Введение в экосистему Hadoop

Максим Губин

Томск

econophysica

На этой лекции будут рассмотрены

- Hadoop
- MapReduce
- HDFS
- Avro
- Pig
- Spark
- MLLIB

Что такое Hadoop

- Hadoop - это программная среда с открытым исходным кодом, используемая для распределенного хранения и обработки наборов данных больших данных с использованием модели программирования MapReduce.
- Он состоит из компьютерных кластеров, построенных из оборудования потребительского уровня.
- Все модули в Hadoop разработаны с фундаментальным предположением, что аппаратные сбои являются обычным явлением и должны автоматически обрабатываться платформой.

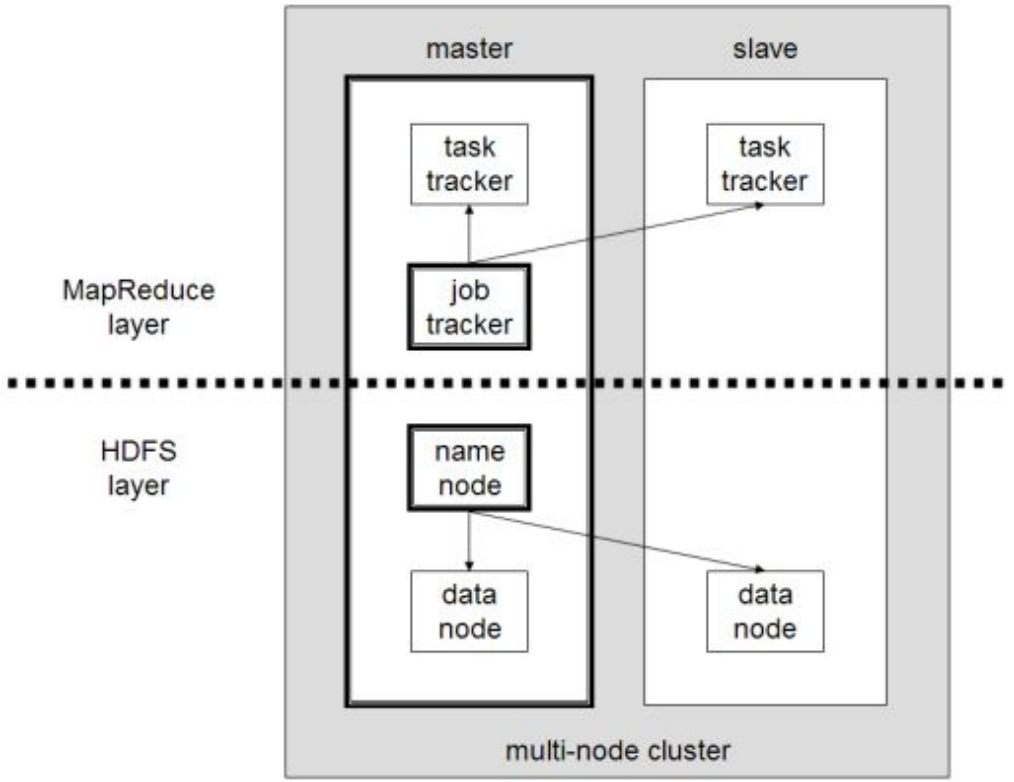
История развития Hadoop

- Статья "Файловая система Google", опубликована в октябре 2003 года;
- Hadoop 0.1.0 был выпущен в апреле 2006 года;
- HBase и Pig были созданы в октябре 2007 года;
- В 2009 году Hadoop был использован для обработки петабайта данных;
- Выпуск Hadoop 1.0: ноябрь 2012 г .;
- Apache Spark был готов к производству в 2014 году;
- Hadoop 3.0 был выпущен в 2017 году.
- Развитие Hadoop быстро продолжается и сейчас.

Зачем нужен Hadoop

- Гибкость для хранения и обработки данных любого типа, будь то структурированные, полуструктурированные или неструктурированные. Он не ограничен одной схемой.
- Преимущество при обработке данных сложного характера. Его масштабируемая архитектура распределяет рабочие нагрузки по многим узлам.
- Еще одним дополнительным преимуществом является то, что его гибкая файловая система устраняет узкие места ETL.
- Экономно масштабируется, может использоваться на обычном оборудовании. Помимо этого его природа с открытым исходным кодом защищает от vendor lock.

Архитектура Hadoop



- Hadoop работает по системе master-slave. Есть главный узел и есть подчиненные узлы.
- Главный узел управляет, поддерживает и контролирует подчиненных, тогда как подчиненные выполняют вычислительные задачи.
- В архитектуре Hadoop Master должен развертываться на хорошем серверном оборудовании, а не на обычном оборудовании.

Hadoop Distributed File System

- Основная система хранения Hadoop.
- Файловая система на основе Java;
- Масштабируемая;
- Отказоустойчивая;
- Надежная и экономичная.

Другие поддерживаемые файловые системы

- FTP file system;
- Amazon S3 (Simple Storage Service) file system;
- Windows Azure Storage Blobs (WASB) file system
- IBM General Parallel File System;
- Parascale file system;
- CloudIQ Storage;
- IBRIX Fusion;
- MapR FS

econophysica

Hadoop Daemons

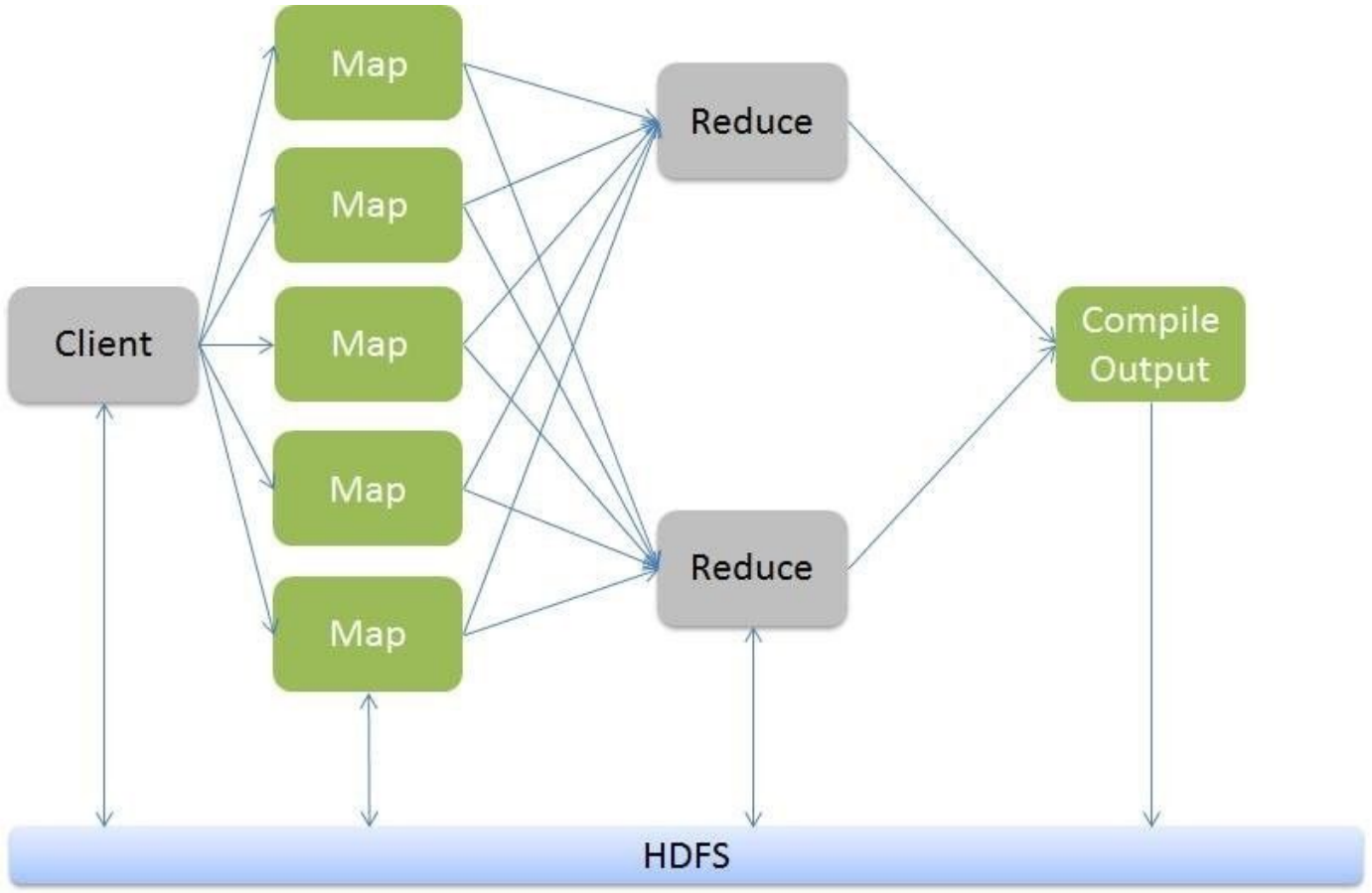
- Namenode - работает на главном узле для HDFS.
- DataNode - работает на подчиненных узлах для HDFS.
- ResourceManager - работает на главном узле для Yarn.
- NodeManager - работает на подчиненном узле для Yarn.

