## Все вопросы

1. Что такое большие данные (3v, 4v)?
2. Чем знаменит IBM Watson?
3. Что называют традиционными архитектурами хранения данных?
4. В чем отличие систем хранения больших данных?
5. Что такое Hadoop (HDFS + MapReduce)?
6. На каком языке написан Hadoop?
7. Какие технологии, связанные с Hadoop, вы знаете?
8. Хорошо ли использовать Hadoop для транзакций?
9. Что такое блок в Hadoop?
10. Что такое репликация?
11. Назовите любую команду для работы с Hadoop.
12. Какие узлы есть в Hadoop кластере?
13. Что такое Name Node?
14. Что такое Job Tracker?
15. Что такое Task Tracker?
16. Что такое Data Node?
17. Что на вход принимает операция Map?
18. Что на вход принимает операция Reduce?
19. Что происходит между Map и Reduce?
20. Отказоустойчивость. Что, если откажет Data Node, Name Node, Task Tracker, Job Tracker?
21. Зачем нужен Combiner?
22. Какие языки высокоуровневого программирования вы знаете?
23. Что такое Pig?
24. Что такое Hive? Приведите пример команды на Hive.
25. Что такое Jaql?
26. Что такое Yarn? Основная идея.
27. Что такое Spark? Основная идея.
28. Зачем нужны NoSQL базы данных?
29. Модель данных HBase.
30. Какие средства для распределенной обработки данных на языке R вы знаете?
31. Основные этапы извлечения информации из текстов.
32. Подходы к распознаванию сущностей из текстов.
33. Язык AQL. Компоненты языка AQL.

Доп. вопросы

## Информация

Составлено Антоном Брызгаловым (tony.bryzgaloff@gmail.com) в январе 2019-го года.

## We are hiring!

Если тебе интересна сфера больших данных и стек Hadoop — пиши мне на почту. Я руковожу разработкой хранилища данных и ищу талантливых разработчиков в свою команду.

Даже если с момента написания этого документа прошло несколько лет, и ты хочешь поработать в этой сфере — напиши, я помогу тебе найти свое место в Big Data :)

## 1. Что такое большие данные (3v, 4v)?

Большие данные (и связанная с ними инфраструктура) характеризуются свойствами, известными как 3v:

- **Volume** (объем): размер данных выходит за пределы возможностей программных средств типичной базы данных по сбору, хранению, обработке и анализу данных.
- **Variety** (разнообразие): инфраструктура Big Data должна быть способна обрабатывать и интегрировать множество различных источников, типов и форматов данных. В качестве источников могут выступать «боевые» OLTP-базы, системы сбора данных с веб-сайтов, системы аналитики, файлы, веб-логи, IoT-устройства и пр.
- **Velocity** (скорость): качество характеризует с одной стороны скорость появления новых данных в системах-источниках, а с другой — скорость обработки данных, которая должна быть приближена к реальному времени (т.н. потоковая обработка).

К этому набору свойств могут быть добавлены еще два, образуя 4v и 5v Big Data:

- **Veracity** (достоверность): способ накопления данных не должен нарушать их корректность. Если данным нельзя доверять — их нельзя использовать.
- **Value** (ценность): выводы *(англ. insights)*, сделанные в результате обработки данных, должны быть применимы и иметь практическую ценность.

## 2. Чем знаменит IBM Watson?

- Watson состоит из 90 серверов Power7 750, каждый из которых содержит по четыре 8-миядерных процессора POWER7. Суммарная оперативная память — 15 ТБ.
- В феврале 2011 года Watson принял участие в телешоу Jeopardy! (аналог «Своей игры»). Его соперниками были рекордсмены игры: обладатель самого большого выигрыша и рекордсмен по длительности беспроигрышной серии. Имея доступ к 200 миллионам страниц структурированной и неструктурированной информации объемом 4 ТБ, включая полный текст Википедии, но без доступа к интернету, Watson одержал победу.
- Watson используется в медицине: изучив 605 тыс. медицинских документов (около 2 млн страниц текста) и проанализировав 25 тыс. историй болезни, Watson способен не только ставить диагнозы, но и определять курс лечения. Точность предлагаемых Watson курсов лечения рака легких составляет 90%. Точность врачей — всего 50%.
- В процессе тестирования в колл-центрах IBM Watson на 40% быстрее выдавал клиенту информацию, чем оператор.

