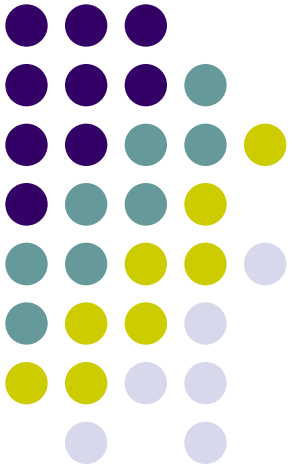
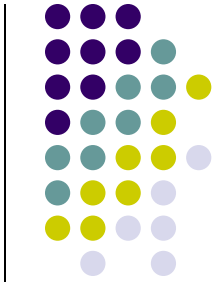
Введение в распределенные методы обработки информации

Лекция №7

Большие данные +Hadoop



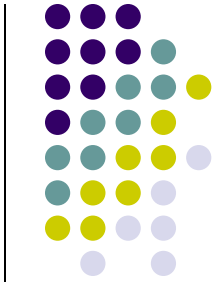
Предыстория



Первое упоминание – научная публикация

- Дата рождения термина — 3 сентября 2008 года, когда вышел специальный номер старейшего британского научного журнала Nature, посвященный поиску ответа на вопрос «Как могут повлиять на будущее науки технологии, открывающие возможности работы с большими объемами данных?».
- Специальный номер подытоживает предшествующие дискуссии о роли данных в науке вообще и в электронной науке (e-science) в частности.

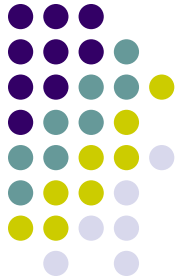
Предыстория



Далее термин стали использовать бизнес-издания
Большие данные сравнивали с

- минеральными ресурсами —
  - the new oil (новая нефть),
  - goldrush (золотая лихорадка),
  - data mining (разработка данных), чем подчеркивается роль данных как источника скрытой информации;
- с природными катаклизмами —
  - data tornado (ураган данных),
  - data deluge (наводнение данных),
  - data tidal wave (половодье данных), видя в них угрозу;
- с промышленным производством —
  - data exhaust (выброс данных),
  - firehose (шланг данных),
  - Industrial Revolution (промышленная революция).

Существует ли проблема Больших Данных ?



- Большие Данные - red herring (букв. «копченая селедка» — ложный след, отвлекающий маневр
- Большие Данные - прежде всего маркетинговый ход разработчиков, продвигающих свою продукцию

Возможно, Большие Данные есть что-то качественно иное, чем то, к чему подталкивает обыденное сознание.

Big Data — многозначный термин

Под термином Большие Данные в разном контексте могут подразумеваться:

- ***данные большого объема***
- ***технология их обработки***,
- ***проекты или рынок компаний***, активно использующих эту технологию

Обсудим правомерность каждого из этих определений.

Данные большого объема

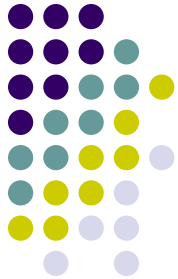
В основе информационного взрыва лежит цифровизация нашей жизни.

Данные накапливаются эксабайтами (10^{18} байт).

Данные становятся все детальнее и персонифицированнее и собираются у все большего числа игроков.

Наблюдается положительная обратная связь в накоплении научных данных: данные измерений после обработки порождают новые данные, за счет чего процесс накопления научных данных постоянно ускоряется

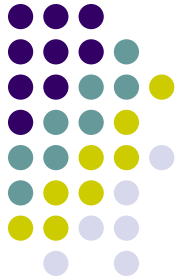
Источники Больших Данных



Научные исследования

- *Ядерная физика*
  - в большом адронном коллайдере в ЦЕРНе соударения частиц происходят с частотой 20 млн в секунду. За день получается количество данных, сопоставимое с объемом всего интернета
  - данные с датчиков наблюдений за ядерными реакторами
- *Астрономические наблюдения*
  - один архив телескопа «Хаббл», накопленный за 15 лет, занимает около 25 Тбайт
- *Метеорологические наблюдения*
  - японская компания Weathernews, поставляющая датчики температуры, влажности и давления для смартфонов WxBeacon и одноименное приложение, собирает информацию от 13 млн человек; погодные сенсоры, по всей видимости, вскоре появятся в наших мобильных девайсах, соответственно свой вклад в метеопрогнозы сможет внести каждый

Источники Больших Данных

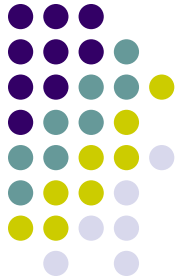


Интернет

По состоянию на 2012 год:

- количество email, отправляемых каждую секунду в мире, — 2,9 млн.
- объем видео, закачиваемого на YouTube каждую минуту, — 20 часов.
- объем данных, обрабатываемых Google за день, — 24 петабайт.
- количество сообщений на Твиттере в день — 50 млн.
- на Facebook ежечасно загружается более 10 млн фотографий.
- объем данных, переданных/полученных на мобильные устройства, — 1,3 эксабайт.
- количество продуктов, заказываемых в Amazon в секунду, — 72,9.

Источники Больших Данных

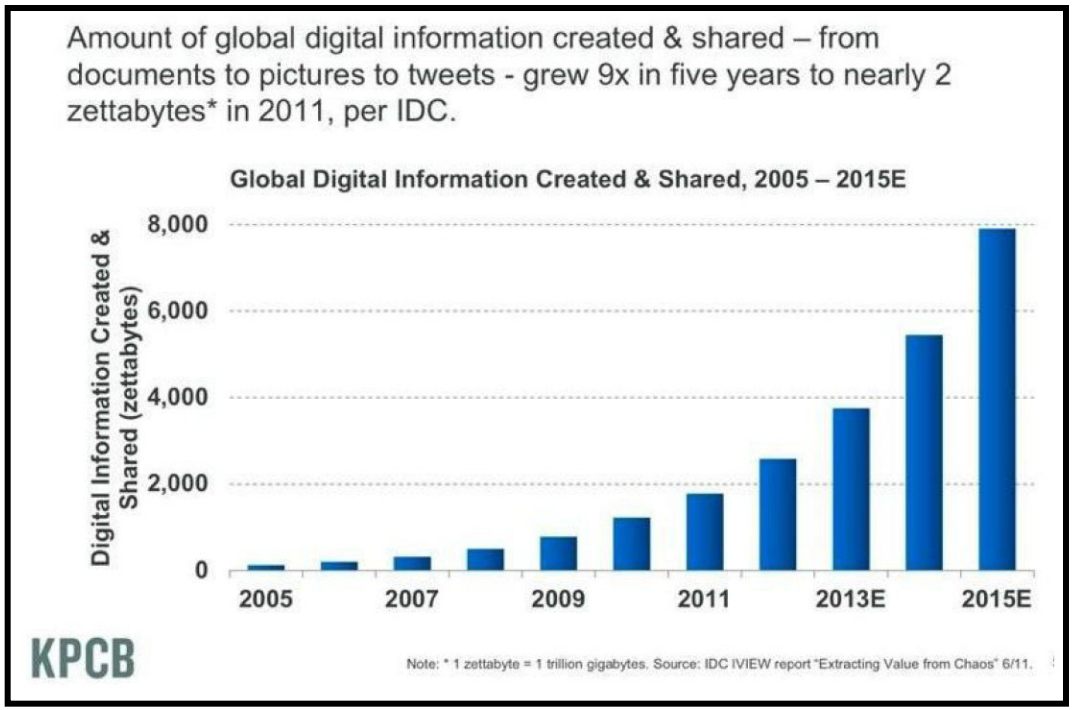


Мобильные устройства:

Ожидается, что к 2020 году

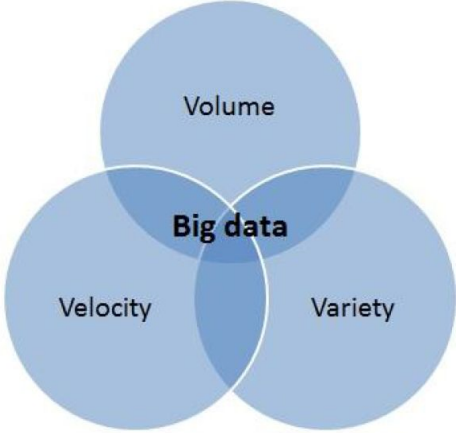
- количество смартфонов увеличится с сегодняшних 2,7 млрд до 6,1 млрд,
- общий объем мобильного трафика, генерируемого смартфонами будет в пять раз превышать сегодняшний.
- совокупные ежегодные темпы роста рынка информационной аналитики, основанной на анализе Больших Данных увеличиться почти на 50 процентов - а годовой доход, как ожидается, достигнет 5,4 млрд долларов США

Рост объемов цифровых данных



- На графике изображен рост объемов информации, согласно исследованию IDC — в 2011 году это значение приблизилось к двум триллионам гигабайт, к 2020 году общий объем цифровых данных достигнет 40 зеттабайт ($40*10^{21}$ байт) . Если записать 40 зеттабайт на самые емкие современные диски Blue-ray, суммарный вес дисков без упаковки будет равен весу 424 авианосцев.

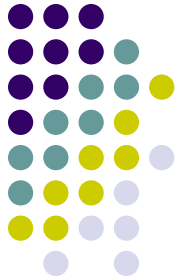
Определяющие характеристики



Аналитики Gartner в своих статьях описали три основных характеристики «Больших Данных», обозначаемых как «три V»:

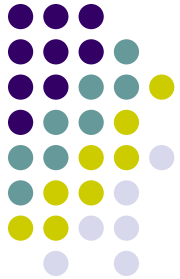
- *объём* (англ. **v**olume) – физический объем хранимых данных;
- *скорость* (англ. **v**elocity) – скорость изменчивости данных и, как следствие, последующий анализ этих изменений;
- *многообразие* (англ. **v**ariety) – разнообразие обрабатываемых типов данных: как структурированные, так и неструктурированные данные.

Большие данные = технологии их обработки



- Данных действительно становится все больше и больше, но при этом способность порождать данные оказалась сильнее, чем способность их перерабатывать.
- Кроме задачи сохранения данных, с которыми современные аппаратно-программные средства справляются, в общем-то, успешно, стоят задачи и аналитической обработки этих данных в режиме максимально приближенном к режиму реального времени.
- Принципиально иных, чем привычные, разработанные еще 10-15 лет назад, подходов к обработке данных требуют и задачи прогнозной аналитики, имитационного моделирования, статистического анализа, машинного обучения

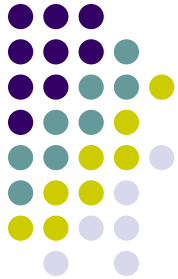
Определение больших данных как технологии



Большие данные – это:

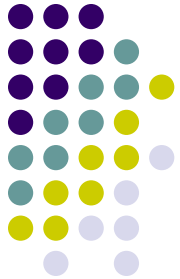
- серия подходов, инструментов и методов
  - *обработки* структурированных и неструктурированных данных огромных объёмов и значительного многообразия
    - *для получения* воспринимаемых человеком результатов,
      - эффективных в условиях *непрерывного прироста* и *распределения* по многочисленным узлам вычислительной сети,
        - *альтернативных традиционным системам* управления базами данных.