Эти 4 процесса необходимы для того, чтобы экземпляр Hadoop работал.

MapReduce

https://www.joelonsoftware.com/2006/08/01/can-your-programming-language-do-this/

Hadoop MapReduce



econophysica

Hadoop MapReduce

- JobTracker:
  - Принимает задания от клиента;
  - Передает задания на доступные узлы TaskTracker в кластере;
- TaskTracker:
- Запускает задания и отправляет обновления статуса в JobTracker.

По умолчанию Hadoop использует планировщик FIFO, но его можно настроить для других планировщиков, таких как fair планировщик, разработанный Facebook.

Hadoop MapReduce

- Простота - задания MapReduce легко выполнять. Приложения могут быть написаны на любом языке, например, Java, C ++ и Python.
- Масштабируемость - MapReduce может обрабатывать петабайты данных.
- Скорость - с помощью параллельной обработки, решение задач, на которые обычно уходят дни, выполняется за часы или минуты с помощью MapReduce.

  Отказоустойчивость - MapReduce устойчива к сбоям. Если одна копия данных недоступна, на другой машине будет копия данных, которую можно использовать для решения подзадачи.

Yet Another Resource Negotiator

- Технология управления ресурсами с открытым исходным кодом, которая развернута в кластере Hadoop.
- YARN стремится эффективно распределять ресурсы между различными приложениями.
- Он запускает два процесса, которые выполняют две разные задачи: отслеживание заданий и мониторинг прогресса.
- Эти два демона называются **resource manager** и **application master** соответственно.
- Диспетчер ресурсов распределяет ресурсы между различными приложениями, а мастер приложений контролирует выполнение процессов.

YARN

- Гибкость - включает другие специализированные модели обработки данных, помимо MapReduce (пакетные), такие как интерактивная и потоковая. Благодаря этой функции YARN, другие приложения также могут запускаться вместе с программами Map Reduce в Hadoop2.
- Эффективность. Поскольку многие приложения работают в одном кластере, эффективность Hadoop увеличивается без особого влияния на качество обслуживания.
- Shared - обеспечивает стабильную, надежную, безопасную основу и разделяемые операционные сервисы для нескольких рабочих нагрузок. Дополнительные модели программирования, такие как обработка графов и итеративное моделирование, теперь возможны для обработки данных с помощью YARN.

Hive

Apache Hive - это система организации хранилища данных с открытым исходным кодом для запроса и анализа больших наборов данных, хранящихся в файлах Hadoop.

Выполняет три основные функции:

- обобщение данных,
- выполнение запроса,
- анализ.

Hive использует язык запросов HiveQL (HQL), который похож на SQL. HiveQL автоматически переводит SQL-подобные запросы в задания MapReduce, которые будут выполняться в Hadoop.

HBase

- Распределенная база данных, предназначенная для хранения структурированных данных в таблицах, которые могут содержать миллиарды строк и миллионы столбцов.
- HBase - это масштабируемая, распределенная база данных No-sql, построенная на основе HDFS.
- HBase Master: согласовывает распределение нагрузки по всем RegionServer.
- RegionServer: обрабатывает запросы на чтение, запись, обновление и удаление от клиентов, работает на узлах данных.

Avro

- Avro - это проект с открытым исходным кодом, который предоставляет сериализацию данных и услуги обмена данными для Hadoop. Эти услуги могут использоваться вместе или независимо. Большие данные могут обмениваться программами, написанными на разных языках, используя Avro.
  Он опирается на схемы для сериализации / десериализации.
  Когда данные Avro хранятся в файле, его схема сохраняется вместе с ним, так что файлы могут быть обработаны позже любой программой.

Другие форматы данных

Parquet, JSON, CSV поддерживаются "из коробки", есть множество плагинов для различных других форматов данных.

Sqoop

Sqoop импортирует данные из внешних источников в связанные компоненты экосистемы Hadoop, такие как HDFS, Hbase или Hive. Он также экспортирует данные из Hadoop в другие внешние источники. Sqoop работает с реляционными базами данных, такими как teradata, Netezza, oracle, MySQL.

- Импортирует последовательные наборы данных с мэйнфреймов.
- Импортирует напрямую в файлы ORC. Улучшает сжатие и облегчает индексирование, а также повышает производительность запросов.
- Параллельная передача данных - для более быстрой работы и оптимального использования системы.
- Эффективный анализ данных - Повышает эффективность анализа данных путем объединения структурированных данных и неструктурированных данных в схеме чтения озера данных.
- Быстрое копирование данных - из внешней системы в Hadoop.

Zookeeper

Apache Zookeeper - это централизованная служба и компонент экосистемы Hadoop для поддержки информации о конфигурации, наименованиb, предоставлении и распределенной синхронизации групповых политик. Zookeeper координирует большой кластер машин.

Zookeeper работает лучше всего с рабочими нагрузками, где чтение данных происходит чаще, чем запись. Идеальное соотношение чтения / записи составляет 10: 1.

Упорядоченно - Zookeeper ведет учет всех транзакций.