## 3. Что называют традиционными архитектурами хранения данных?

Ключевой компонент традиционных архитектур — объектно-реляционное хранилище структурированных данных.

Свойства традиционных архитектур СУБД:

- дисковое хранение;
- транзакции;
- индексация;
- журнализация;
- многопоточность;
- блокировки.

Недостатки:

- Производительность серверов растет медленнее, чем объемы данных.
- Оптимизации и индексы влекут накладные расходы, а следовательно, сильное падение производительности при нагрузках.
- Кеш на чтение влечет усложнение клиентского ПО.
- Очередность операций влечет проблемы хранения и ограниченности очереди.
- Горизонтальное масштабирование влечет денормализацию и ограничения JOIN’ов.
- Денормализация приводит к избыточности данных.

Пути решения:

- Отказ от строгой консистентности.
- Денормализация. И, следовательно, внедрение избыточности.
- Потеря выразительности SQL и моделирование части функций программно.
- Усложнение клиентского ПО.

4. В чем отличие систем хранения больших данных?
   - **Хорошо масштабируется** благодаря shared nothing архитектуре (SNA). SNA — архитектура независимых распределенных вычислений, в которой отдельные узлы имеют собственную память, диски и устройства ввода/вывода. Каждый узел в такой архитектуре самодостаточен и ничем не делится с другими узлами сети.
   - Поддерживают **произвольные форматы данных**, в том числе слабо- и неструктурированные.
   - Имеют простой и эффективный механизм обеспечения **отказоустойчивости**.

5. Что такое Hadoop (HDFS + MapReduce)?

   Hadoop — open-source экосистема вокруг больших данных. Hadoop позволяет фильтровать, сортировать, хранить и обрабатывать огромные объемы данных, в том числе неструктурированных.

   HDFS (Hadoop Distributed File System) — файловая система Hadoop, работающая поверх файловой системы операционной системы. Поддерживает большие файлы с возможностями потокового доступа к данным.

   MapReduce — модель распределённых вычислений, предложенная Google, которая позволяет обрабатывать в компьютерных кластерах данные объемом до нескольких петабайт. Обработка данных состоит из двух шагов:
   - На Map-шаге главный узел (master node) разделяет входные данные на независимые части и передает их рабочим узлам (worker nodes) для параллельной обработки: для каждой строки входных данных Map-операция выдает ни одной, одну или несколько промежуточных строк.
   - Затем промежуточные данные сортируются, группируются и обрабатываются в рамках Reduce-шага: все строки из каждой группы поступают на один рабочий узел и обрабатываются на нем.

6. На каком языке написан Hadoop?

   Java.

7. Какие технологии, связанные с Hadoop, вы знаете?

   В лекциях перечислены (даны ссылки на соответствующие билеты):
   - IBM Eclipse — платформа для построения интегрированных сред разработки;
   - Lucene — библиотека для полнотекстового поиска;
   - HBase — NoSQL (wide-column) СУБД;
   - Hive — средство построения хранилища данных на базе Hadoop, имеющее SQL-интерфейс;
   - Pig — аналитическая платформа;
   - Jaql — средство для работы с форматом JSON в Big Data;
   - ZooKeeper — служба управления и координации для распределенных приложений;
   - и др.

8. Хорошо ли использовать Hadoop для транзакций?

   Hadoop предназначен для аналитических задач (OLAP), и не является OLTP-решением (в том числе не поддерживает транзакции).

9. Что такое блок в Hadoop?

   Блок данных в Hadoop имеет размер 64 МБ по умолчанию (рекомендованный размер — 128 МБ). Для сравнения, блок файловой системы Unix — 4 КБ. Блоки одного файла по возможности распределяются по разным data nodes, обеспечивая доступность и отказоустойчивость.

10. Что такое репликация?