Большие данные = *проекты или рынок компаний*, **активно использующих эту технологию**



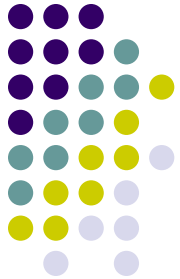
- При принятии взвешенного решения о выдаче кредита за пятнадцать минут нужно проанализировать серьезный массив данных.
- У интернет-магазинов время измеряется уже секундами. Пока клиент задумался, нужен ли ему тот или иной товар, Amazon должен успеть проанализировать историю его поведения в интернете, сравнить с поведением аналогичных клиентов и подсунуть наиболее заманчивые альтернативы из ассортимента более чем в миллион позиций.
- Компания ComScore, крупнейший поставщик профилей интернет-пользователей, обрабатывает в день 30 млрд событий — просмотров страниц, длительности сеансов, демографических сведений.
- У поисковых систем времени на работу — доли секунды, в течение которых они должны не просто найти подходящие варианты ответов на запрос, но и сопоставить их с актуальностью профиля конкретного юзера.

Большие данные = *проекты или рынок компаний*, **активно использующих эту технологию**



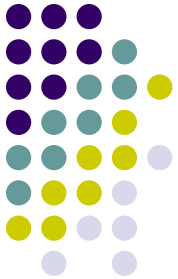
- В 2010 году появляются первые продукты и решения, относящихся исключительно и непосредственно к проблеме обработки Больших Данных.
- К 2011 году большинство крупнейших поставщиков информационных технологий для организаций в своих деловых стратегиях используют понятие о Больших Данных, в том числе IBM, Oracle, Microsoft, Hewlett-Packard, EMC.

Большие данные = *проекты или рынок компаний*, **активно использующих эту технологию**



- Путем объединения и анализа больших объемов данных можно раскрыть коммерческий потенциал мегамассивов данных за счет поиска ценных закономерностей и фактов.
- Традиционные подходы к их хранению и обработке стали неэффективными, а следовательно, необходимы новые технологии.
- Перед бизнесом встала задача не только выбора адекватного инструментария по анализу информации, но и построения оптимальной вычислительной инфраструктуры, которая была бы эффективной и не очень дорогой.

Большие данные – обобщающее определение



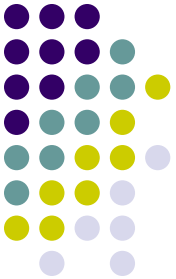
Большие данные – это наборы данных такого *объема*, что

традиционные инструменты *не способны осуществлять*

- их захват,
- управление и
- обработку

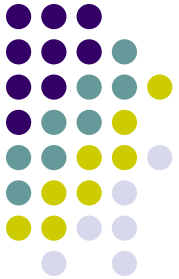
за приемлемое для практики время.

Реальная проблема Больших Данных



- Технические возможности работы с данными явно опередили уровень развития способностей к их использованию
- Целью обработки данных является получении новой информации и новых знаний из уже существующих массивов данных
- Данные обрабатываются для получения информации, которой должно быть ровно столько, чтобы человек мог превратить ее в знание.

Классификация Больших Данных

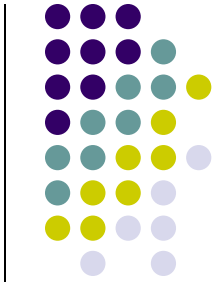


Дайон Хинчклиф, редактора журнала Web 2.0 Journal делит Большие данные на 3 группы:

- Быстрые Данные (Fast Data), их объем измеряется терабайтами;
- Большая Аналитика (Big Analytics) — петабайтные данные
- Глубокое Проникновение (Deep Insight) — экзабайты, зеттабайты.

Группы различаются между собой не только оперируемыми объемами данных, но и качеством решения по их обработки.

Fast Data



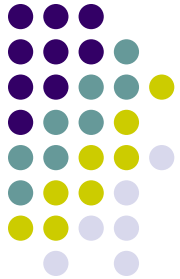
Обработка данных для Fast Data:

- не предполагает получения новых знаний,
- ее результаты соотносятся с априорными знаниями и позволяют судить о том, как протекают те или иные процессы,
- позволяет лучше и детальнее увидеть происходящее, подтвердить или отвергнуть какие-то гипотезы.

Только небольшая часть из существующих сейчас технологий подходит для решения задач Fast Data, в этот список попадают некоторые технологии работы с хранилищами (продукты Greenplum, Netezza, Oracle Exadata, Teradata, СУБД типа Verica и kdb).

Скорость работы этих технологий должна возрастать синхронно с ростом объемов данных.

Технологии Big Analytics



- должны помогать в получении новых знаний
- они служат для преобразования зафиксированной в данных информации в новое знание
- не предполагают наличие искусственного интеллекта при выборе решений или каких-либо автономных действий аналитической системы
- строятся по принципу «обучения с учителем» - весь аналитический потенциал закладывается в процессе обучения.

Самый очевидный пример — машина Watson , играющая в Jeopardy!.

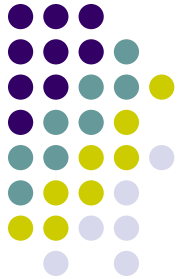
Классическими представителями такой аналитики являются продукты MATLAB, SAS, Revolution R, Apache Hive, SciPy Apache и Mahout.

Deep Insight

- предполагает обучение без учителя (unsupervised learning)
- использование современных методов аналитики, а также
- использование различных способов визуализации.

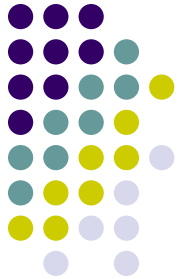
На этом уровне возможно обнаружение знаний и закономерностей, априорно неизвестных.

Большие данные - перспективы



- Большие данные - качественный переход в компьютерных технологиях, способный повлечь за собой серьезные изменения.
- Большие данные — очередная техническая революция со всеми вытекающими последствиями.

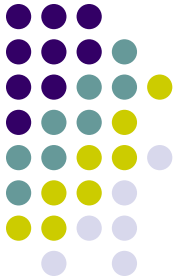
Большие данные – примеры использования



Парковки:

- технологии обработки больших данных помогают менеджерам
- оптимизировать использование наемного/временного персонала – предсказывать пики занятости парковок и «спокойные» периоды, на основе чего оптимизируется выход персонала на работу и заказ спецтехники.
- клиенты получают информацию о наличии свободных мест онлайн, через сайт или приложения
- предиктивная система может показывать людям, например, отсутствие свободных мест даже тогда, когда они еще есть – чтобы они заранее искали другие места для парковки, потому что алгоритм предсказывает, что свободные места будут заняты в течение нескольких минут.

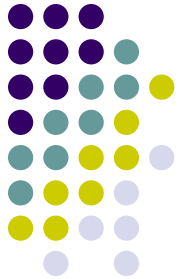
Большие данные – примеры использования



ГОРНЫЕ ЛЫЖИ

- *борьба с фродом* – недополученная выручка горнолыжных курортов может составлять до 8% из-за фрода
  - внедрение RFID (Radio Frequency Identification) в билеты, ски-пассы и карты клиентов позволило видеть в онлайне загруженность различных склонов и заведений, а фрод сведен почти к нулю
- *управление загруженностью склонов*
  - поступающие данные о погоде позволяют менеджерам предупреждать клиентов о проблемах и управлять машинами искусственного снега и ратраками для обеспечения минимального времени простоя и повышения удовлетворенности клиентов
- *производство горных лыж*
  - с помощью технологии Больших Данных удалось на 48% улучшить предсказание спроса с помощью big data, и на 30% снизить простои производственных линий, Производственный цикл сократился до двух недель

Большие данные – примеры использования

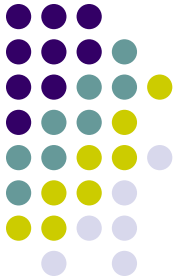


НАРУЖНАЯ РЕКЛАМА

агентства смогут планировать размещение рекламы на основании того,

- как передвигается в течении дня нужная им аудитория,
- куда едет или идет,
- где стоит на светофорах,
- куда скорее всего смотрит в этот момент

Большие данные – примеры использования

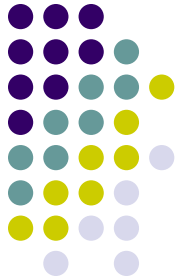


УПРАВЛЕНИЕ ДОРОЖНЫМ ДВИЖЕНИЕМ

Используя относительно недорогие сенсоры радиоволн различных типов, исходящих из пользовательских устройств, администрации городов

- собирают данные о потоках людей и машин в различных частях города
- отслеживают количество машин на улицах, как быстро они перемещаются и какие дороги наиболее загружены
- управляют светофорами для минимизации пробок
- прокладывают альтернативных маршрутов при перекрытии дорог и планировании путей привоза-отвоза публики на общественном транспорте при планировании крупных мероприятий

Большие данные – примеры использования

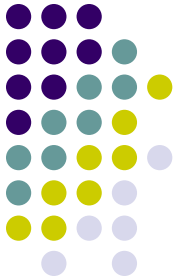


НАЙМ ПЕРСОНАЛА

Изучая миллионы записей о персонале в различных компаниях, можно определить общие паттерны поведения в зависимости от различных факторов, например:

- сотрудники, имеющие уголовное прошлое, работают лучше тех, кто был «чист» перед законом
- повторно нанятые сотрудники покидают компанию на 44% быстрее, чем те, кто пришел в компанию впервые

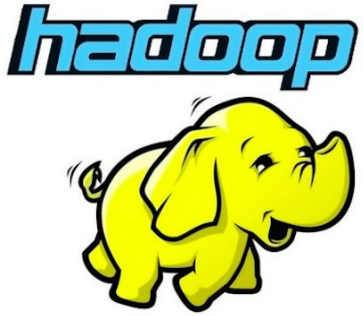
Большие данные – примеры использования



АЗАРТНЫЕ ИГРЫ

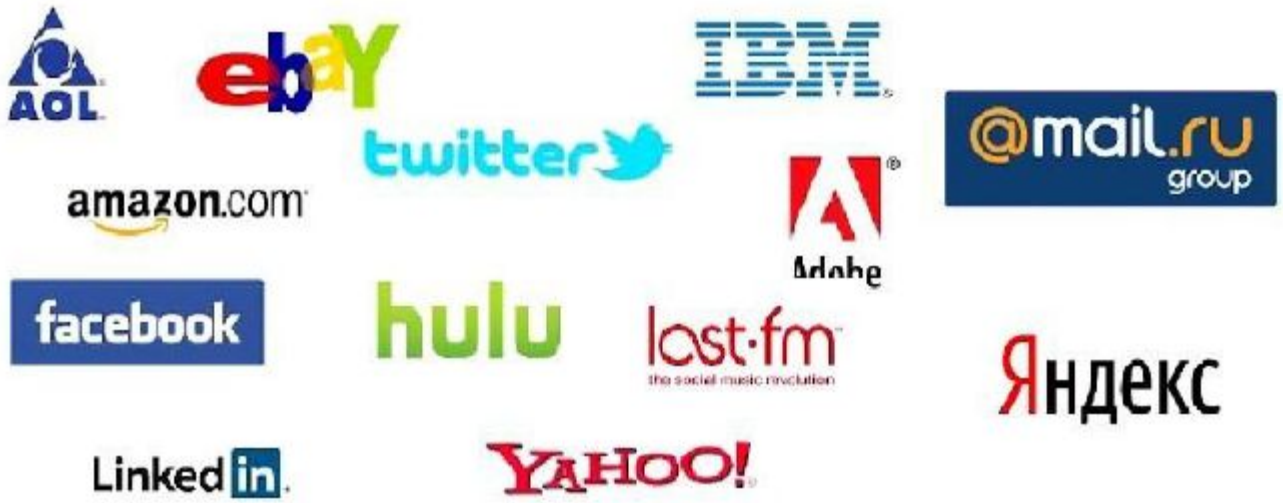
- Онлайн-казино и сайты с разными типами азартных игр производят массу данных о своих клиентах – их привычках в играх, любимых играх, времени, которое они проводят в конкретной игре, сколько они тратят на какие игры, и так далее.
- Эти данные используются для того, чтобы анализировать то, как можно увеличить время, которое разные люди играют в различные игры, а раз они проводят там больше времени, то соответственно и больше тратят.
- Онлайн-казино проводят бесчисленное количество A/B тестов в своих игровых интерфейсах, чтобы выяснить, какой вид оформления и игровых механик благоприятно влияет на время, которое игроки проводят у них.
- Помимо данных об их игровых привычках, также собирается масса информации о том, как/откуда они приходят в онлайн-казино, и на какие сайты идут после них, и эта информация используется в маркетинге других товаров и услуг.