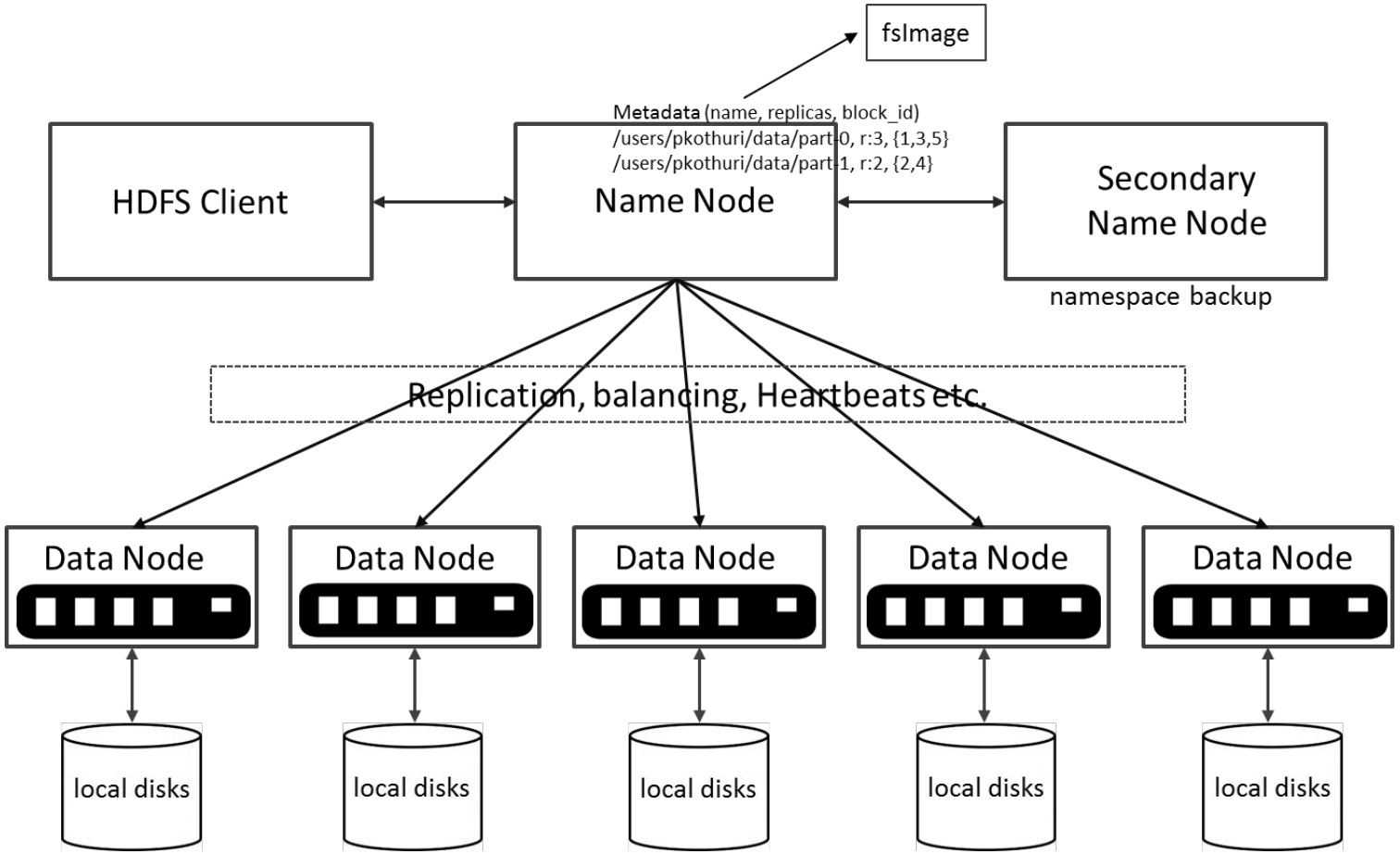
HDFS: Hadoop distributed file system

- HDFS - это распределенная файловая система, отказоустойчивая, масштабируемая и чрезвычайно легко расширяемая.
- HDFS является основным распределенным хранилищем для приложений Hadoop.
- HDFS предоставляет интерфейсы для приложений, чтобы перемещаться ближе к данным.
- HDFS разработана, чтобы «просто работать», однако практические знания помогают в диагностике и улучшениях.

HDFS: структура

- NameNode: сердце файловой системы HDFS, он поддерживает и управляет метаданными файловой системы. Например; какие блоки составляют файл и на каких DataNode эти блоки хранятся.
- DataNode: здесь HDFS хранит фактические данные, обычно этих узлов довольно много.

HDFS: структура



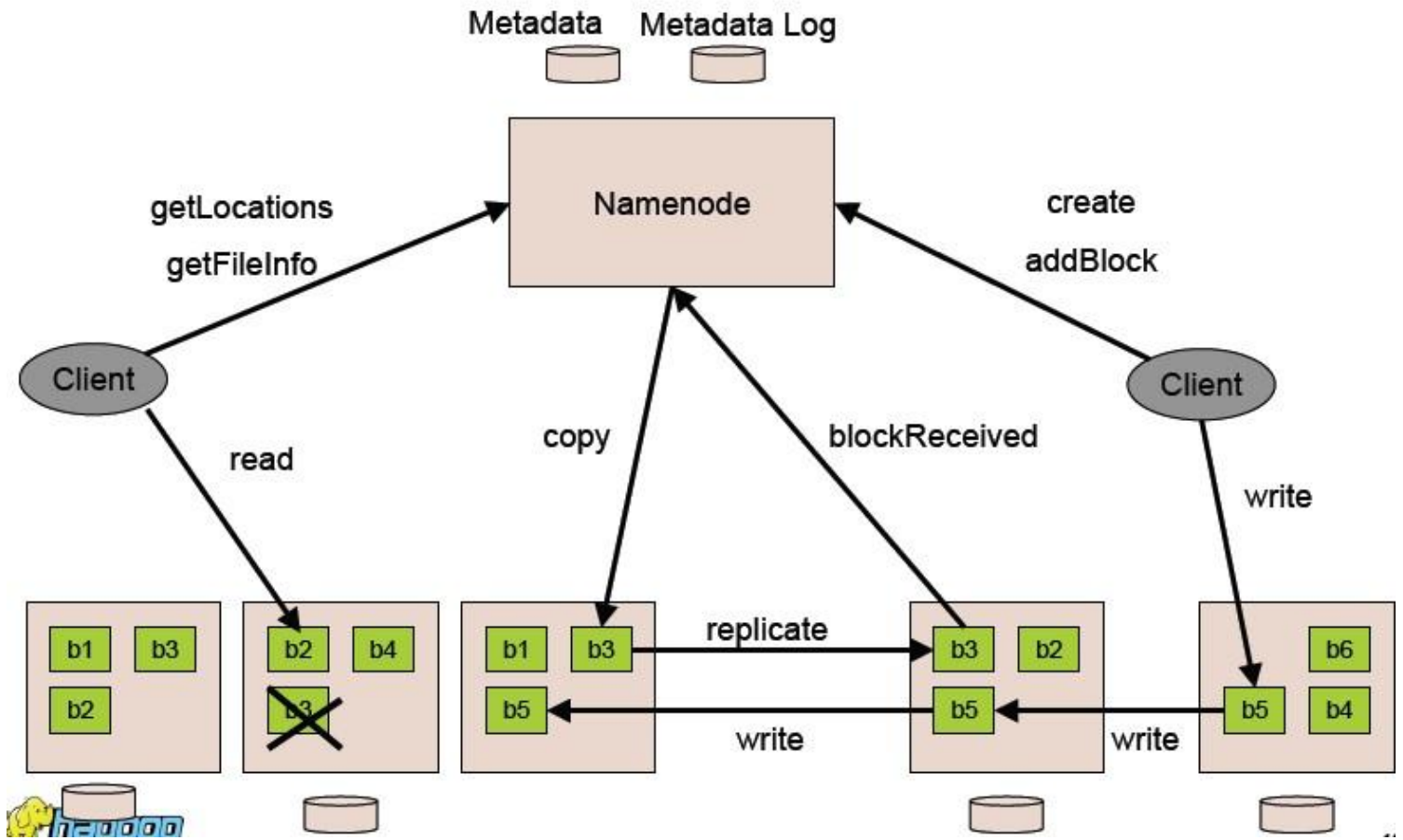
HDFS: особенности

- Отказоустойчивость - данные дублируются на нескольких узлах данных для защиты от сбоев компьютера. По умолчанию коэффициент репликации равен 3 (каждый блок хранится на трех машинах);
- Масштабируемость - передача данных происходит непосредственно с узлами данных, поэтому ваша емкость чтения / записи достаточно хорошо масштабируется с количеством узлов данных;
- Объём - нужно больше места на диске? Просто добавьте больше узлов данных и перебалансируйте;
- Промышленный стандарт - Другие распределенные приложения построены на основе HDFS (HBase, Map-Reduce)
- HDFS предназначена для обработки больших наборов данных с семантикой однократная запись-чтение, она не предназначена для доступа с низкой задержкой.

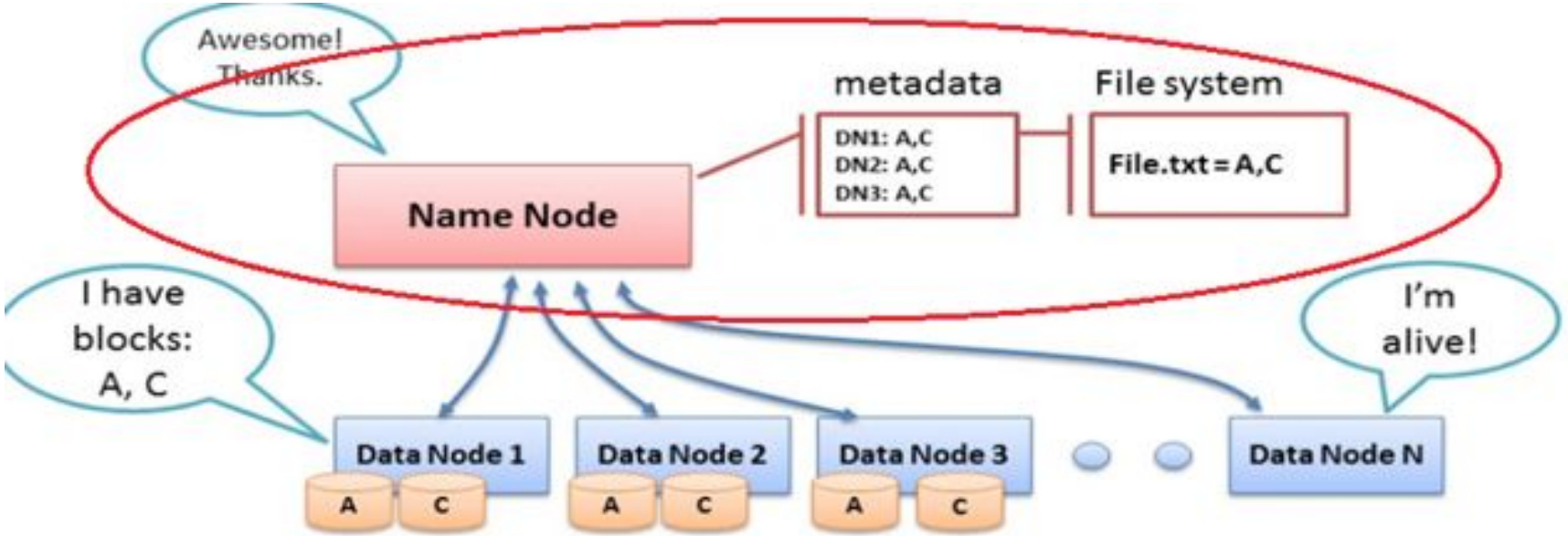
HDFS: Хранение данных

- Каждый файл, записанный в HDFS, разбивается на блоки данных
- Каждый блок хранится на одном или нескольких узлах
- Каждая копия блока называется репликой
- Политика размещения блоков
  - Первая реплика размещается на локальном узле
  - Вторая реплика находится в другой стойке
  - Третья реплика находится в той же стойке, что и вторая реплика

