Распределенные Вычисления. Apache Hadoop

NEWPROLAB.COM

Что такое распределенная система?

- распределенная система это набор компьютеров, представляющихся пользователю единой системой

- распределенная система это такая система, в которой взаимодействие и синхронизация программных компонентов, выполняемых на независимых компьютерах, осуществляется посредством передачи сообщений

- распределенная система - набор независимых компьютеров, не имеющих совместно используемой памяти и общего единого времени и взаимодействующих посредством передачи сообщений

Основные признаки распределенных систем

- отсутствие единого времени

- отсутствие общей памяти - каждый узел оперирует только своей физической оперативной памятью (возможны варианты виртуальной адресации)

- географическое распределение вычислительных узлов

- независимость и гетерогенность - вычислительные узлы могут иметь разную производительность и разные конфигурации

Зачем нужны распределенные системы

- увеличение производительности вычислений

- географическое распределение вычислений

- совместное использование ресурсов

- отказоустойчивость

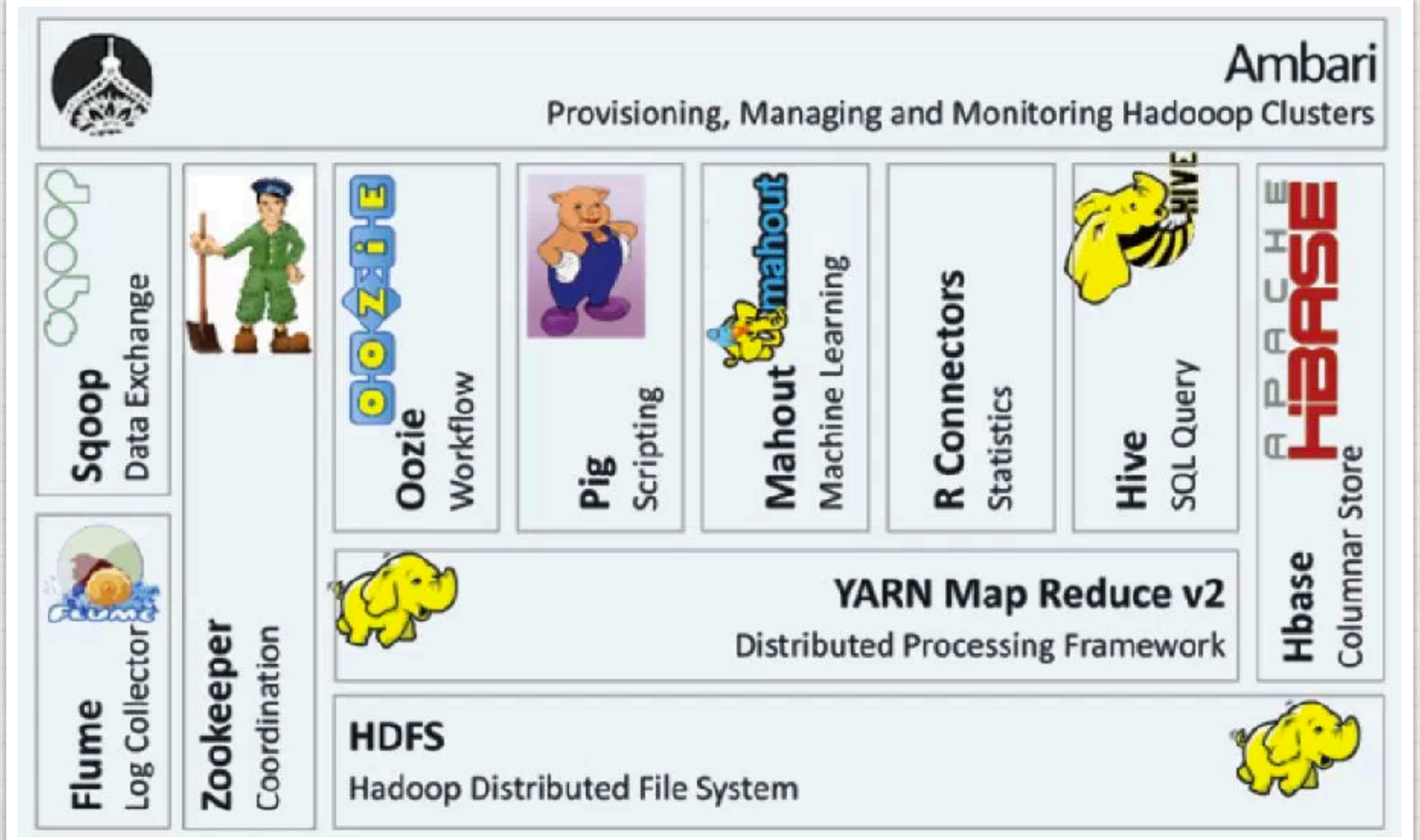
Восемь заблуждений относительно распределенных вычислений