Платформа Hadoop

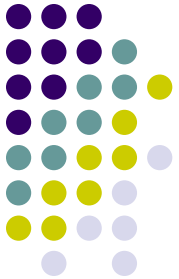


- **Hadoop** – это это *свободно распространяемый* набор программных средств (Software Framework) для разработки и выполнения *распределённых приложений*, предназначенных для *массивно-параллельной обработки* (*Massive Parallel Processing*, MPP) данных.
- Hadoop наиболее эффективен при работе чрезвычайно *большими* объемами данных, но фактически система может применяться и при обработке массивов
- Термин big data появился несколько *позже* развития концепции платформы Hadoop (2008 vs 2004-2006)

Пользователи Hadoop

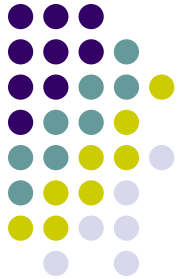


Компоненты платформы Hadoop



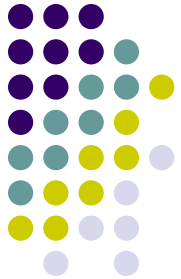
- *Hadoop Distributed File System (HDFS)* : распределенная файловая система, которая обеспечивает высокоскоростной доступ к данным приложения;
- *Hadoop MapReduce*: программная платформа для распределенной обработки больших объемов данных на вычислительном кластере
- *Hadoop Common:* библиотеки и сценарии управления распределенной обработкой, файловой системой, развертывания инфраструктуры;
- *Hadoop YARN:* система планирования заданий и управления ресурсами кластеров

Основные технические характеристики платформы Hadoop



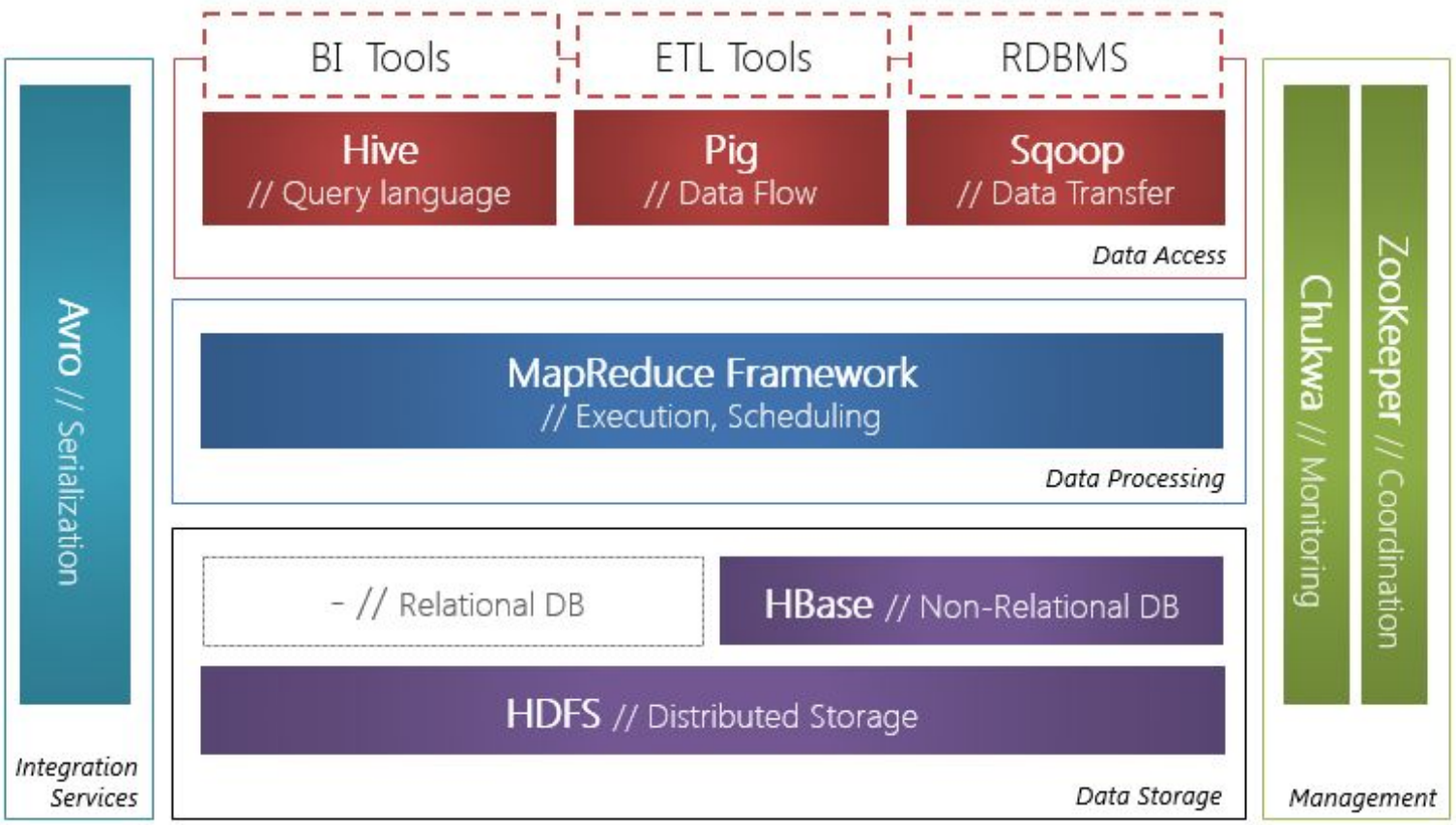
- ***Масштабируемость***: платформа масштабируется линейно и позволяет хранить и обрабатывать петабайты данных;
- ***Устойчивость к сбоям***: все хранящиеся данные избыточны, все проваленные задания по обработке данных перезапускаются;
- ***Кроссплатформенность***: библиотеки Hadoop написаны (в основном) на Java, и могут выполняться в любой операционной системе, поддерживающей JVM (Java VM);
- ***Автоматическое распараллеливание выполнения задачи***: Hadoop создает «чистые» абстракции для разработчиков, снимая с них работу по планированию, контролю и агрегатированию результатов параллельной обработки данных.

Бизнес-выгоды от использования Hadoop



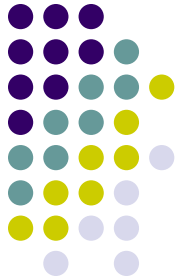
- ***Гибкость***: хранение и анализ структурированных и неструктурированных типов данных;
- ***Эффективность***: в большинстве случаев более низкая стоимость хранения / обработки терабайта данных по сравнению существующими решениями;
- ***Низкая стоимость*** *создания кластера*: для создания Hadoop-кластера не требуется дорогое серверное аппаратное обеспечение.
- ***Сравнительная легкость адаптации***: Hadoop имеет широкую и активно развивающуюся экосистему;
- ***Минимальные риски***, связанные с некорректной работой ядра платформы: на сегодняшний день платформа Hadoop успешно используется для обработки петабайт информации;
- *«**Open Source**» лицензирование*: низкая стоимость внедрения и владения платформой Hadoop, большое «developer community».

Экосистема платформы Hadoop



Reference: Tom White. Hadoop: The Definitive Guide

Экосистема платформы Hadoop



- Центральное место экосистемы Hadoop занимает *хранилище данных* (Data Storage). Hadoop поддерживает хранение как *неструктурированных* данных с распределенной файловой системе HDFS, так и *структурированных* данных в нереляционной базе данных HBase.
- Фреймворк *MapReduce* отвечает за *планирование задач* (Job Scheduling) и выполнение *распределенных вычислений*.

Экосистема платформы Hadoop (Hive)

Hive – это надстройка над Hadoop для выполнение следующих задач:

- суммирование данных,
- непрограммируемые запросы
- анализ больших наборов данных

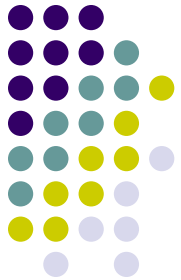
Hive обеспечивает SQL-подобный язык, называемый HiveQL, сохраняя полную поддержку Map/Reduce.

Hive создает задания MapReduce, которые исполняются на кластере Hadoop.

Hive поддерживает анализ больших массивов данных, хранящихся в Hadoop в HDFS и совместим с другими файловыми системами такими, как Amazon S3.

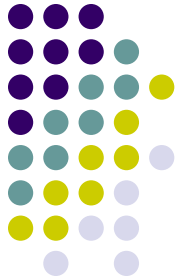
Определения таблиц в Hive надстраиваются над данными в HDFS.

Экосистема платформы Hadoop (Pig)



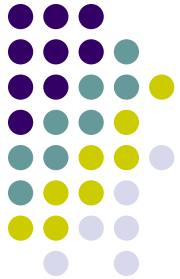
- Pig – это надстройка, предназначенная для анализа больших наборов данных и состоящая из языка высокого уровня (PigLatin) для написания программ анализа данных и инфраструктуры для запуска этих программ.
- Язык характеризуется относительно простым синтаксисом. Написанные сценарии скрыто преобразуются в задачи MapReduce, которые исполняются на кластере Hadoop.
- Pig и Hive обеспечивают практически одинаковый функционал работы с данными, хотя Hive работает быстрее. Оба модуля дают разные инструменты реализации, поэтому выбор зависит исключительно от разработчика приложений.

Экосистема платформы Hadoop (Sqoop)



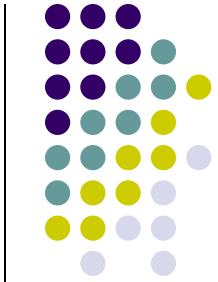
- Sqoop – это инструмент, предназначенный для эффективной передачи больших массивов данных между Hadoop и структурированными СУБД (например, реляционными) в обоих направлениях.
- Sqoop стал ключевым решением для трансфера данных между SQL и Hadoop.
- Проект предоставляет коннекторы для популярных систем MySQL, PostgreSQL, Oracle, SQL Server и DB2.
- Есть возможность разрабатывать высокоскоростные коннекторы для специализированных систем, таких как корпоративные хранилища данных.

Spark In memory database



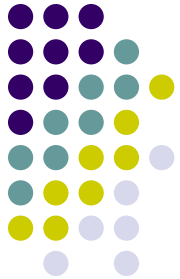
- Spark In memory database – специализированная распределенная система для ускорения обработки данных в памяти.
- Интегрирован с Hadoop. По сравнению с предоставляемым в Hadoop механизмом MapReduce, Spark обеспечивает в 100 раз более высокую производительность при обработке данных в памяти и в 10 раз при размещении данных на дисках.
- Движок может выполняться на узлах кластера Hadoop как при помощи Hadoop YARN, так и в обособленном режиме.