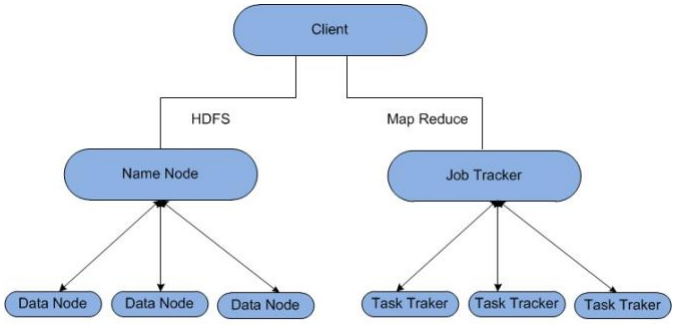
    Блоки дублируются на нескольких data nodes, чтобы данные оставались доступными в случае недоступности одного из мест хранения. По умолчанию replication factor — 3, т.е. каждый блок хранится в трех экземплярах на трех разных машинах кластера.

11. Назовите любую команду для работы с Hadoop.

    Hadoop имеет консольный клиент (CLI), пример вызова: `hadoop fs -ls .` — аналог `ls .` в Unix.

## 12. Какие узлы есть в Hadoop кластере?

- Узлы HDFS:
  - Name Node;
  - Data Node;
- Узлы MapReduce:
  - Job Tracker;
  - Task Tracker.




Логически архитектура имеет схему, изображенную справа. Физически одни и те же узлы могут выступать в нескольких ролях одновременно.

Также есть другие узлы, не рассмотренные в курсе (например, Secondary Name Node).

## 13. Что такое Name Node?

Name Node — главный узел (master node). Особенности:

- Единственный на весь кластер.
- Управляет метаданными и пространством имен файловой системы.
- Single point of failure (точка отказа). Поэтому:
  - В кластере две машины конфигурируются как Name Node, одна из которых находится в состоянии Active, другая — Standby. Вторая является подчиненной первой, самостоятельно следит за синхронизацией и берет управление при необходимости.
  - Аппаратная составляющая Name Node обычно более надежная, чем у Data Nodes.

## 14. Что такое Job Tracker?

Job Tracker получает запросы от клиента на запуск MapReduce операций, планирует и отслеживает их выполнение. В кластере присутствует в единственном экземпляре (с аналогичными особенностями отказоустойчивости, что и Name Node).

## 15. Что такое Task Tracker?

Узлы типа Task Tracker непосредственно выполняют код MapReduce операций и читают данные с Data Nodes.

## 16. Что такое Data Node?

Data Node — узел, непосредственно хранящий и отдающий клиенту данные. Особенности:

- В состав одного кластера входит множество Data Nodes.
- Периодически сообщает Name Node информацию о содержащихся блоках.
- Традиционно для Data Node используется более дешевое и универсальное аппаратное обеспечение, чем для Name Node.

## 17. Что на вход принимает операция Map?

Map на вход получает **входные данные** построчно в формате пар <key, value>. Каждая строка входных данных в результате Map-операции может быть преобразована:

- в одну промежуточную строку;
- ни одну, т.е. пропущена, например, в случае фильтрации: если входная строка не удовлетворяет условиям фильтра, на Map-шаге она исключается;
- или несколько выходных строк, например, при разбиении входной строки текста на отдельные слова.

Промежуточные строки также имеют вид пары <key, value> (но и key, и value уже могут отличаться от значений, поданных на вход). Далее сортируются по ключу (key), группируются и передаются на вход операции Reduce. Ключи выходных пар должны быть сравнимыми, чтобы по ним можно было сортировать и группировать.

По факту между Map и Reduce присутствует операция Combine, но ей посвящен отдельный билет и в общей схеме (MapReduce) она не участвует.

## 18. Что на вход принимает операция Reduce?

Операция Reduce принимает на вход **группы пар <key, value>** с одинаковыми ключами: все промежуточные строки, полученные в результате Map-операции, с одинаковым ключом попадают в один редьюсер.

Таким образом, редьюсер может оперировать со всеми строками одной группы. (В отличие от Map, которая обрабатывает каждую строку независимо.)

В результате Reduce-шага, аналогично Map, группы промежуточных строк могут быть преобразованы в одну, ни одну или несколько выходных строк. Формат выходных строк — тоже пары <key, value>, где и key, и value снова могут отличаться от поданных на выход.