HDFS: Операции чтения и записи

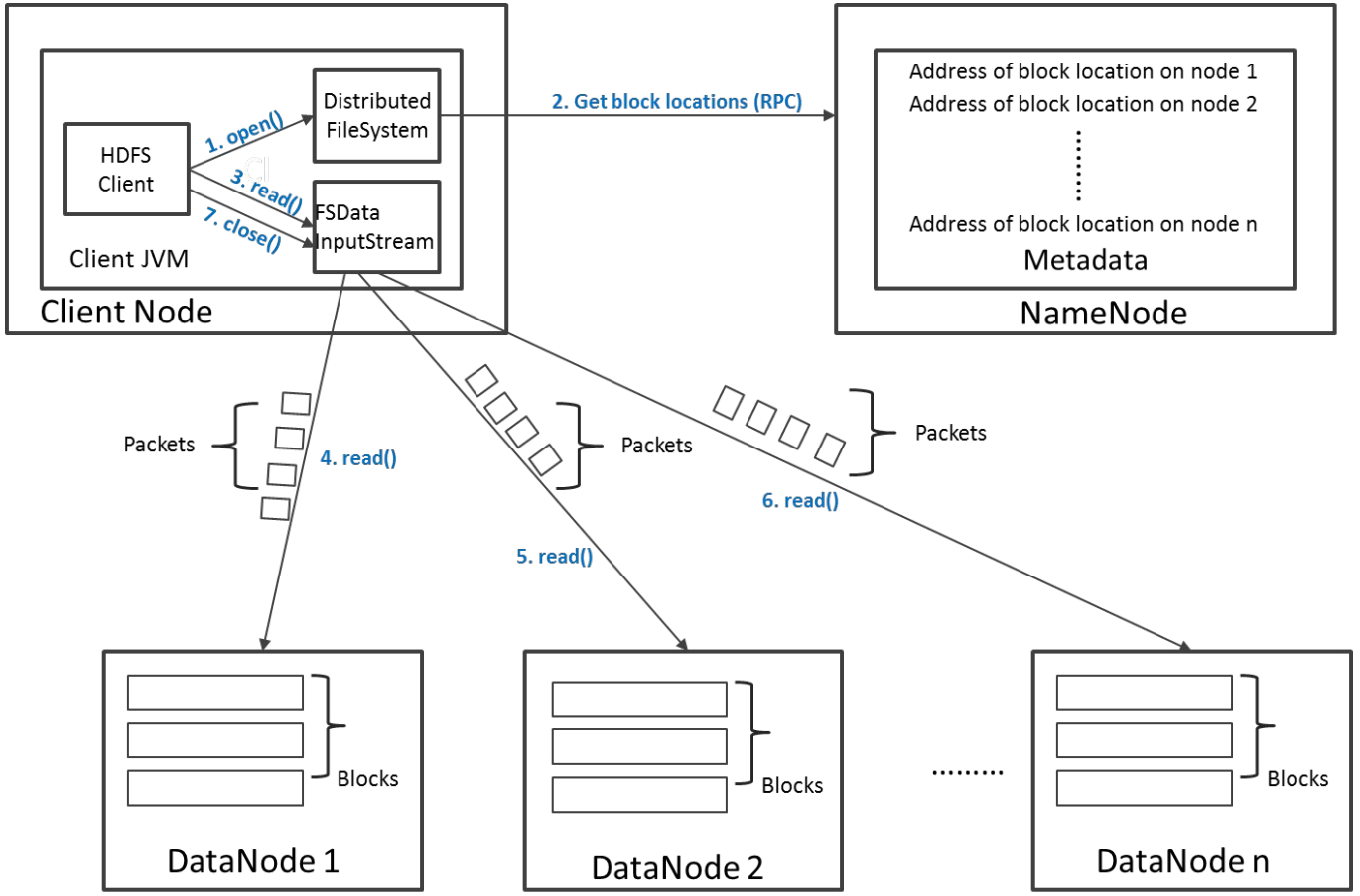


HDFS: Heartbeats

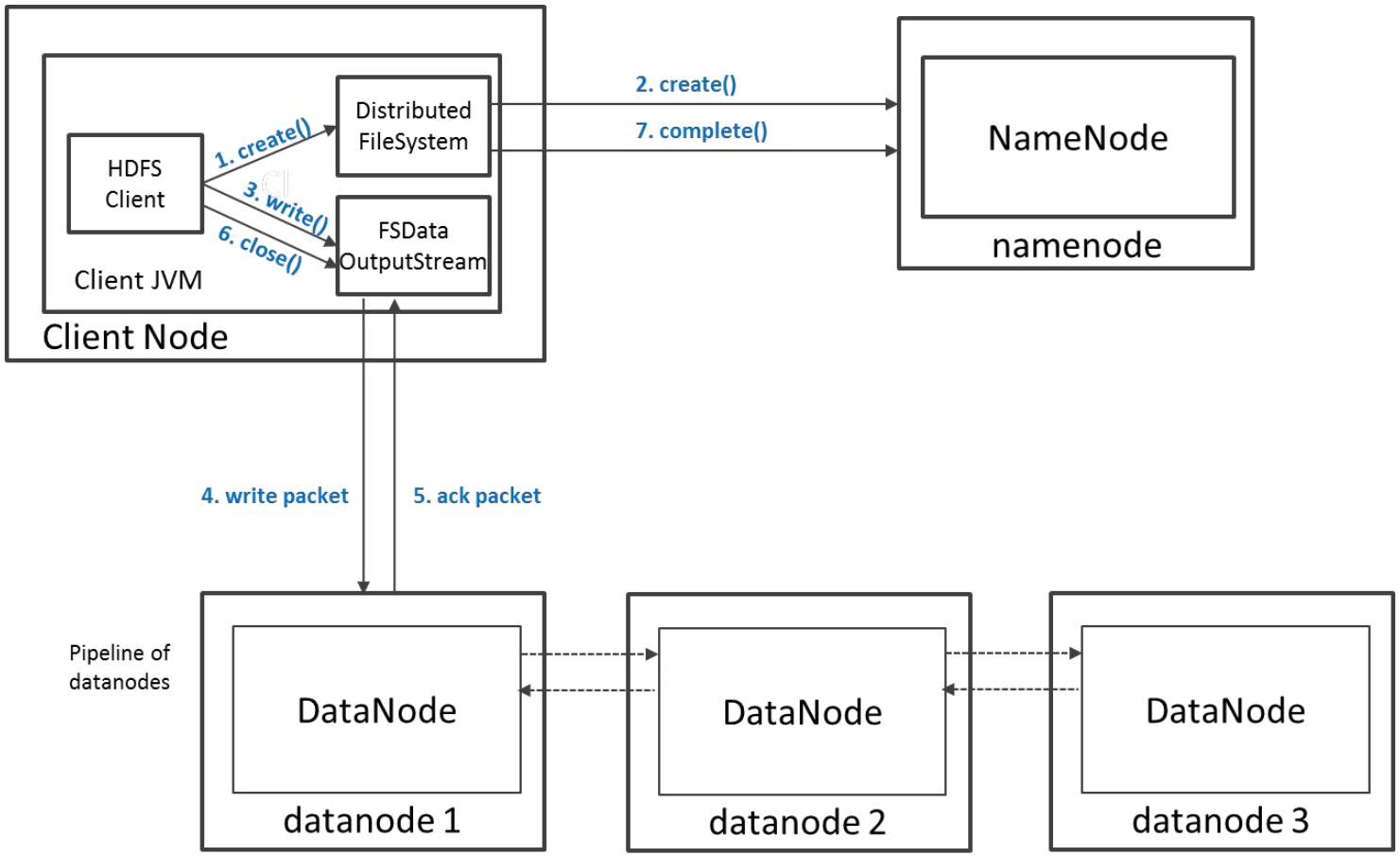


- Data Node sends Heartbeats
- Every 10th heartbeat is a Block report
- Name Node builds metadata from Block reports
- TCP – every 3 seconds