Spark In memory database



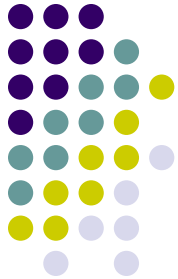
- Поддерживается обработка данных в хранилищах HDFS, HBase, Cassandra, Hive и любом формате ввода Hadoop (InputFormat).
- В отличие от MapReduce Spark не сохраняет промежуточные наборы результатов на диск (если они не слишком большие, чтобы вписаться в RAM).
- Spark формирует RDDs (Resilient Distributed Datasets), который может находится и обрабатываются в оперативной памяти полностью либо частично. RDDs не имеет жесткого формата.
- Система позиционируется как быстрый инструмент для работы с данными хранящимися в кластере Hadoop.

Shark



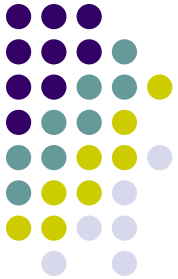
- Shark – компонент Spark, обеспечивающий распределенный механизм SQL-запросов для данных Hadoop.
- Shark полностью совместим с клиентом Hive.
- Shark является компонентом Spark.
- Это высокопроизводительный продукт с открытым исходным кодом,обеспечивающий распределенный механизм SQL-запросов для данных Hadoop.
- Благодоря Shark обеспечивается высокая производительность и усовершенствованная аналитика пользователям Hive.

Файловая система HDFS

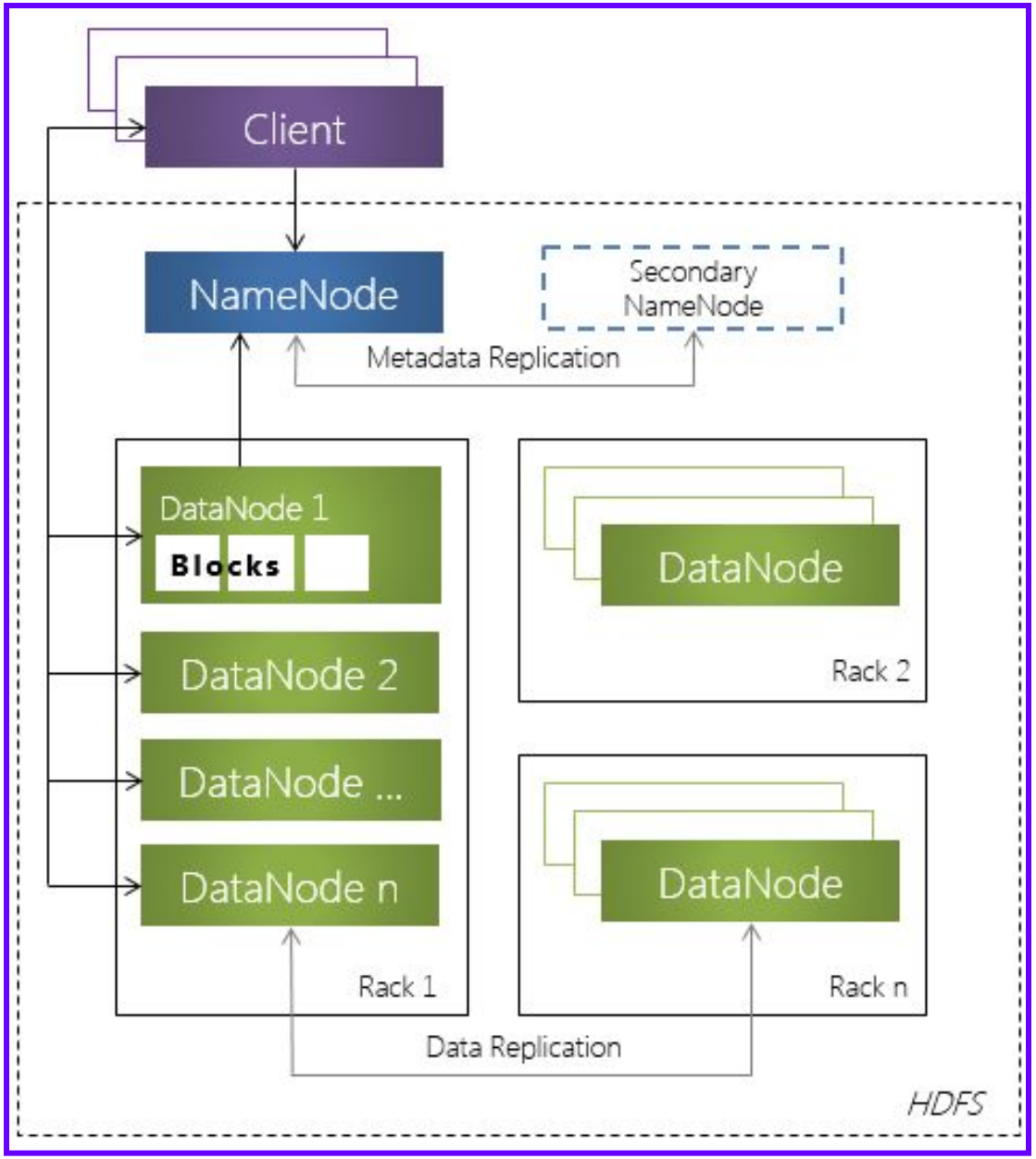


- **Hadoop Distributed File System** (**HDFS**) - распределенная файловая система, которая обеспечивает высокоскоростной доступ к данным приложения.
- HDFS является иерархической файловой системой. Таким образом, в HDFS имеется поддержка вложение каталогов. В каталоге может располагаться ноль или более файлов, а также любое количество подкаталогов.

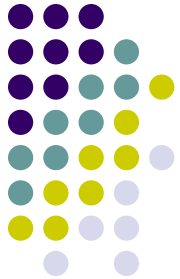
Обязательные компоненты HDFS



- **Узел имен** (**NameNode**) – программный код, выполняющийся, в общем случае, на выделенной машине экземпляра HDFS и отвечающий за файловые операции (работу *с метаданными*);
- **Узел данных** (**DataNode**) – программный код, как правило, выполняющийся выделенной машине экземпляра HDFS и отвечающий за операции уровня файла (работа *с блоками данных*).
- Hadoop содержит единственный узел типа NameNode и произвольное количество узлов типа DataNode.


Client
NameNode
Secondary NameNode
Metadata Replication
DataNode 1
Blocks
DataNode 2
DataNode ...
DataNode n
Rack 1
DataNode
Rack 2
DataNode
Rack n
Data Replication
HDFS

Основные концепции и архитектурные решения



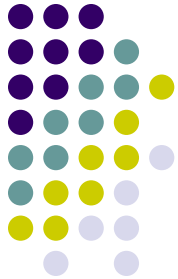
Объем данных

- HDFS не должна иметь достижимых в обозримом будущем ограничений на объем хранимых данных.

Архитектурные решения:

- HDFS хранит файлы поблочно.
- Блоки в HDFS распределены между узлами данных вычислительного кластера.
- Все блоки (кроме последнего блока файла) имеют одинаковый размер, кроме того блок может быть размещён на нескольких узлах.

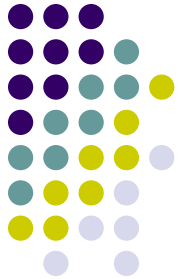
Основные концепции и архитектурные решения



Отказоустойчивость

- HDFS расценивает выход из строя узла данных как норму, а не как исключение (и, действительно, вероятность выхода хотя бы одного узла из тысячи даже на надежном физическом оборудовании существенная).
- Архитектурные решения:
  - Для обеспечения отказоустойчивости все данные в HDFS реплицируются настраиваемое количество раз.
  - Защита от копирования поврежденных данных решена с помощью хранения контрольных сумм в отдельном скрытом файле.
  - Копирование метаданных с помощью вторичного узла имен.

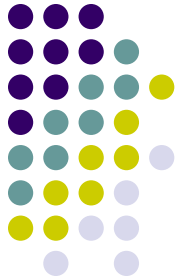
Основные концепции и архитектурные решения



Самодиагностика

- Диагностика исправности узлов в Hadoop-кластере не должна требовать дополнительного администрирования.
- Архитектурные решения:
  - Каждый узел данных через определенные интервалы времени отправляет диагностические сообщения узлу имен.
  - Логгирование операций над файлами в специальный журнал узла имен.

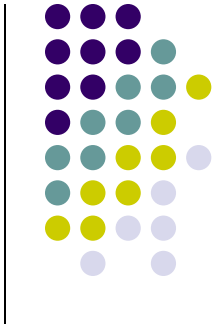
Основные концепции и архитектурные решения



Производительность

- В апреле 2008 года Hadoop побил мировой рекорд производительности в стандартизованном тесте производительности по сортировке данных — 1 ТБайт был обработан за 309 сек. на кластере из 910 узлов.
- Архитектурные решения:
  - Принцип «один раз записать – много раз прочитать» (Write-once and read-many, WORM) полностью освобождает систему от блокировок типа «запись-чтение». Избавиться от конфликтов множественной записи проектировщики решили, разрешив запись в файл в одно время только одному процессу.
  - HDFS оптимизирован под потоковую передачу данных.
  - Снизить нагрузку на каналы передачи данных (а именно эти каналы чаще всего являются узким местом в распределенных средах), а также более рационально использовать место на жестких дисках позволило сжатие данных.
  - Репликация происходит в асинхронном режиме.
  - Хранение всех метаданных узла NameNode в оперативной памяти.

Узлы имен



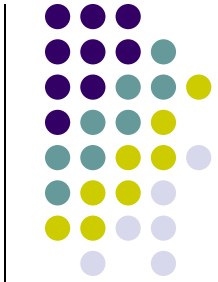
Узел имен (**NameNode**) представляет собой программный код, выполняющийся, в общем случае, на выделенной машине экземпляра HDFS..

NameNode отвечает за:

- за файловые операции, такие как открытие и закрытие файлов, создание и удаление каталогов
- управление пространством имен файловой системы;
- управление доступом со стороны внешних клиентов;
- соответствие между файлами и реплицированными на узлах данных блоками.

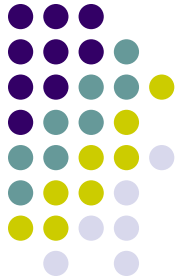
Hadoop содержит единственный узел типа NameNode. Что порождает уязвимость всего кластера, вызванную выходом узел типа NameNode (единичная точка отказа).

Узлы имен



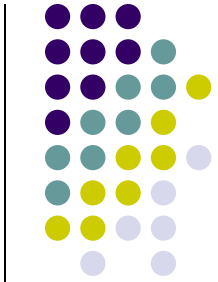
- HDFS поддерживает вторичный узел имен – Secondary NameNode. Часто это факт является причиной заблуждения, что при отказе первичного узла имен, его автоматически заменит вторичный узел имен.
- Вторичный узел имен выполняет следующие функции:
- копирует образ HDFS (расположенный в файле FsImage) и лог транзакций операций с файловыми блоками (EditLog) во временную папку;
- применяет изменения, накопленные в логе транзакций к образу HDFS;
- записывает новый образ FsImage на узел NameNode, после чего происходит очистка EditLog.

Узлы данных



- Узел данных (DataNode), как и узел NameNode, также представляет собой программный код, выполняющийся, как правило, на выделенной машине экземпляра HDFS и отвечающий за операции уровня файла, такие как:
  - запись и чтение данных,
  - выполнение команд создания, удаления и репликации блоков, полученные от узла NameNode.
- Кроме того, узел DataNode отвечает за:
  - периодическую отправку сообщения о состоянии (heartbeat-сообщения);
  - обработку запросов на чтение и запись, поступающие от клиентов файловой системы HDFS, т.к. данные проходят с остальных машин кластера к клиенту мимо узла NameNode.