Общая схема: (input) <k1, v1> -> map -> <k2, v2> -> combine -> <k2, v2> -> reduce -> <k3, v3> (output).

## 19. Что происходит между Map и Reduce?

Между Map и Reduce существует промежуточный шаг: **Combine**. Подробнее в отдельном билете.

На Reduce-шаге операции reduce предшествуют следующие этапы:

- **Shuffle** (перемешивание): этап передачи выходных данных Map-шага на вход редьюсерам. Для каждой промежуточной строки (выходной строки Map) определяется, на какой машине будет выполняться соответствующий ей редьюсер (т.е. редьюсер, работающий с группой строк с тем же ключом).
- **Sort** (сортировка): входные данные для Reduce сортируются по ключу для разбиения на отдельные Reduce-задачи. Каждому ключу сопоставляется группа соответствующих ему записей, которая обрабатывается в рамках отдельной Reduce-задачи.
- **Secondary Sort** (дополнительная сортировка): помимо сортировки пар по ключу может потребоваться пользовательская сортировка данных внутри группы. Если дополнительная сортировка значений не сконфигурирована пользователем, этап пропускается.

Shuffle и Sort выполняются параллельно: по мере завершения Map-задач их выходные данные рассылаются по редьюсерам и упорядочиваются слиянием (merge).

Если Reduce-шаг в задаче не используется, то никакие связанные с ним этапы, в том числе сортировка, не выполняются. Выходные данные Map-шага становятся выходными данными всей MapReduce-операции.

## 20. Отказоустойчивость. Что, если откажет Data Node, Name Node, Task Tracker, Job Tracker?

- **Data Node** отчитывается о работоспособности (heartbeat) Name Node. Если в течение 10 минут сигнал от Data Node не поступает, Data Node считается недоступной. Name Node запускает репликацию находившихся на утраченной Data Node блоков (т.к. их копии хранятся на других Data Nodes).
- В случае отказа **Name Node** кластер становится недоступен (single point of failure). Name Node должна быть перезапущена или перенесена на другую машину. Способы повышения отказоустойчивости упомянуты в билете по Name Node (Secondary Name Node, надежное аппаратное обеспечение).
- **Task Tracker** отчитывается о работоспособности (heartbeat) Job Tracker'у. Если Task Tracker перестает отправлять сигнал, Job Tracker исключает его из пула и перестает назначать ему задачи. Связанные с ним MapReduce-задачи перезапускаются, даже если Map-шаг завершен.
- **Task failure**. Также неуспешно может выполниться сама Map- или Reduce-задача. В таком случае Task Tracker помечает попытку выполнения задачи как неуспешную, освобождает ресурсы под выполнение других задач и сообщает об этом Job Tracker. Тот может перезапустить задачу (если количество неуспешных попыток не исчерпано) или признать всю MapReduce-задачу невыполненной.
- В случае отказа **Job Tracker** все выполняющиеся задачи неуспешно завершаются, и информация о них теряется. После перезапуска Job Tracker'а все запущенные на нем в момент отказа задачи должны быть заново отправлены пользователем в Job Tracker. Решением проблемы single point of failure для Job Tracker стал YARN. См. соответствующий билет.