HDFS: Чтение данных



HDFS: Запись данных



econophysica

HDFS: права доступа к данным

- Аутентификация в Hadoop
  - Простой - небезопасный способ использования имени пользователя ОС для аутентификации в hadoop;
  - Kerberos - аутентификация с использованием токенов Kerberos;
    - Устанавливается с помощью hadoop.security.authentication = simple | kerberos
- Права доступа к файлам и каталогам такие же, как в POSIX
  - разрешения на чтение (r), запись (w) и выполнение (x)
  - также есть владелец, группа и режим
  - включена по умолчанию (dfs.permissions.enabled = true)
- ACL используются для разрешений реализации, которые отличаются от естественной иерархии пользователей и групп
  - включается dfs.namenode.acls.enabled = true

HDFS: Доступ

- Java API (DistributedFileSystem)
- Обертка на C (libhdfs)
- Протокол HTTP
- Протокол WebDAV
- Командная строка

Командная строка является одним из самых простых и знакомых способов.

HDFS: Командная строка

- Пользовательские Команды
  - **hdfs dfs** - запускает команды файловой системы на HDFS
  - **hdfs fsck** - запускает команду проверки файловой системы HDFS
- Команды администрирования
  - **hdfs dfsadmin** - запускает команды администрирования HDFS

HDFS: Команды

- Показать содержимое директории

```
hdfs dfs -ls
hdfs dfs -ls /
hdfs dfs -ls -R /var
```

- Показать использование места на диске

```
hdfs dfs -du -h /
hdfs dfs -du /hbase/data/hbase/namespace/
hdfs dfs -du -h /hbase/data/hbase/namespace/
hdfs dfs -du -s /hbase/data/hbase/namespace/
```

HDFS: Команды

- Перенести данные в HDFS

```
hdfs dfs -mkdir mydata
hdfs dfs -ls
hdfs dfs -copyFromLocal data/somefile.avro mydata
hdfs dfs -ls -R
```

- Перенести данные обратно в локальную файловую систему

```
cd data/
hdfs dfs -copyToLocal mydata/somefile.avro data.avro
md5sum somefile.avro data.avro
```

HDFS: Команды

- Посмотреть acl для файла

```
hdfs dfs -getfacl mydata/somefile.avro
```

- Посмотреть статистику для файла (%r – фактор репликации)

```
hdfs dfs -stat "%r" mydata/somefile.avro
```

- Писать в HDFS из stdin

```
echo “foo bar" | hdfs dfs -put - testdata/myfile.txt
hdfs dfs -ls -R
hdfs dfs -cat testdata/myfile.txt
```

HDFS: Команды (fsck)

- Удалить файл

```
hdfs dfs -rm mydata/somefile.avro
hdfs dfs -ls -R
```

- Посмотреть расположение блоков файла

```
hdfs fsck mydata/somefile.avro -files -blocks
-locations
```

- Посмотреть список отсутствующих блоков и файлов, которым они принадлежат

```
hdfs fsck / -list-corruptfileblocks
```

HDFS: Команды администрирования

- Запросить отчет об состоянии кластера

```
hdfs dfsadmin -report
```

- Дерево стоек и узлов в них

```
hdfs dfsadmin -printTopology
```

- Получить статус определенного узла данных

```
hdfs dfsadmin -getDatanodeInfo
localhost:50020
```

HDFS: Команды

- Получить список namenode кластера

```
hdfs getconf -namenodes
```

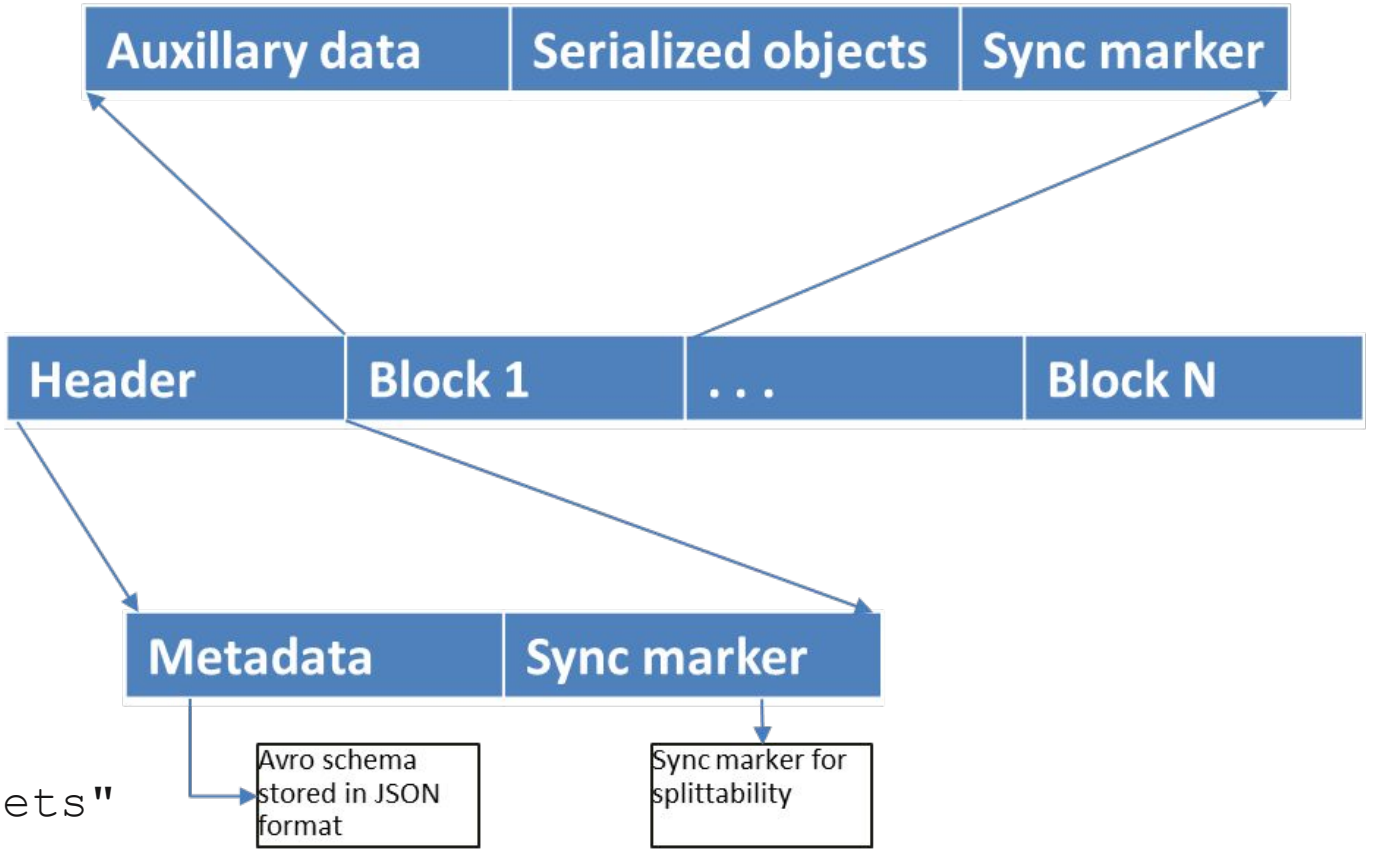
- Сбросить образ файловой системы HDFS в XML-файл

```
cd /var/lib/hadoop-hdfs/cache/hdfs/dfs/name/current
hdfs oiv -i fsimage_0000000000000003388 -o
/tmp/fsimage.xml -p XML
```

Avro

Пример схемы AVRO в формате JSON

```
{
  "type" : "record",
  "name" : "tweets",
  "fields" : [ {
    "name" : "username",
    "type" : "string",
}, {
    "name" : "tweet",
    "type" : "string",
}, {
    "name" : "timestamp",
    "type" : "long",
} ],
  "doc:" : "schema for storing tweets"
}
```



econophysica

Parquet

- колоночный формат хранения
- Ключевым преимуществом является хранение вложенных данных в действительно столбчатом формате с использованием уровней определения и повторения