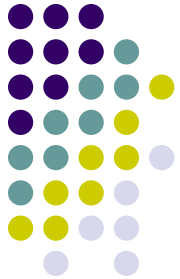
Клиенты HDFS



Клиенты представляют собой программных клиентов, работающих с файловой системой.

В роли клиента может выступать любое приложение или пользователь, взаимодействующий через специальный API с файловой системой HDFS.

Клиенты HDFS

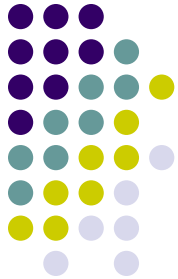


Для клиента HDFS выглядит как обычная файловая система – иерархия каталогов с вложенными в них подкаталогами и файлами.

Как и в файловых системах общего назначения, клиенту, при наличии достаточных прав, разрешены следующие операции:

- создание,
- удаление,
- переименование, перемещение.

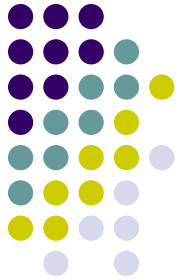
Клиенты HDFS



Наиболее существенное отличие работы клиента с файловой системой HDFS от работы с файловой системой общего назначение – это то, что при создании файла

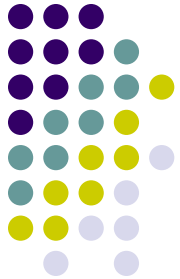
- клиент может явно указать размер блока файла (по умолчанию 64 Мб) и
- количество создаваемых реплик (по умолчанию значение равно 3-ем).

Взаимодействие компонентов HDFS



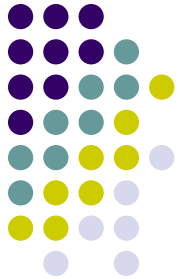
- Взаимодействие узлов имен, узлов данных и клиентов осуществляется по протоколам, основывающимся на протоколе TCP/IP.
- Клиент создает соединение через специально сконфигурированный для взаимодействия TCP-порт на целевом узле NameNode. Взаимодействия клиента с узлом NameNode происходит по протоколу ClientProtocol. Узлы DataNode взаимодействуют с узлом NameNode, используя протокол DataNode Protocol
- И ClientProtocol, и DataNode Protocol «обернуты» в Remote Procedure Call (RPC). Узел NameNode никогда не инициализирует вызовы RPC – он только отвечает на RPC-вызовы узлов DataNode и клиентов .

Файловые операции и репликация



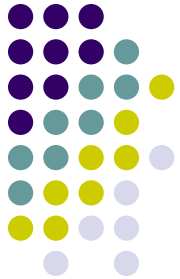
- Набор допустимых файловых операций в распределенной файловой системе HDFS схож с набором файловых операций в «локальных» файловых системах за исключением операции модификации файла – модификация в HDFS не поддерживается по причинам, связанным с архитектурными особенностями (в том числе и вопросами производительности и блокировок) этой файловой системы.
- За все файловые операции отвечает узел NameNode. Операции с конкретными файлами находятся в зоне ответственности узла DataNode, на котором эти файлы находятся.

Файловые операции и репликация



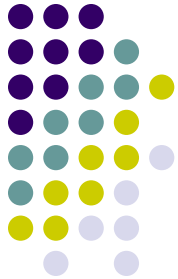
- Изначально клиент кэширует необходимую для записи информацию где-то во временном (или постоянном – его дело) хранилище.
- После того, как объём информации достигает предполагаемого клиентом размера блока в HDFS, клиент отправляет на узел NameNode запрос на создание файла, опционально указав размер блока для создаваемого файла и количество реплик.
- Узел NameNode отвечает клиенту, отправив в ответ идентификатор узла данных и блок назначения, на который будет вестись запись.
- Также узел NameNode уведомляет другие узлы DataNode, на которые будут писаться реплики файлового блока.

Файловые операции и репликация



- После начала передачи файлового потока узлу DataNode, принимающий узел начинает автоматическую ретрансляцию файлового блока на другие узлы реплики.
- Окончание записи файлового блока фиксируется в журнале узла имен.
- Все файловые блоки реплицируются указанное клиентом при создании раз.
- Вторая реплика файлового блока хранится на другом узле, а третья – на узле, расположенном на другой стойке. Расположение следующих реплик вычисляется произвольно.
- Если узел NameNode не принимает от узла DataNode heartbeat-сообщений, то узел имен помечает это узел DataNode как «умерший» и реплицирует данные, хранящиеся на «умершем» узле из «оставшихся в живых» копий.

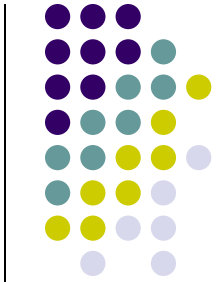
Ограничения HDFS



Файловая система HDFS обладает следующими ограничениями:

- узел имен NameNode является единой точкой отказа;
- отсутствие полноценной репликации Secondary NameNode;
- отсутствие возможности дописывать или оставить открытым для записи файлы в HDFS (как следствие, в plain Hadoop отсутствует поддержка обновляемых и потоковых данных);
- отсутствие поддержки реляционных моделей данных;
- отсутствие инструментов для поддержки ссылочной целостности данных;
- низкая безопасность данных.

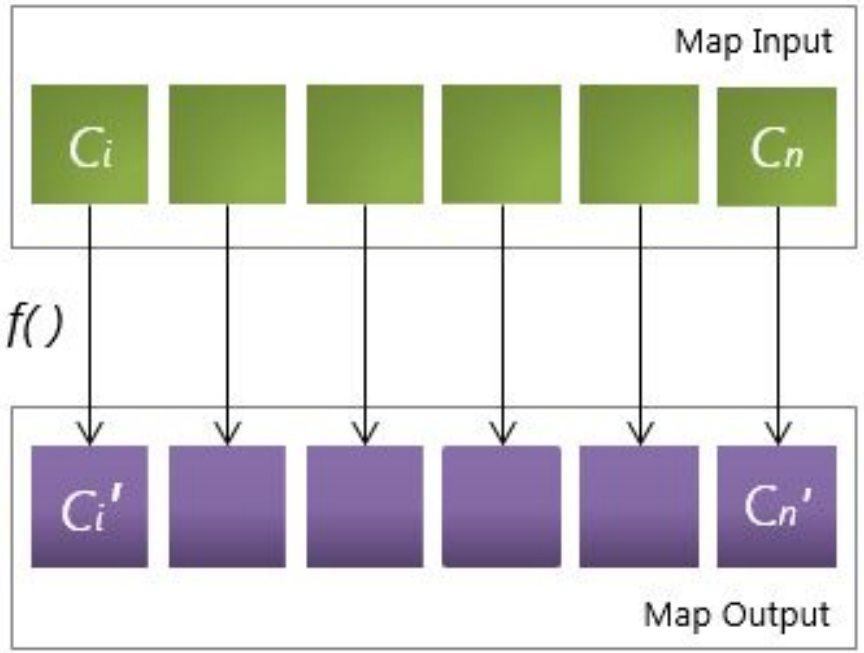
Hadoop MapReduce (общие сведения)



- Выполнение распределенных задач на платформе Hadoop происходит в рамках парадигмы map/reduce.
- **map/reduce** – это парадигма (программная модель) выполнения распределенных вычислений для больших объемов данных.
- В общем случае, для map/reduce выделяют 2 фазы
  - Первый шаг (Map) заключается в первичная обработке данных: компьютер, называемый главным узлом — master node, получает входные данные, разделяет их на части и передает другим компьютерам (рабочим узлам — worker node) для предварительной обработки.
  - Второй шаг (Reduce) заключается в агрегации обработанных данных: главный узел получает ответы рабочих узлов и формирует результат.

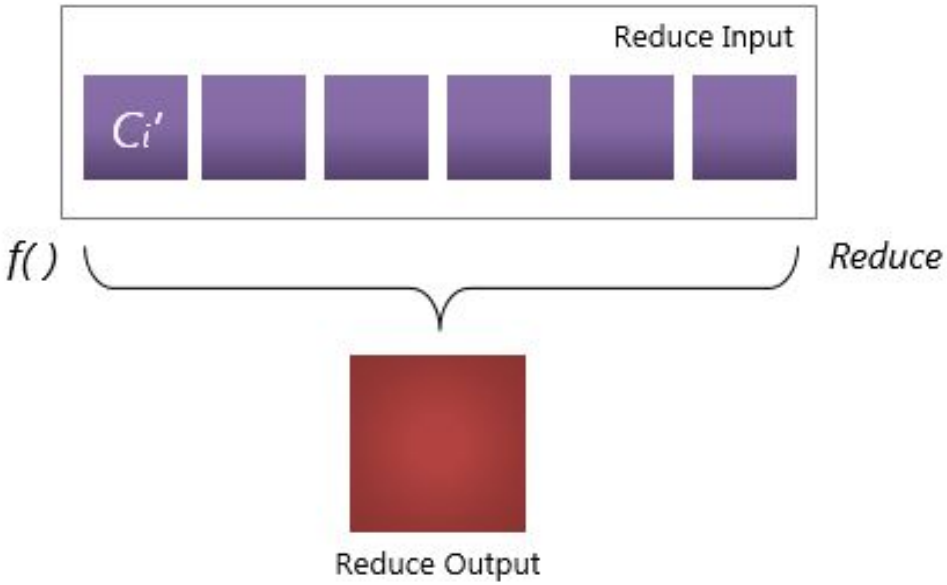
Фаза map(*f*,c)

- Принимает функцию *f* и список *c*. Возвращает выходной список, являющийся результатом применения функции *f* к каждому элементу входного списка *c*.

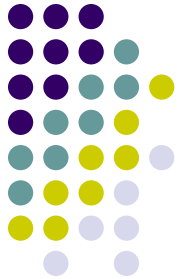


Фаза reduce(*f*,c)

- Принимает функцию *f* и список *c*. Возвращает объект, образованный через свертку коллекции *c* через функцию *f*.

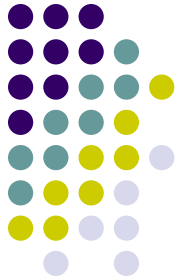


Программная модель map/reduce



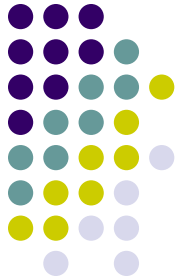
- Программная модель map/reduce была позаимствована из функционального программирования, хотя в реализации Hadoop и имеет некоторые семантические отличия от прототипа в функциональных языках.
- Как и в функциональных языках, при использовании программной модели map/reduce:
  - входные данные не изменяются;
  - разработчик кодирует, что нужно сделать, а не как нужно сделать.

Примеры программных реализаций модели map/reduce



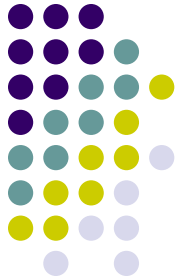
- *Google MapReduce* – закрытая реализация от Google на C++;
- *CouchDB и MongoDB* – реализации для NoSQL баз данных;
- *Hadoop MapReduce* – открытая реализация на Java для Apache Hadoop

Hadoop MapReduce – детальный обзор



- **Hadoop MapReduce** – программная модель (framework) выполнения распределенных вычислений для больших объемов данных в рамках парадигмы map/reduce, представляющая собой набор Java-классов и исполняемых утилит для создания и обработки заданий на параллельную обработку.
- Основные концепции Hadoop MapReduce можно сформулировать как:
  - обработка/вычисление больших объемов данных;
  - масштабируемость;
  - автоматическое распараллеливание заданий;
  - работа на ненадежном оборудовании;
  - автоматическая обработка отказов выполнения заданий.