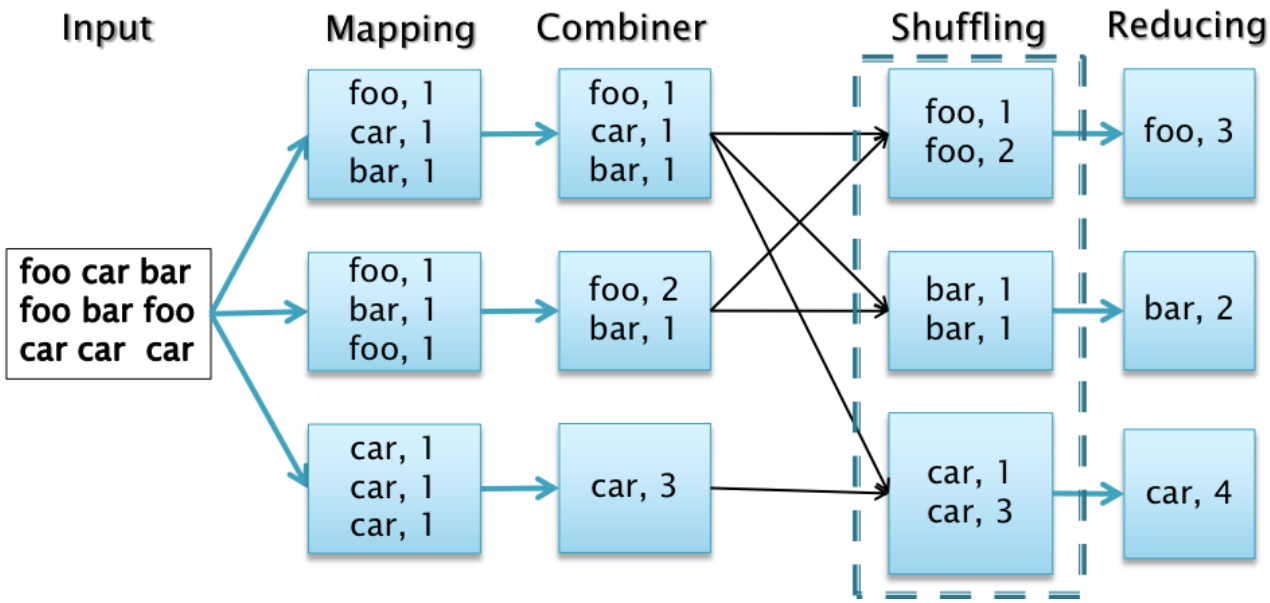
## 21. Зачем нужен Combiner?

Операция Combine получает на вход результат работы Map-шага и может провести их предварительную агрегацию.

Пример. Задача подсчета слов в тексте.

> На шаге Map каждая строка текста разбивается на слова, в результате получаются пары <word, 1>. На шаге Reduce для каждого слова-ключа word суммируются все значения, т.е. все единицы. Таким образом для каждого слова посчитаны все вхождения в текст.
>
> Map выполняются параллельно на нескольких машинах. Reduce-шаг начинается только после того, как выполнены все Map-задачи.

Оптимизация: вместо пар <word, 1> можно передать в Reduce пары <word, n>, где n — количество раз, когда слово word встретилось в одном Map. Для этого с помощью Combine может быть выполнена предагрегация результатов Map, аналогичная Reduce: для каждого word предподсчитаем, сколько раз слово встретилось на одном mapper'е. Иллюстрация ниже:




Отличия Combine от Reduce:

- На вход комбинирующей функции попадают не все значения, соответствующие одному ключу, а только пары, являющиеся результатом одной Map-задачи.
- Combiner'ы выполняются параллельно сразу по завершению Map-задачи. Reduce ожидает полного завершения Map-шага.
- Combiner'ы выполняются на той же машине, что и Map, что позволяет избежать передачи данных по сети.
- Hadoop не гарантирует того, что Combine-шаг будет запущен, поэтому форматы выходных значений Map и Combine должны совпадать.

## 22. Какие языки высокоуровневого программирования вы знаете?

В контексте Hadoop можно перечислить связанные с ним технологии из соответствующего билета. Большинство из них: например, Pig, Hive (HiveQL), HBase, Jaql — предоставляют SQL-подобный интерфейс для работы с данными, хранящимися в HDFS.

## 23. Что такое Pig?

Pig — аналитический инструмент поверх Hadoop. Он включает в себя язык Pig Latin и среду исполнения написанных на нем сценариев. Сценарий описывает серию преобразований над входными данными, в результате которых буду получены выходные данные. Среда исполнения преобразует сценарий в серию MapReduce-операций.

Pig упрощает обработку данных:

- Нет необходимости самостоятельно писать код для стандартных SQL-like операций: filter, join, group by и т.д. — они эффективно реализованы в Pig Latin.
- Снимает ограничения двухфазности MapReduce модели. Pig-сценарий состоит из произвольного числа шагов.
- Процедурный стиль описания преобразований: потоки данных можно присваивать переменным и использовать их в нескольких выражениях.