Nested schema

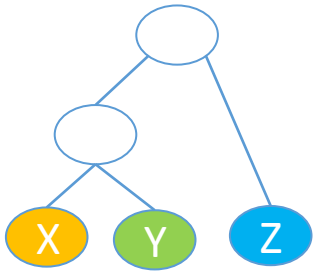
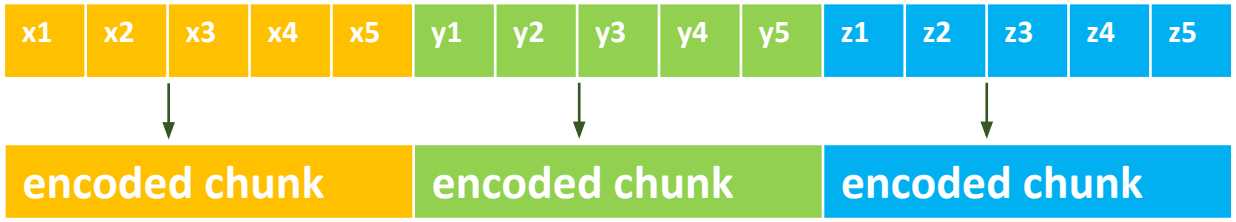


Table representation

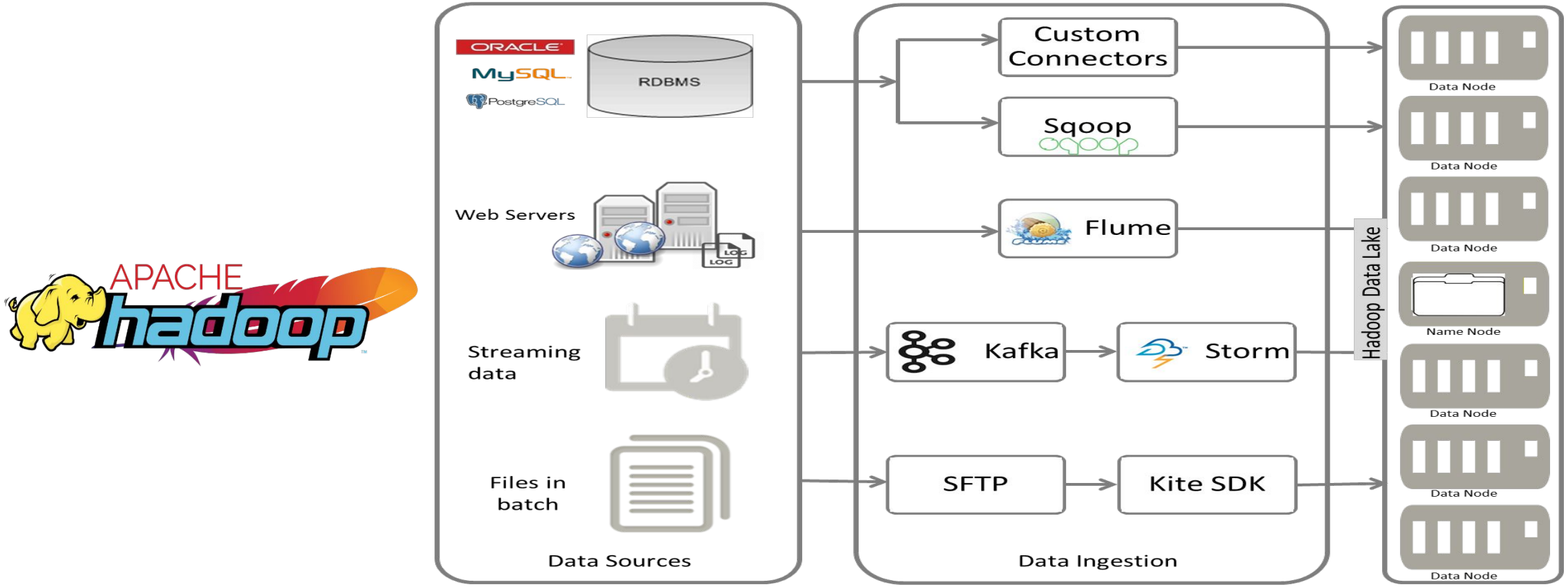
X	Y	Z
x1	y1	z1
x2	y2	z2
x3	y3	z3
x4	y4	z4
x5	Y5	z5

Row format

Columnar format



Способы внесения данных в Hadoop



econophysica

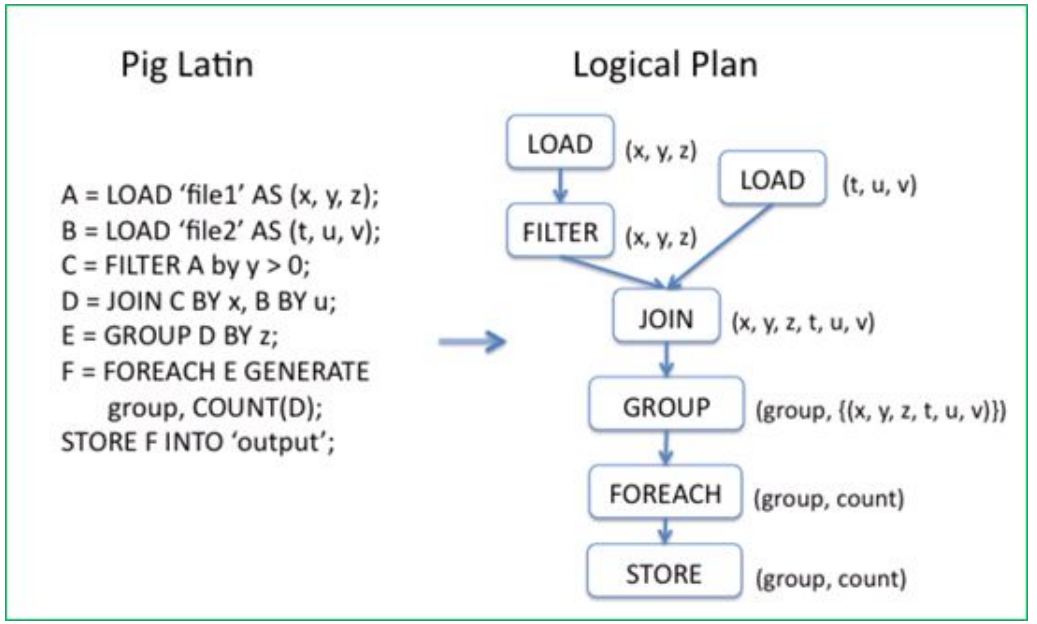
Sqoop

- Sqoop сокращает количество запросов для осуществления импорта и экспорта с помощью MapReduce
- Sqoop всегда требует драйвер JDBC
- Для Sqoop требуются драйверы JDBC для конкретного сервера базы данных, их следует скопировать в / usr / lib / sqoop / lib
- Пример запроса:

```
Sqoop TOOL PROPERTY_ARGS SQOOP_ARGS
```

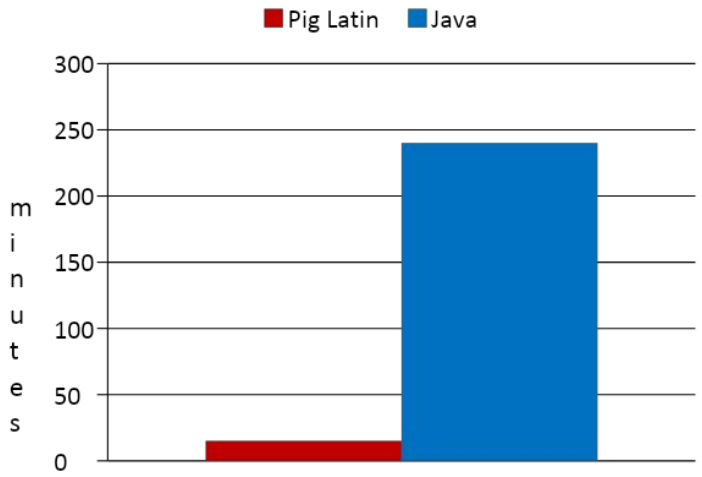
Что такое Pig?

