметрам

+

* Колоночные базы данных позволяют эффективно делать сложные выборки на больших таблицах.
* Изменение структуры больших таблиц происходит мгновенно.
* Сжатие данных позволяет сэкономить место

-

* Медленно работают на запись
* Не поддерживают транзакции
* Имеют ряд ограничений для разработчика (sql синтаксис не полный)
* Обычные выборки по ключу всех столбцов без профита

## Для чего необходим лог-файл WAL в HBase?

Write Ahead Log(WAL). Так как данные при записи попадают в memstore, существует некоторый риск потери данных из-за сбоя. Для того чтобы этого не произошло все операции перед собственно осуществление манипуляций попадают в специальный лог-файл. Это позволяет восстановить данные после любого сбоя.