Пример сценария Pig Latin (задача найти топ-10 веб-страниц для каждой категории):

```
visits      = LOAD '/data/visits' AS (user, url, time);
gVisits     = GROUP visits BY url;
visitsCount = FOREACH gVisits GENERATE url, count(visits);
urlInfo     = LOAD '/data/urlInfo' AS (url, category, pRank);
visitCounts = JOIN visitCounts BY url, urlInfo BY url;
gCategories = GROUP visitCounts BY category;
topUrls     = FOREACH gCategories GENERATE top(visitCounts, 10);
STORE gVisits INTO '/data/groupedVisits';
```

## 24. Что такое Hive? Приведите пример команды на Hive.

Hive — open source решение для построения хранилищ данных (data warehouse) на платформе Hadoop. В состав Hive входит Hive Query Language (HQL) — SQL-like язык запросов.

Пример команды на HQL:

```
INSERT OVERWRITE TABLE t1
SELECT * FROM t2;
```

Hive берет на себя задачи по (де-)сериализации, секционированию (логическому разбиению — например, по датам) и бакетированию (физическому разбиению — например, по первичному ключу) данных, а также обработке запросов к ним, предоставляя высокоуровневый SQL-like интерфейс.

HiveQL имеет ограничения по сравнению с полным стандартом ANSI SQL: например, из операций JOIN доступно только эквисоединение (equi-joins) таблиц, например: `t1.id == t2.t1_id` — т.е. по равенству значений в столбцах.

## 25. Что такое Jaql?

Jaql — JSON Query Language, язык и среда исполнения для него, предназначенные для обработки данных в JSON, что позволяет эффективно работать не с «плоскими» (flat) данными (прерогатива Pig/Hive), а со вложенными структурами.

По факту является не только языком запросов, а полноценным языком программирования, предоставляющий полный контроль над потоком выполнения.

Jaql SQL является одним из компонентов Jaql и предоставляет SQL-интерфейс с теми же ограничениями, что и Hive (например, только equi-joins). SQL-выражения могут быть использованы как часть инструкций по обработке данных в обычной Jaql-программе.

## 26. Что такое Yarn? Основная идея.

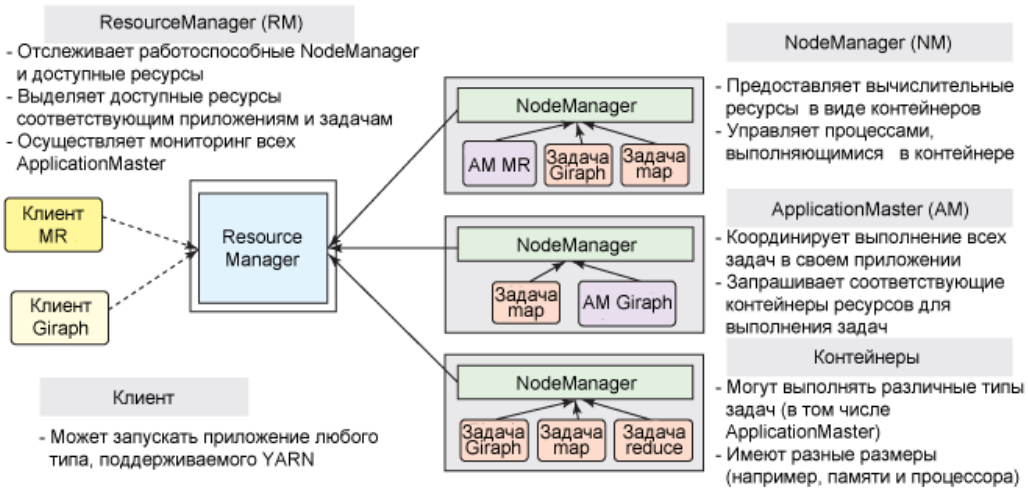
MapReduce v1 в реализации Apache Hadoop имеет единственный Job Tracker на весь кластер, что чревато следующими проблемами:

- Единственный Job Tracker является single point of failure. Отказ этого узла ведет к полной невозможности выполнять задачи и потере данных задач, выполнявшихся на момент отказа.
- Большое число обязанностей, порученных одному процессу, приводит к практическим ограничениям масштабируемости: Job Tracker приходится постоянно отслеживать тысячи Task Tracker'ов, сотни заданий и десятки тысяч задач Map и Reduce.