- Среда для анализа больших неструктурированных и полуструктурированных данных поверх Hadoop.
- Pig Engine анализирует, компилирует скрипты Pig Latin в задания MapReduce, запускаемые поверх Hadoop.
- Pig Latin - декларативный SQL-подобный язык; языковой интерфейс высокого уровня для Hadoop.



econophysica

Мотивация для использования Pig



- Ускорение разработки
- Меньше строк кода (написание MapReduce, как написание SQL-запросов)
- Повторно используйте код (Pig library, Piggy bank)
- Один тест: найдите 5 слов с наиболее высокой частотой повторения:
  - 10 строк Pig Latin V.S 200 строк на Java
  - 15 минут на Pig Latin V.S 4 часа на Яве

Посчет количества слов на MapReduce

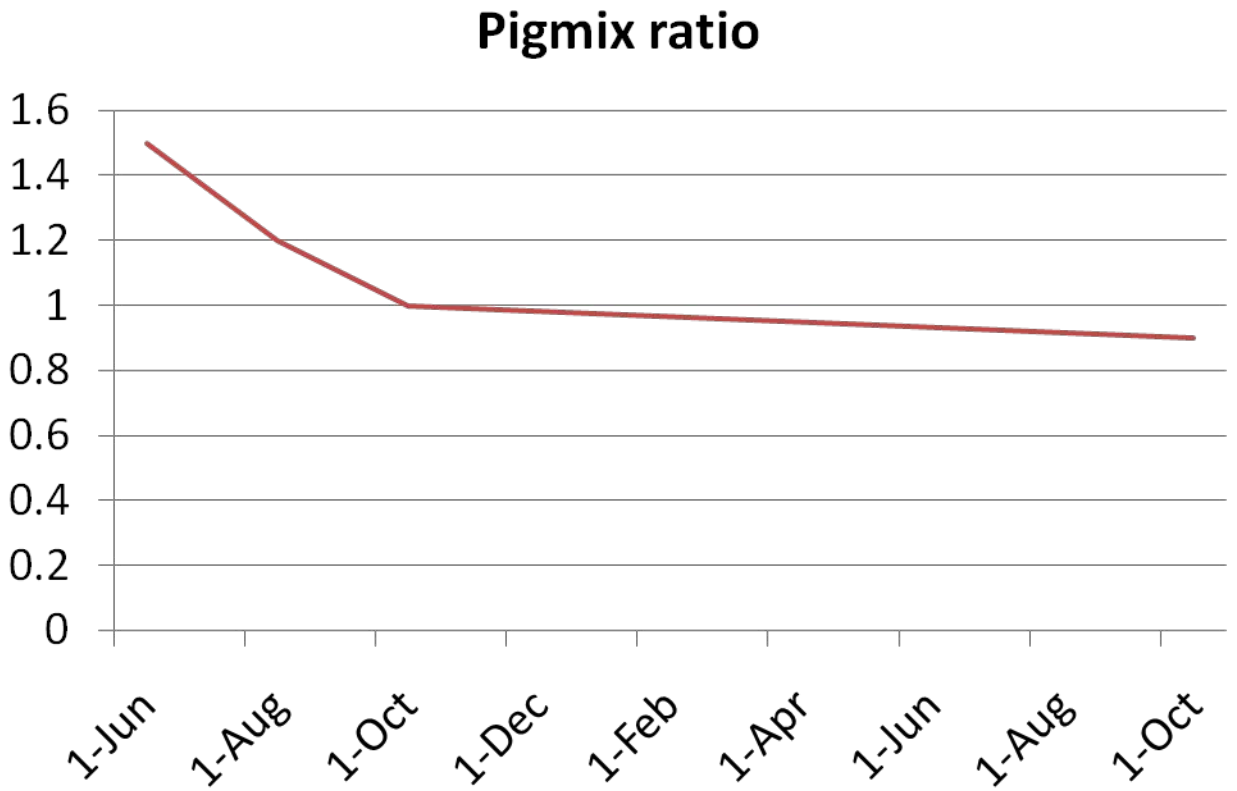
```
import java.io.IOException;
import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;
import java.util.StringTokenizer;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.conf.Configured;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapred.FileInputFormat;
import org.apache.hadoop.mapred.FileOutputFormat;
import org.apache.hadoop.mapred.JobClient;
import org.apache.hadoop.mapred.JobConf;
import org.apache.hadoop.mapred.MapReduceBase;
import org.apache.hadoop.mapred.Mapper;
import org.apache.hadoop.mapred.OutputCollector;
import org.apache.hadoop.mapred.Reducer;
import org.apache.hadoop.mapred.Reporter;
import org.apache.hadoop.util.Tool;
import org.apache.hadoop.util.ToolRunner;

public class WordCount extends Configured implements Tool {

  public static class MapClass extends MapReduceBase
    implements Mapper<LongWritable, Text, Text, IntWritable> {

    private final static IntWritable one = new IntWritable(1);
    private Text word = new Text();

    public void map(LongWritable key, Text value,
                    OutputCollector<Text, IntWritable> output,
                    Reporter reporter) throws IOException {
      String line = value.toString();
      StringTokenizer itr = new StringTokenizer(line);
      while (itr.hasMoreTokens()) {
        word.set(itr.nextToken());
        output.collect(word, one);
      }
    }
  }

  public static class Reduce extends MapReduceBase
    implements Reducer<Text, IntWritable, Text, IntWritable> {

    public void reduce(Text key, Iterator<IntWritable> values,
                       OutputCollector<Text, IntWritable> output,
                       Reporter reporter) throws IOException {
      int sum = 0;
      while (values.hasNext()) {
        sum += values.next().get();
      }
      output.collect(key, new IntWritable(sum));
    }
  }
```

```
  public int run(String[] args) throws Exception {
    JobConf conf = new JobConf(getConf(), WordCount.class);
    conf.setJobName("wordcount");

    conf.setOutputKeyClass(Text.class);
    conf.setOutputValueClass(IntWritable.class);

    conf.setMapperClass(MapClass.class);
    conf.setCombinerClass(Reduce.class);
    conf.setReducerClass(Reduce.class);

    List<String> other_args = new ArrayList<String>();
    for(int i=0; i < args.length; ++i) {
      try {
        if ("-m".equals(args[i])) {
          conf.setNumMapTasks(Integer.parseInt(args[++i]));
        } else if ("-r".equals(args[i])) {
          conf.setNumReduceTasks(Integer.parseInt(args[++i]));
        } else {
          other_args.add(args[i]);
        }
      } catch (NumberFormatException except) {
        System.out.println("ERROR: Integer expected instead of " + args[i]);
        return printUsage();
      } catch (ArrayIndexOutOfBoundsException except) {
        System.out.println("ERROR: Required parameter missing from " +
                           args[i-1]);
        return printUsage();
      }
    }
    // Make sure there are exactly 2 parameters left.
    if (other_args.size() != 2) {
      System.out.println("ERROR: Wrong number of parameters: " +
                         other_args.size() + " instead of 2.");
      return printUsage();
    }
    FileInputFormat.setInputPaths(conf, other_args.get(0));
    FileOutputFormat.setOutputPath(conf, new Path(other_args.get(1)));
    JobClient.runJob(conf);
    return 0;
  }

  public static void main(String[] args) throws Exception {
    int res = ToolRunner.run(new Configuration(), new WordCount(), args);
    System.exit(res);
  }
}
```

То же самое на Pig

```
Lines=LOAD ‘input/hadoop.log’ AS (line: chararray);
Words = FOREACH Lines GENERATE  FLATTEN(TOKENIZE(line)) AS word;
Groups = GROUP Words BY word;
Counts = FOREACH Groups GENERATE  group, COUNT(Words);
Results = ORDER Words BY Counts DESC;
Top5 = LIMIT Results 5;
STORE Top5 INTO /output/top5words;
```

Производительность Pig по сравнению с чистым MapReduce



econophysica

Особенности Pig

- UDF могут быть написаны, чтобы воспользоваться преимуществом combiner;
- Четыре реализации join
- Написание функций загрузки и сохранения элементарно, если существуют InputFormat и OutputFormat
- Multi-query: pig объединит определенные типы операций в один конвейер, чтобы уменьшить количество обрабатываемых данных.
- Order By обеспечивает сбалансированное упорядочение по reduce-задачам
- Piggybank, коллекция пользовательских UDF

Для чего используют Pig

- 70% вычислений в Yahoo (10ks в день)
- Twitter, LinkedIn, Ebay, AOL,… используют Pig чтобы:
  - Обрабатывать веб-журналы
  - Создавать модели поведения пользователей
  - Обрабатывать изображения
  - Построить карты Интернета
  - Проводить исследования на больших наборах данных

Доступ к Pig

Варианты:

- Пакетный режим: отправить скрипт напрямую
- Интерактивный режим: Grunt, командная строка
- Java-класс PigServer, JDBC-подобный интерфейс