Этапы работы Hadoop MapReduce

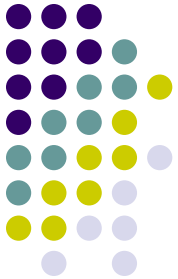


Работу Hadoop MapReduce можно условно поделить на следующие этапы:

- **Input read**
  Входные данные делятся на блоки данных предопределенного размера (от 16 Мб до 128 Мб) –*сплиты* (от англ. split). MapReduce Framework закрепляет за каждой функцией Map определенный сплит.
- **Map**
  Каждая функция Map получает на вход список пар «ключ/значение» <k,v>, обрабатывает их и на выходе получает ноль или более пар <k',v'>, являющихся промежуточным результатом.
  map(k, v) -> [(k', v')]где k' - в общем случае, произвольный ключ, не совпадающий с k.

Все операции map() выполняются параллельно и не зависят от результатов работы друг друга. Каждая функция map() получает на вход свой уникальный набор данных, не повторяющийся ни для какой другой функции map().

Этапы работы Hadoop MapReduce



- **Partition / Combine**
  Целью этапа *partition* (разделение) является распределение промежуточных результатов, полученных на этапе map, по reduce-заданиям.
  (k', reducers_count) -> reducer_id, где reducers_count - количество узлов, на которых запускается операция свертки;
  reducer_id - идентификатор целевого узла.
  В простейшем случае,reducer_id = hash(k') mod reducers_count

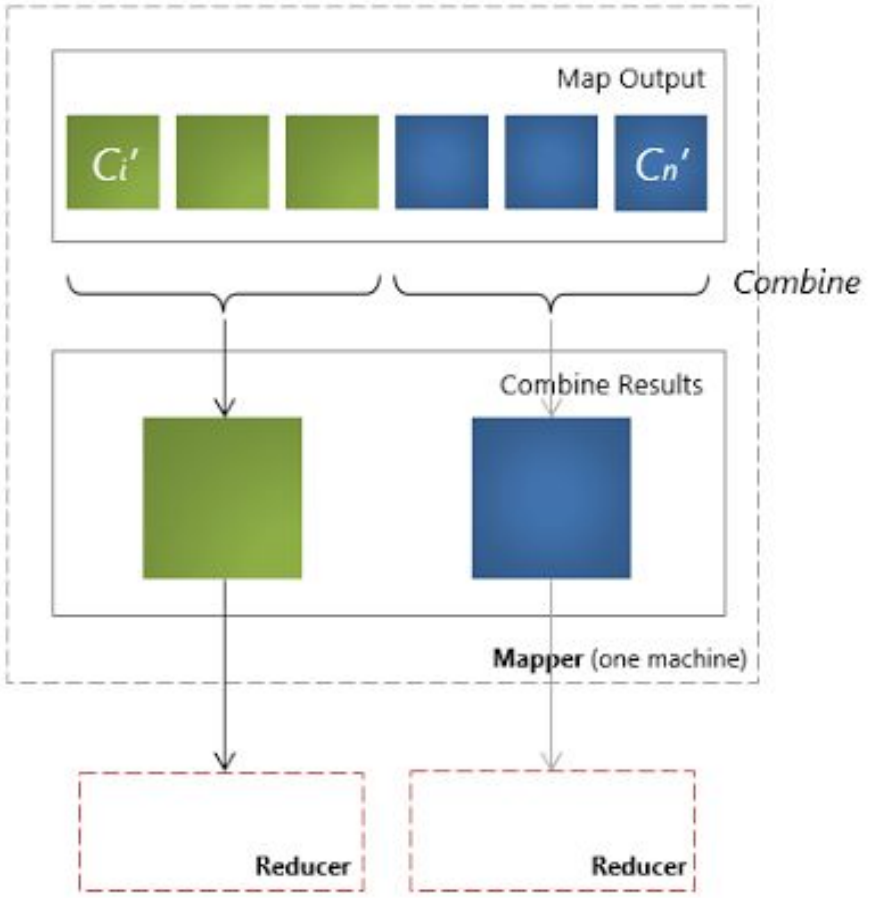
Основная цель этапа partition – это балансировка нагрузки. Некорректно реализованная функция partition может привести к неравномерному распределению данных между reduce-узлами.

Этапы работы Hadoop MapReduce

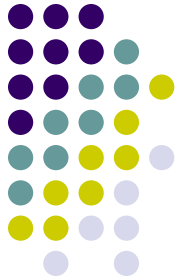
Функция *combine* запускается после map-фазы.

В ней происходит промежуточная свертка, локальных по отношению к функции map, значений.[(k', v')] -> (k', [v'])

Основное значение функции combine – комбинирование промежуточных данных, что в свою очередь ведет, к уменьшению объема передаваемой между узлами информации.

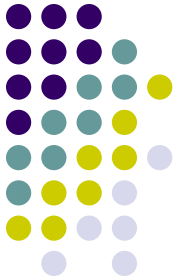


Этапы работы Hadoop MapReduce

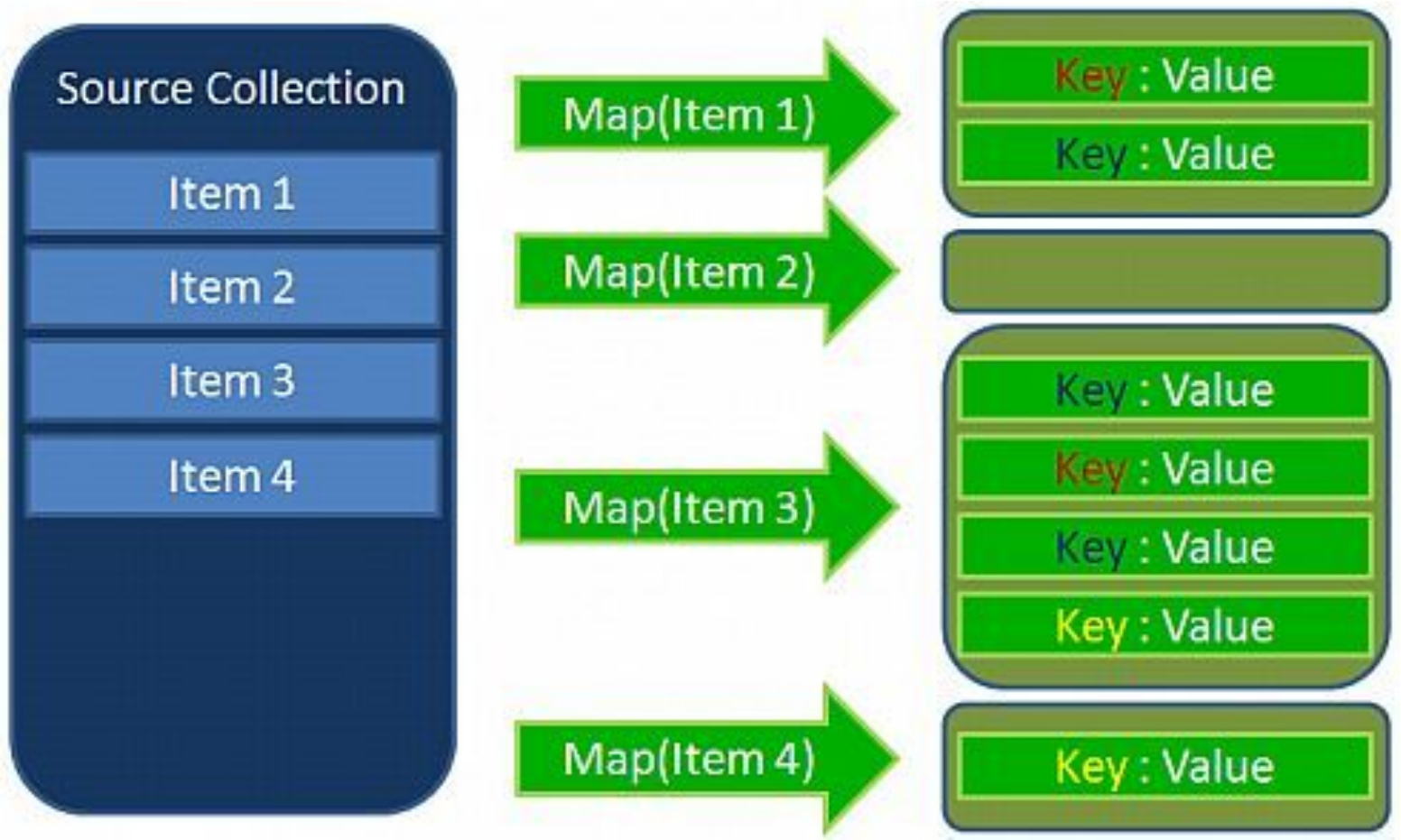


- **Copy** / **Compare** / **Merge**
  На этом этапе происходит:
  - *Copy*: копирование результатов, полученных в результате работы функций map и combine (если такая была определена), с map-узлов на reduce-узлы.
  - *Compare (или Sort)*: сортировка, группировка по ключу k полученных в результате операции copy промежуточных значений на reduce-узле.compare(k'n, k'n+1) -> {-1, 0, +1}
  - *Merge*: «слияние» данных, полученных от разных узлов, для операции свёртки.