Для решения проблемы масштабируемости предлагается простая, но эффективная идея: снять часть обязанностей с одного процесса Job Tracker и делегировать некоторые из них процессам Task Tracker.

YARN — Yet Another Resource Negotiator, разработанный как часть нового поколения вычислительной платформы Hadoop.

Архитектура представлена на изображении (описание компонентов — ниже):




В архитектуре YARN глобальный **Resource Manager** отслеживает работоспособные узлы и доступные ресурсы в кластере и распределяет их между приложениями, запускаемыми пользователями.

Когда пользователь отправляет приложение на выполнение, запускается экземпляр нетребовательного к ресурсам процесса **Application Master** для координации выполнения всех задач приложения, а именно: мониторинга задач, перезапуска задач после сбоев, упреждающего выполнения медленных задач. Ранее эти обязанности для всех заданий выполнял один процесс Job Tracker.

Application Master и задачи его приложения выполняются в контейнерах ресурсов, управляемых процессами Node Manager. **Node Manager** — это более универсальная и эффективная версия Task Tracker. Вместо фиксированного числа слотов map и reduce Node Manager имеет несколько динамически создаваемых контейнеров ресурсов.

В процессе планирования выполнения пользовательского приложения Resource Manager использует подключаемый планировщик (Scheduler). **Scheduler** занимается только планированием, определяя очередность получения ресурсов кластера (в виде контейнеров), но не выполняет мониторинг задач в приложении и не пытается перезапустить задачи после сбоев.

Для Resource Manager, Node Manager и контейнера не важен тип приложения или задачи. Весь специфичный для конкретной инфраструктуры код приложения помещается в его Application Master, чтобы YARN мог поддерживать любую распределенную инфраструктуру (например, Giraph, Storm, Spark и т.п.), если кто-то реализует для нее соответствующий Application Master.

Этот универсальный подход позволяет выполнять в кластере Hadoop YARN самые разные рабочие нагрузки, не только MapReduce, как в версии MRv1.

## 27. Что такое Spark? Основная идея.

Spark — платформа (engine) распределенной обработки данных, которая активно использует оперативную память. В отличие от Hadoop, использующего в качестве хранилище только диски.

Преимущества использования Spark:

- В Spark эффективно реализованы множество высокоуровневых операторов, поэтому его использование освобождает от необходимости реализовывать стандартные операции самостоятельно.
- Аналогично Pig (см. билет), Spark позволяет процедурно оперировать потоками данных.
- Spark поддерживает множество распределенных источников данных, в том числе HDFS.
- Существуют API для Scala, Java, Python и R.

## 28. Зачем нужны NoSQL базы данных?

NoSQL — Not Only SQL — решения призваны исправить недостатки традиционных СУБД (реляционных), с которыми столкнулись разработчики высоконагруженных веб-приложений:

- Требуется значительно более высокая скорость операций чтения-записи, чем могут обеспечить реляционные решения, ограниченные необходимостью обеспечивать строгую консистентность данных (за счет транзакций и ссылочной целостности).
- Частые обновления схемы.
- Большой объем данных, не вмещающийся на одну машину. Горизонтальное масштабирование (увеличение количества машин) невозможно, т.к. традиционные РСУБД не рассчитаны на многомашинную архитектуру в силу особенностей реляционной модели.

Существуют способы преодоления этих препятствий без отказа от традиционных РСУБД, однако более системным решением явилась разработка СУБД нового класса — NoSQL. СУБД этого класса характеризуются ослаблением свойств ACID, т.к. используются в сферах, менее требовательных к целостности данных (обычно — веб-приложения, социальные сети), чем область предназначения реляционных СУБД (часто — финансовая сфера).

NoSQL решения включают в себя четыре вида хранилищ: ключ-значение, колоночные, документные и графовые. Эти решения хорошо масштабируются, обеспечивают высокую скорость на запись, чтение и обновление данных, а также в основном позволяют хранить данные без определения схемы, однако не обеспечивают строгой консистентности ACID.

## 29. Модель данных HBase.

HBase — распределенная колоночная СУБД типа ключ-значение, работающая поверх HDFS.