Режимы исполнения:

- Локальный режим: pig –x local
- Режим Mapreduce: pig –x mapreduce

Типы данных Pig

- Скалярные типы:
  - Int, long, float, double, boolean, null, chararray, bytearry;
- Сложные типы: field, tuple, bag, relation;
- field - это поле данных
- tuple - это упорядоченный набор полей
- bag - это коллекция кортежей
- Отношение это bag

Примеры:
Tuple - строка в базе данных
(0002576169, Том, 20, 4,0)
Сумка это Таблица или представление в базе данных
{(0002576169, Том, 20, 4,0),
 (0002576170, Mike, 20, 3.6),
 (0002576171 Люси, 19, 4,0),.... }

Операции Pig

- Загрузка данных
  - LOAD loads input data
  - Lines=LOAD ‘input/access.log’ AS (line: chararray);
- Проекция
  - FOREACH … GENERTE … (similar to SELECT)
  - Применяет набор преобразований к каждой записи.
- Группировка
  - GROUP группирует записи с одинаковым ключом
- Dump/Store
  - DUMP выводит результат на экран, STORE сохраняет на диск
- Аггрегации
  - AVG, COUNT, MAX, MIN, SUM

Загрузка данных в Pig

```
students = load 'student.txt' using PigStorage('\t')
                        as (studentid: int, name:chararray, age:int, gpa:double);
```

- PigStorage: загружает / сохраняет отношения, используя текстовый формат с разделителями полей
- TextLoader: загружает отношения из простого текстового формата
- BinStorage: загружает / сохраняет отношения из или в двоичные файлы
- PigDump: хранит отношения, записывая представление toString () кортежей, по одному на строку

Pig: FOREACH

```
studentid = FOREACH  students  GENERATE  studentid, name;
```

- The Foreach … перебирает элементы bag и преобразует их.
- В результате Foreach тоже получается bag
- Элементы называются так же, как во входном bag

Позиционная нотация в Pig

```
students = LOAD 'student.txt' USING PigStorage() AS (name:chararray, age:int, gpa:float);
DUMP A;
(John,18,4.0F)
(Mary,19,3.8F)
(Bill,20,3.9F)
studentname = Foreach students Generate $1 as studentname;
```

- Поля можно адресовать по их позиции.

Pig: Group

```
B = GROUP A BY age;
C = COGROUP A BY name, B BY name;
```

- Группирует данные в одно или несколько отношений
- Операторы GROUP и COGROUP идентичны.
- Оба оператора работают с одним или несколькими отношениями.
- Для удобства чтения GROUP используется в выражениях, включающих одно отношение
- COGROUP используется в утверждениях, включающих два или более отношений.

Pig: Dump&Store

```
A = LOAD ‘input/pig/multiquery/A’;
B = FILTER A by $1 == “apple”;
C = FILTER A by $1 == “apple”;
SOTRE B INTO “output/b”
STORE C INTO “output/c”
```

- Pig способен распознать, что B и C оба основаны на A, и сгенерировать A только один раз;
- Dump: выводит на экран;
- Store: сохраняет на диск.

Pig: Count

```
X = FOREACH B GENERATE COUNT(A);
```

- Вычислить количество элементов в сумке
- Используйте функцию COUNT для вычисления количества элементов в сумке.
- Для COUNT требуется предыдущий оператор GROUP ALL для глобальных счетчиков и оператор GROUP BY для счетчиков групп.

Pig: Order

```
student = ORDER students BY gpa DESC;
```

- Сортирует отношение на основе одного или нескольких полей
- В Pig отношения неупорядочены.
- Если вы сортируете отношение A для получения отношения X, отношения A и X по-прежнему содержат одинаковые элементы.

Запуск сценариев Pig

- Локальный режим
  - Используется локальный хост и локальная файловая система
  - Ни Hadoop, ни HDFS не требуются
  - Полезно для прототипирования и отладки
- Режим MapReduce
  - Запуск на кластере Hadoop и HDFS
- Пакетный режим - запустить скрипт напрямую
  - pig -x local my_pig_script.pig
  - pig -x mapreduce my_pig_script.pig
- Интерактивный режим использует оболочку Pig для запуска скрипта
  - Grunt> Lines = LOAD '/input/input.txt' AS (строка: chararray);
  - Grunt> Unique = DISTINCT Lines;
  - Grunt> DUMP Unique;

Spark

- Быстрая, выразительная кластерная вычислительная система, совместимая с Apache Hadoop
- Работает с любой системой хранения, поддерживаемой Hadoop (HDFS, S3, Avro,…)
- Повышает эффективность благодаря:
  - Примитивам вычислений в памяти
  - Графикам вычислений
- Улучшает удобство использования благодаря:
  - Богатым API в Java, Scala, Python
  - Интерактивной оболочке

Как запустить Spark

- Локальный многоядерный режим: просто библиотека в вашей программе
- EC2: скрипты для запуска кластера Spark
- Частный кластер: Mesos, YARN, Standalone Mode

Поддержка языков в Spark

- API в Java, Scala и Python
- Интерактивные оболочки в Scala и Python

Ключевая идея Spark

- **Работайте с распределенными коллекциями, как с локальными.**

- Концепция: устойчивые распределенные наборы данных (RDD)
- Неизменные коллекции объектов, распределенных по кластеру
- Построены через параллельные преобразования (map, filter, ...)
- Автоматически перестраивается при сбое
- Управляемое сохранение (например, кеширование в оперативной памяти)

Действия в Spark

- Transformations (e.g. map, filter, groupBy, join)
  - Ленивые операции, которые строят RDD из других RDD
- Actions (e.g. count, collect, save)
  - Возвращают результат или записывают его в хранилище

Действия в Spark

- Автономные программы могут быть написаны на любом из трех языков, но консоль - это только Python & Scala
- Разработчики Python: могут остаться с Python для обоих вариантов
- Java-разработчики: рассмотрите возможность использования Scala для консоли (для изучения API)
- Производительность: Java / Scala будет быстрее (статически типизированной), но Python лучше для вычислений с NumPy

Spark Context

- Главная точка входа в функциональность Spark
- Создан для вас в Spark shell как переменная sc
- В автономных программах вы создаете свои собственные контексты

```
from pyspark import SparkContext

sc = SparkContext(“masterUrl”, “name”, “sparkHome”, [“library.py”]))
```

Spark: создаем RDD


```
# Превратить локальную коллекцию в RDD
sc.parallelize ([1, 2, 3])

# Загрузить текстовый файл из локальной FS, HDFS или S3
sc.textFile («file.txt»)
sc.textFile («Каталог / *. TXT»)
sc.textFile («HDFS: // NameNode: 9000 / путь / файл»)

# Использовать любой существующий Hadoop InputFormat
sc.hadoopFile (keyClass, valClass, inputFmt, conf)
```

Spark: трансформации RDD

```
nums = sc.parallelize ([1, 2, 3])
# Пропустить каждый элемент через функцию
squares = nums.map (lambda x: x * x) # => {1, 4, 9}

# Отфильтровать по предикату
even = squares.filter (lambda x: x% 2 == 0) # => {4}

# Сопоставить каждый элементу нескольким другим
nums.flatMap(lambda x: range(0, x))  # => {0, 0, 1, 0, 1, 2}
```

Spark: действия с RDD


```
nums = sc.parallelize([1, 2, 3])

# Взять значение RDD в локальную переменную
nums.collect() # => [1, 2, 3]

# Взять первые K элементов
nums.take(2)   # => [1, 2]

# Получить количество элементов
nums.count()   # => 3

# Соединить элементы ассоциативной функцией
nums.reduce(lambda x, y: x + y)  # => 6

# Сохранить результаты в текстовый файл

nums.saveAsTextFile(“hdfs://file.txt”)
```

Spark

- Быстрая, выразительная кластерная вычислительная система, совместимая с Apache Hadoop
- Работает с любой системой хранения, поддерживаемой Hadoop (HDFS, S3, Avro,…)
- Повышает эффективность благодаря:
  - Примитивам вычислений в памяти
  - Графикам вычислений
- Улучшает удобство использования благодаря:
  - Богатым API в Java, Scala, Python
  - Интерактивной оболочке

Заключение

- Экосистема Hadoop огромна, и содержит элементы для практически любых нужд;
- Hadoop ориентирован на доставку запросов к данным, а не наоборот;
- У Hadoop высокое время отклика, поэтому он не очень подходит для прямого взаимодействия с пользователями.

Спасибо за внимание!

mgubin@tpu.ru

econophysica