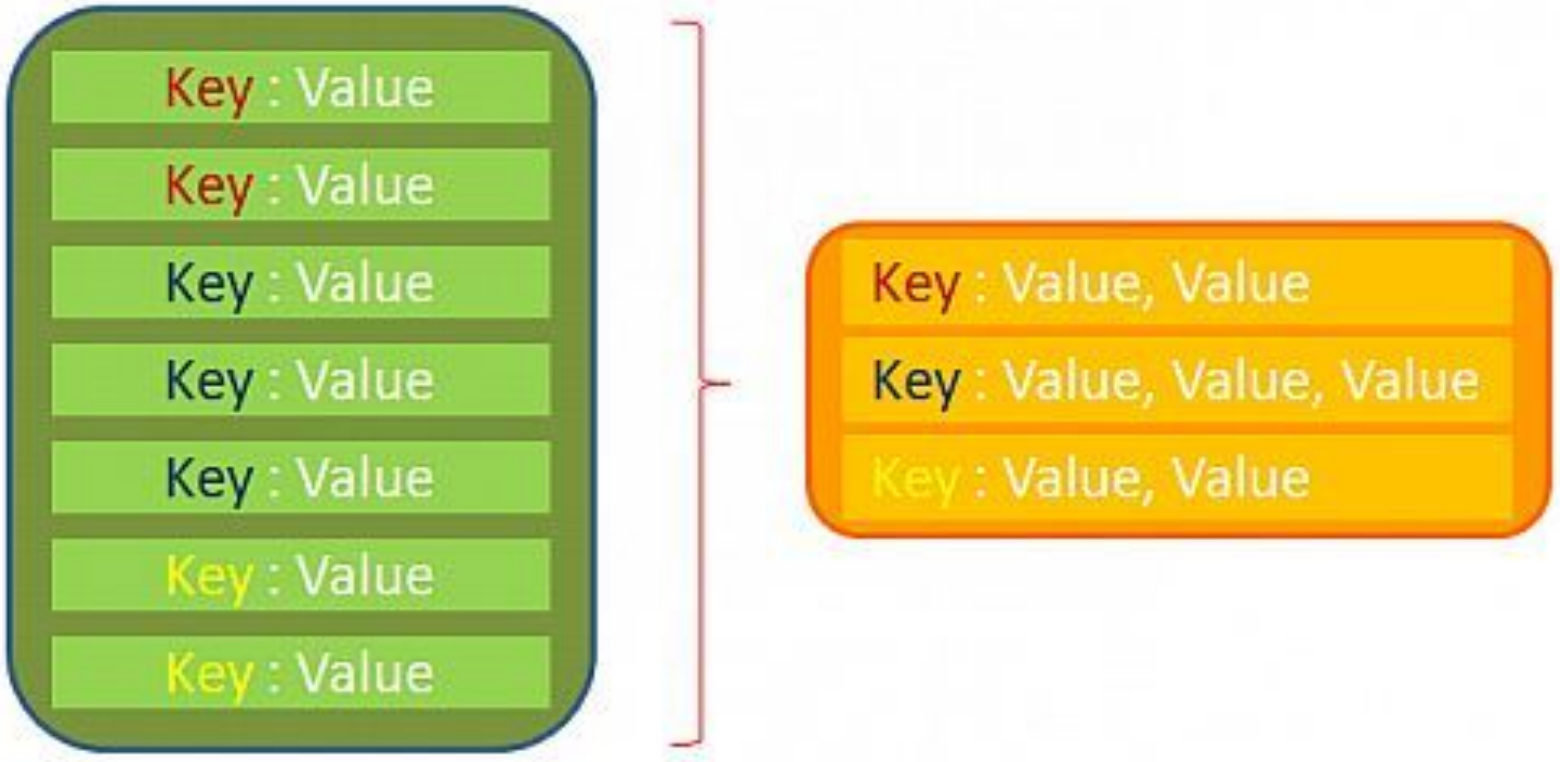
Этапы работы Hadoop MapReduce



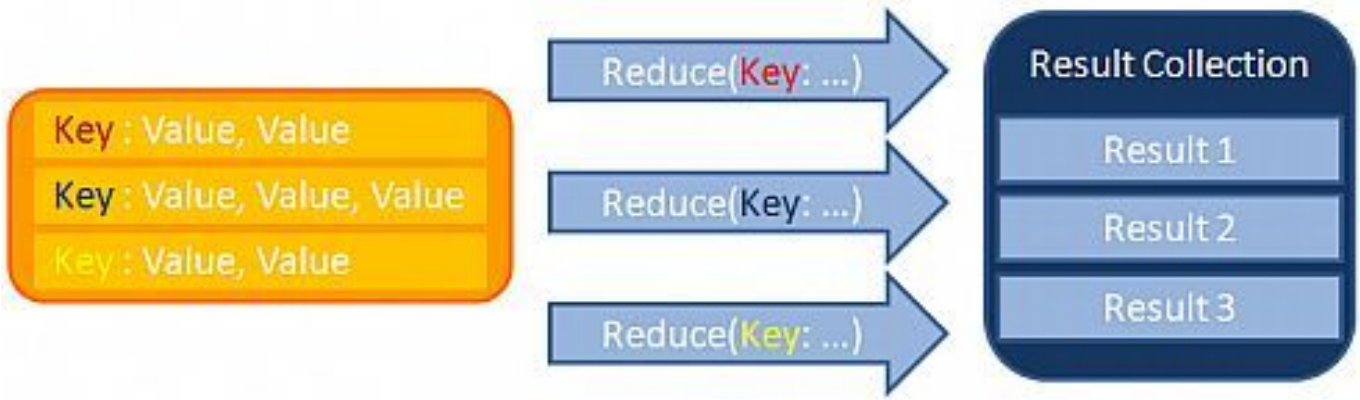
- **Reduce**
  Framework вызывает функцию reduce для каждого уникального ключа k' в отсортированном списке значений.
  reduce(k', [v']) -> [v'']Все операции reduce() выполняются параллельно и не зависят от результатов работы друг друга. Таким образом, результаты работы каждой функции reduce() пишутся в отдельный выходной поток.
- **Output write**
  Результаты, полученные на этапе reduce, записываются в выходной поток (в общем случае, файловые блоки в HDFS). Каждый reduce-узел пишет в собственный выходной поток.



Обязанность Map-функции конвертировать элементы исходной коллекции в ноль или несколько экземпляров Key/Value объектов

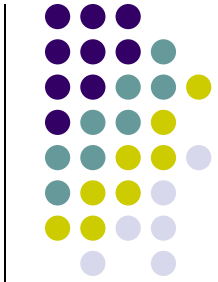


Следующим шагом, алгоритм отсортирует все пары Key/Value и создаст новые экземпляры объектов, где все значения (value) будут сгруппированы по ключу.



В заключении, функция Reduce вернет новый экземпляр объекта, который будет включен в результирующую коллекцию.

Пример



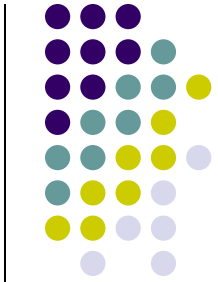
Найти максимальную температуру, наблюдавшуюся в данном городе в определенный период:

Массивы исходных данных

Toronto, 20	Toronto, 17	Toronto, 20	Toronto, 22	Toronto, 20
Whitby, 25	Whitby, 27	Whitby, 20	Whitby, 19	Whitby, 22
New York, 22	New York, 22	New York, 33	New York, 20	New York, 22
Rome, 32	Rome, 32	Rome, 38	Rome, 31	Rome, 29
Toronto, 4	Toronto, 18	Toronto, 32	Toronto, 4	Toronto, 31
Rome, 33	Rome, 37	Rome, 33	Rome, 30	Rome, 30
New York, 18	New York, 32	New York, 30	New York, 18	New York, 19

Ключ – имя города, значение - температура

Пример



Действие функции map – образование последовательностей ключ-значение, где из каждого массива уже выбрана максимальная температура.

(Toronto, 20) (Whitby, 25) (New York, 22) (Rome, 33)

(Toronto, 18) (Whitby, 27) (New York, 32) (Rome,37)

(Toronto, 32) (Whitby, 20) (New York, 33) (Rome, 38)

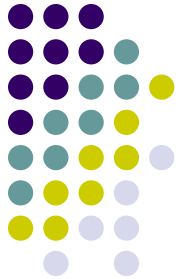
(Toronto, 22) (Whitby, 19) (New York, 20) (Rome, 31)

(Toronto, 31) (Whitby, 22) (New York, 19) (Rome, 30)

Действие функции reduce – объединить ключи и найти максимальную температуру

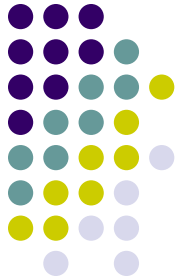
(Toronto, 32) (Whitby, 27) (New York, 33) (Rome, 38)

Hadoop MapReduce – детальный обзор



- Разработчику приложения для Hadoop MapReduce необходимо реализовать базовый обработчик, который на каждом вычислительном узле кластера обеспечит преобразование исходных пар «ключ/значение» в промежуточный набор пар «ключ/значение» (класс, реализующий интерфейс *Mapper*), и обработчик, сводящий промежуточный набор пар в окончательный, сокращённый набор (класс, реализующий интерфейс *Reducer*).

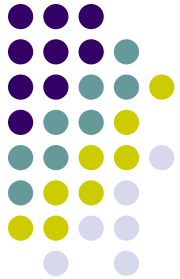
Hadoop MapReduce – детальный обзор



Все остальные фазы выполняются программной моделью MapReduce без дополнительного кодирования со стороны разработчика. Кроме того, среда выполнения Hadoop MapReduce выполняет следующие функции:

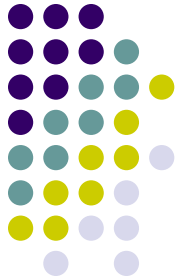
- планирование заданий;
- распараллеливание заданий;
- перенос заданий к данным;
- синхронизация выполнения заданий;
- перехват «проваленных» заданий;
- обработка отказов выполнения заданий и перезапуск проваленных заданий;
- оптимизация сетевых взаимодействий.

Архитектура Hadoop MapReduce



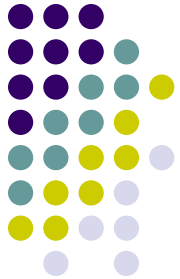
- Hadoop MapReduce использует архитектуру «*master-worker*», где master – *единственный* экземпляр управляющего процесса (*JobTracker*), как правило, запущенный на отдельной машине (вычислительном узле). Worker-процессы – это *произвольное множество* процессов *TaskTracker*, исполняющихся на *DataNode*.
- JobTracker и TaskTracker «лежат» над уровнем хранения HDFS, и запускаются/исполняются в соответствии со следующими правилами:
- экземпляр JobTracker исполняется на NameNode-узле HDFS;
- экземпляры TaskTracker исполняются на DataNode-узле;

Архитектура Hadoop MapReduce



- TaskTracker исполняются в соответствии с принципом «*данные близко*», т.е. процесс TaskTracker располагается топологически максимально близко с узлом DataNode, данные которого обрабатываются.
- принципы расположения JobTracker- и TaskTracker-процессов позволяют существенно сократить объемы передаваемых по сети данных и сетевые задержки, связанные с передачей этих данных – основные «узкие места» производительности в современных распределенных системах.

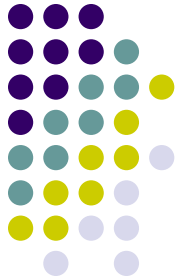
Архитектура Hadoop MapReduce



JobTracker является единственным узлом, на котором выполняется приложение MapReduce, вызываемое программным клиентом. JobTracker выполняет следующие функции:

- планирование индивидуальных (по отношению к DataNode) заданий map и reduce, промежуточных свёрток;
- координация заданий;
- мониторинг выполнения заданий;
- переназначение завершившихся неудачей заданий другим узлам TaskTracker.

Архитектура Hadoop MapReduce



В свою очередь, TaskTracker выполняет следующие функции:

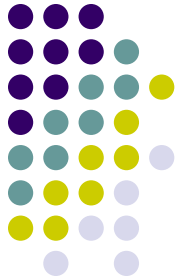
- исполнение map- и reduce-заданий;
- управление исполнением заданий;
- отправка сообщений о статусе задачи и завершении работы узлу JobTracker;
- отправка диагностических heartbeat-сообщений узлу JobTracker.

Взаимодействие TaskTracker-узлов с узлом JobTracker идет посредством RPC-вызовов, причем вызовы идут только от TaskTracker.

Аналогичный принцип взаимодействия реализован в HDFS – между узлами DataNode и NameNode-узлом.

Такое решение уменьшает зависимость управляющего процесса JobTracker от процессов TaskTracker.