Модель данных HBase унаследована от BigTable (Google), а именно:

- Данные организованы в таблицы.
- Таблицы имеют один и только один индекс — ключ строки (row key). Записи фактически хранятся отсортированными по ключу.
- В схеме таблицы задаются не отдельные колонки (атрибуты), а группы колонок (column families):
    - Каждая группа состоит из произвольного числа колонок.
    - Для каждого атрибута может храниться несколько различных версий.
    - 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.
    - На уровне группы колонок задаются такие параметры как time to live (TTL) и максимальное количество хранимых версий.
    - Запросы при чтении таблицы позволяют обратиться только к части групп колонок, не читая таблицу целиком.

Модель данных HBase можно запомнить как соответствие значения ключу:

`<Table, Row Key, Column Family, Column, Version Timestamp>` → `Value`

Ограничения:

- Операторы JOIN не поддерживаются.
- Три типа lookup'ов: по ключу, полное сканирование таблиц и обращение к партиции.
- Ограниченная поддержка атомарности и транзакций: возможны пакетные преобразования над одной строкой.
- Отсутствует SQL-интерфейс. Доступ к данным осуществляется через Java, REST, Thrift APIs.

## 30. Какие средства для распределенной обработки данных на языке R вы знаете?

Distributed R, Apache Spark, RHadoop.

## 31. Основные этапы извлечения информации из текстов.

- Выбор источника и **подготовка корпуса** — набора документов для извлечения информации.
- **Распознавание сущностей**: выделение сущностей в корпусе.
  Методы: по словарю или по правилам (регулярными выражениями).
- **Аннотирование сущностей**: категоризация найденных сущностей, т.е. сопоставление каждой сущности класса из набора предопределенных.
- **Разрешение неоднозначностей**: если сущность неоднозначна, необходимо уточнить ее значение.
  Методы: по контексту, по количеству вхождений в корпус или основываясь на связях между сущностями (сущности из одного домена чаще появляются в одном документе вместе).

## 32. Подходы к распознаванию сущностей из текстов.

- По **словарям**.
    - Алгоритмически реализуется через префиксные деревья, сложность поиска составляет O(textLength × maxWordLength).
    - Недостатки:
        - Словари должны быть предварительно составлены.
        - Необходимо поддерживать словари актуальными. Для активно меняющихся наборов данных это становится серьезным препятствием для использования словарей.
        - Плохо работают с вариациями имен (например, спряжениями).
        - Не работают с неизвестным набором сущностей.
- На основе **правил**.
    - Обычно используются регулярные выражения. Регулярные выражения реализуются через конечные автоматы.

## 33. Язык AQL. Компоненты языка AQL.

Annotation Query Language входит в состав InfoSphere® BigInsights™ Text Analytics. AQL —декларативный SQL-like язык, предназначенный для разработки экстракторов для текстовой аналитики.

Компоненты:

- `CREATE VIEW` — создает представление и определяет входящие в его состав кортежи.
- `EXTRACT` — определяет правила извлечения информации из текста.
- `SELECT` — средство для формирования составных паттернов из простых блоков.
- Detag — пре-процессинг по удалению HTML- и XML-тегов из текста.
- Словари (см. билет по подходам к распознаванию сущностей).
- Встроенные функции — высокоуровневые функции для использования в правилах извлечения (extraction rules).
- Определенные пользователем функции для извлечения.

Доп. вопросы

- **Какие существуют архитектуры распределенных систем, кроме SNA? В чем особенность Hadoop SNA?**
  Билет про SNA.
  SNA подразумевает, что вычислительные узлы не делят никакой ресурс. Значит, альтернатива такому подходу — делить какой-либо ресурс. Например, можно иметь общий диск. В такой архитектуре данные рассылаются к месту обработки (вычислительным узлам).
  В Hadoop же наоборот: код рассылается к месту хранения данных для запуска mapper'ов на тех машинах, где находятся данные, чтобы избежать лишней передачи данных по сети.
- **Отличия Spark от MapReduce.**
  Помимо возможностей, указанных в билете, лектор акцентирует внимание на возможности итерационно выполнять одну и ту же операцию над одними и теми же данными. Это дает возможность эффективно использовать Spark для алгоритмов машинного обучения.