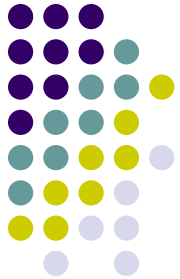
Архитектура Hadoop MapReduce



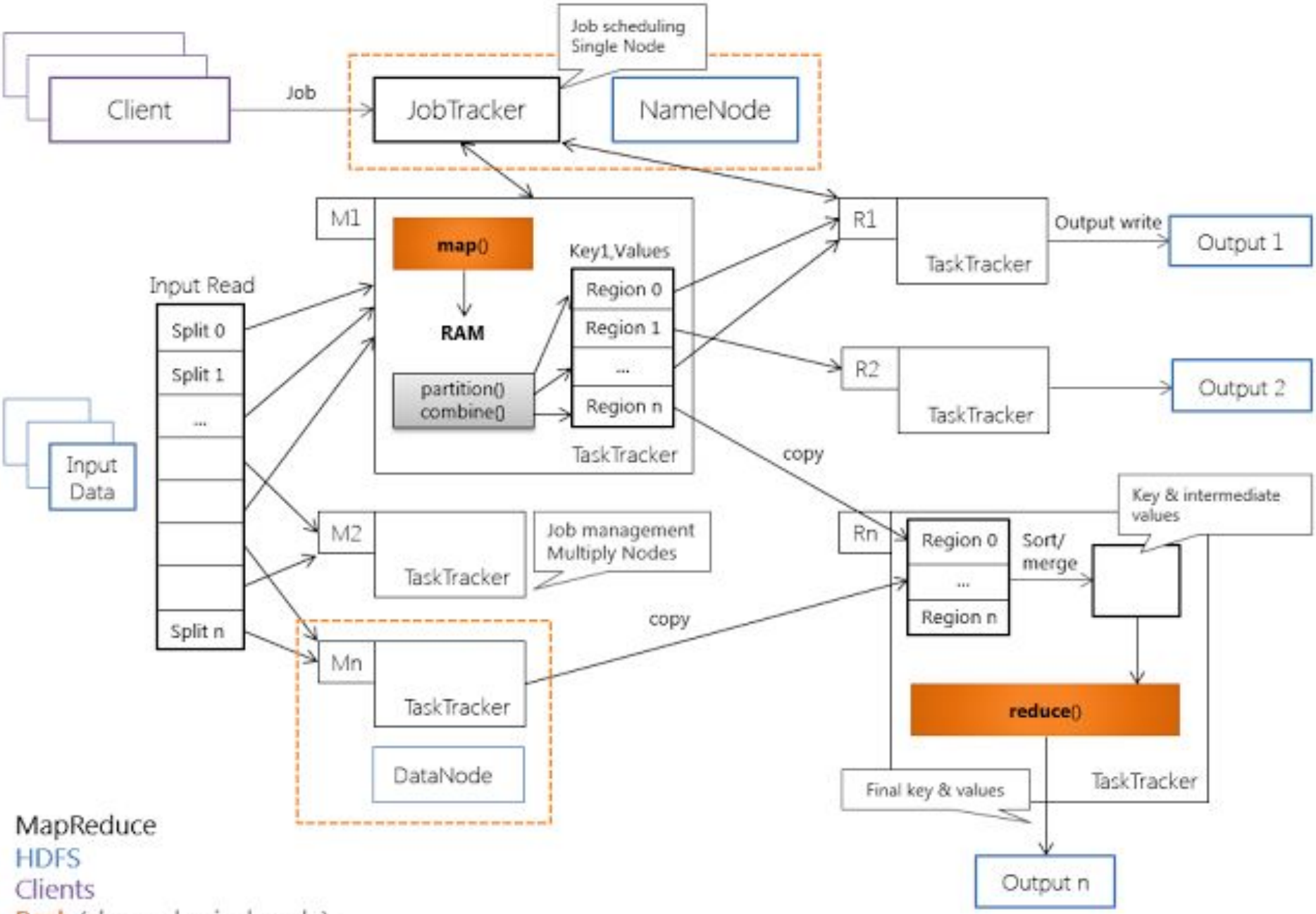
Взаимодействие JobTracker-узла с клиентом (программным) проходит по следующей схеме:

- JobTracker принимает задание (Job) от клиента и разбивает задание на множество M map-задач и множество R reduce-задач.
- Узел JobTracker использует информацию о файловых блоках (количество блоков и их месторасположение), расположенную в узле NameNode, находящемуся локально, чтобы решить, сколько подчиненных задач необходимо создать на узлах типа TaskTracker.
- TaskTracker получает от JobTracker список задач (тасков), загружает код и выполняет его.
- Периодично TaskTracker отсылает JobTracker статус выполнения задачи.
- Взаимодействия TaskTracker-узлов с программным клиентом отсутствуют.

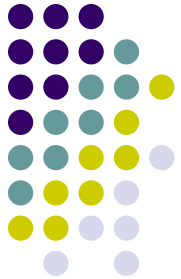
Архитектура Hadoop MapReduce



- По аналогии с архитектурой HDFS, где NameNode является единичной точкой отказа (*Single point of failure*), JobTracker также является таковой.
- При сбое TaskTracker-узла JobTracker-узел переназначает задания неисправного узла другому узлу TaskTracker.
- В случае неисправности JobTracker-узла, для продолжения исполнения MapReduce-приложения, необходим перезапуск JobTracker-узла.
- При перезапуске узел JobTracker читает из специального журнала данные, о последней успешной контрольной точке (checkpoint), восстанавливает свое состояние на момент записи checkpoint и продолжает работу с места последней контрольной точки.

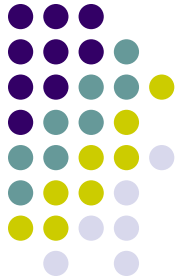

Client
Job
JobTracker
Job scheduling
Single Node
NameNode
M1
map()
Key1,Values
Region 0
Region 1
...
Region n
RAM
partition()
combine()
TaskTracker
Input Read
Split 0
Split 1
...
Split n
Input
Data
M2
TaskTracker
Job management
Multiply Nodes
Mn
TaskTracker
DataNode
copy
copy
R1
TaskTracker
Output write
Output 1
R2
TaskTracker
Output 2
Rn
Region 0
...
Region n
Sort/
merge
Key & intermediate
values
reduce()
Final key & values
TaskTracker
Output n
MapReduce
HDFS
Clients
Rack (share physical node)

Преимущества Hadoop MapReduce



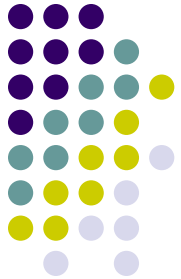
- Эффективная работа с большим (от 100 Гб) объемом данных;
- Масштабируемость;
- Отказоустойчивость;
- Унифицированность подхода;
- Предоставление разработчику сравнительно «чистой» абстракции;
- Снижение требований к квалификации разработчика, в том числе его знаний и опыта по написанию многопоточного кода;
- Дешевизна лицензирования (Open Source).

Ограничения Hadoop MapReduce



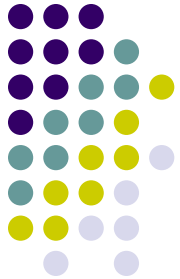
- Смешение ответственности для Reducer (сортировка и агрегация данных). Таким образом, Reducer – это все, что «не map»;
- Отсутствие контроля над потоком данных у разработчика (поток данных управляется фреймворком Hadoop MapReduce автоматически);
- Как следствие предыдущего пункта, невозможность простыми средствами организовать взаимодействие между параллельно выполняющимися потоками.

Недостатки Hadoop MapReduce



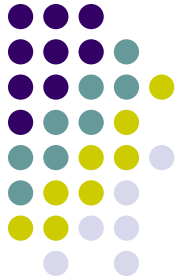
- Применение MapReduce по производительности менее эффективно, чем специализированные решения;
- Эффективность применение MapReduce снижается при малом количество машин в кластере (высоки издержки на взаимодействие, а степень распараллеливания невелика);
- Невозможно предсказать окончание стадии map;
- Этап свертки не начинается до окончания стадии map;
- Как следствие предыдущего пункта, задержки в исполнении любого запущенного map-задания ведут к задержке выполнения задачи целиком;
- Низкая утилизация ресурсов вследствие жесткого деления ресурсов кластера на map- и reduce-слоты.
- Сбой узла JobTracker приводит к простою всего кластера.

Области применения и ограничения Hadoop



- Hadoop MapReduce наиболее эффективно на больших кластерах, где издержки на межсетевые взаимодействие пренебрежительно малы по сравнению со степенью распараллеливания.
- Задачи, решаемые с помощью Hadoop MapReduce, должны отвечать одному основному требованию – они должны относиться к классу задач, параллельных по данным. Под это ограничение попадает широкий спектр исследовательских и бизнес-задач, в число которых входит:
  - анализ данных (*Data Mining*);
  - машинное обучение (*Machine Learning*);
  - индексирование неструктурированных данных и поиск по ним.

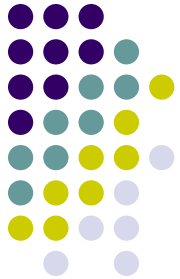
Области применения и ограничения Hadoop



Ограничениям платформы Hadoop складываются как из ограничений отдельных компонентов этой платформы (ограничения HDFS и ограничения Hadoop MapReduce), так и ограничений собственно самой платформы.

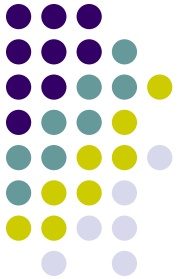
- К последним можно отнести:
  - Ограничение масштабируемости кластера Hadoop: ~4K вычислительных узлов; ~40K параллельных заданий;

Области применения и ограничения Hadoop



- Сильная связанность фреймворка распределенных вычислений и клиентских библиотек, реализующих распределенный алгоритм. Как следствие:
  - Отсутствие поддержки альтернативной программной модели выполнения распределенных вычислений: в Hadoop v1.0 поддерживается только модель вычислений map/reduce.
  - Невозможность использования в средах с высокими требованиями к надежности из-за наличие в платформе Hadoop единичных точек отказа;
  - Проблемы версионной совместимости: требование по единовременному обновлению всех вычислительных узлов кластера при обновлении платформы Hadoop (установке новой версии или пакета обновлений);
  - Отсутствие поддержки работы с обновляемыми/потоковыми данными.

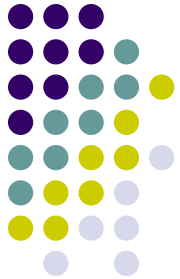
Основные достоинства Hadoop



- Hadoop является способом построения программных комплексов с архитектурой, входящей в подмножество архитектур MIMD – *SPMD* (Single Program Multiple Data).
- программный платформа Hadoop разработана как инфраструктурное решение.

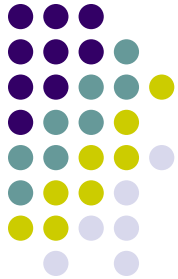
Это позволяет как архитектору ПО, так и разработчику уделять меньшее внимание вопросам, связанным с распараллеливание выполнения программы, и большее, связанное с предметной областью разрабатываемого программного комплекса.

Основные достоинства Hadoop



- Hadoop – уникальный продукт, объединивший многие известные концепции, и позволивший вынести задачу распараллеливания ближе к инфраструктурному уровню, нежели к уровню логики приложения.
- Установка автоматического управления распределенными вычислениями одним из приоритетов заставила пересмотреть позицию относительного многих аспектов функционирования больших вычислительных кластеров.
- Платформа Hadoop – это не просто программная реализация симбиоза нескольких концепций, это качественно новый подход к проектированию приложений для анализа и работы с большими объемами данных.

Недостатки Hadoop



- Среди наиболее частых недостатков Hadoop вызывают: отсутствие у платформы поддержки схем данных (по аналогии со схемами в БД) и чрезмерная низкоуровневость интерфейсов, с которыми приходится работать разработчикам
- Отчасти следствием указанных ограничений является не слишком широкий спектр задач параллельных по данным, который платформа Hadoop может решать более эффективно, чем решения на основе реляционных баз данных.

http://www.codeinstinct.pro/2012/08/hadoop-overview.html