1. Сеть является надежной
2. Задержки передачи сообщения равны нулю
3. Полоса пропускания неограниченно
4. Сеть является безопасной
5. Сетевая топология неизменна
6. Система обслуживается одним администратором
7. Издержки транспортной инфраструктуры равны нулю
8. Сеть является однородной

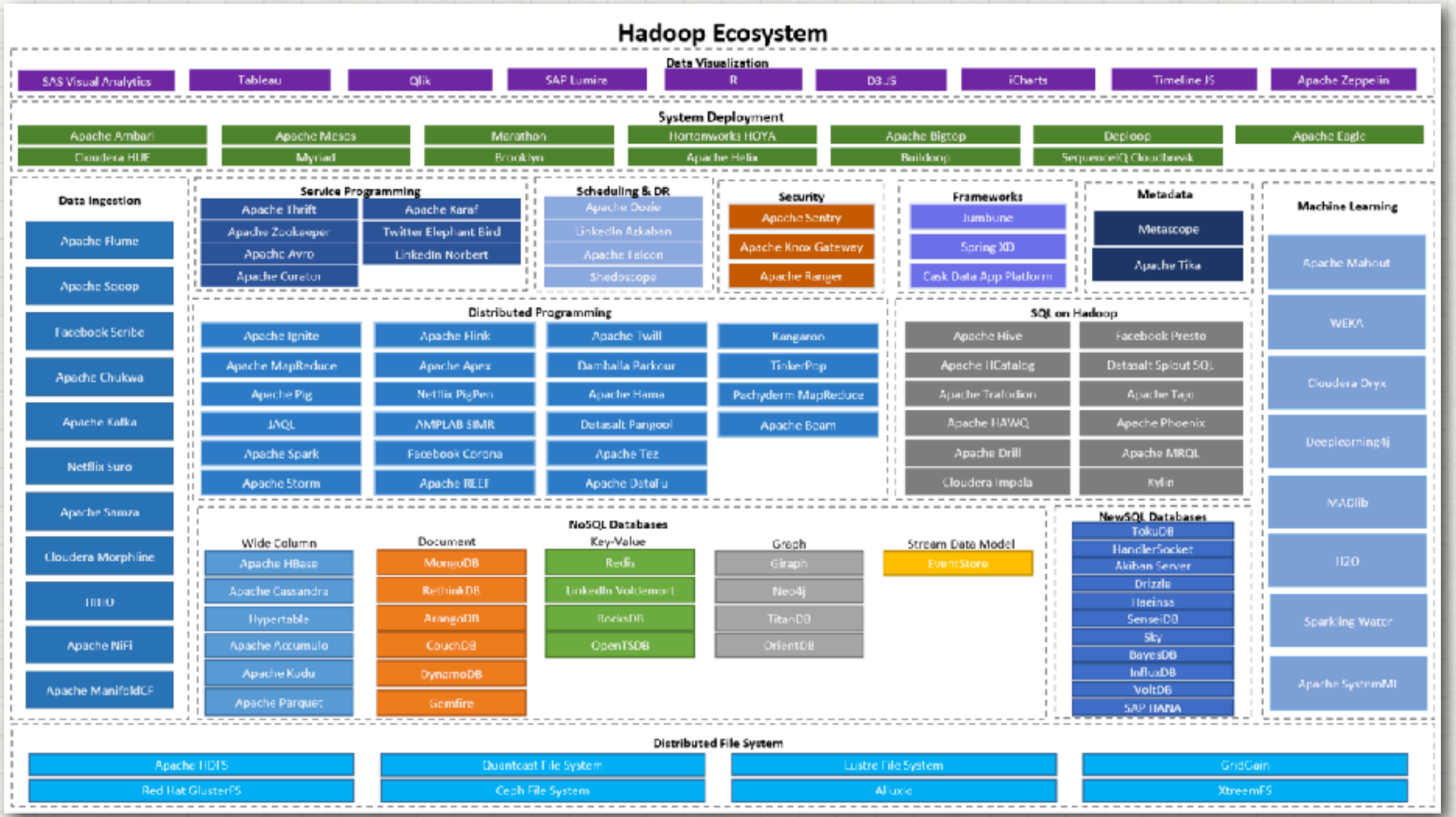
HADOOP HISTORY

- 2002 – запуск проекта Nutch
- 2003 – публикация с описанием GFS
- 2004 – создание NDFS (Nutch Distributed File System)
- 2004 – публикация Google и MapReduce
- 2005 – реализация MR в Nutch
- 2006 – выделение проекта Hadoop
- 2008 – выход Hadoop в лидеры ASF (Apache Software Foundation)
- 2008 - Apache Pig
- 2009 - Apache Scoop
- 2010 - выделение проекта HBase (ноябрь - Facebook messages)
- 2010 - Apache Hive
- 2011 - Apache Kafka
- 2012 - Apache Impala
- 2012 - Apache Accumulo
- 2013 - Apache YARN
- 2013 - Apache Spark (начало работы в 2009)

HADOOP WORLD 1.0



HADOOP WORLD 2017



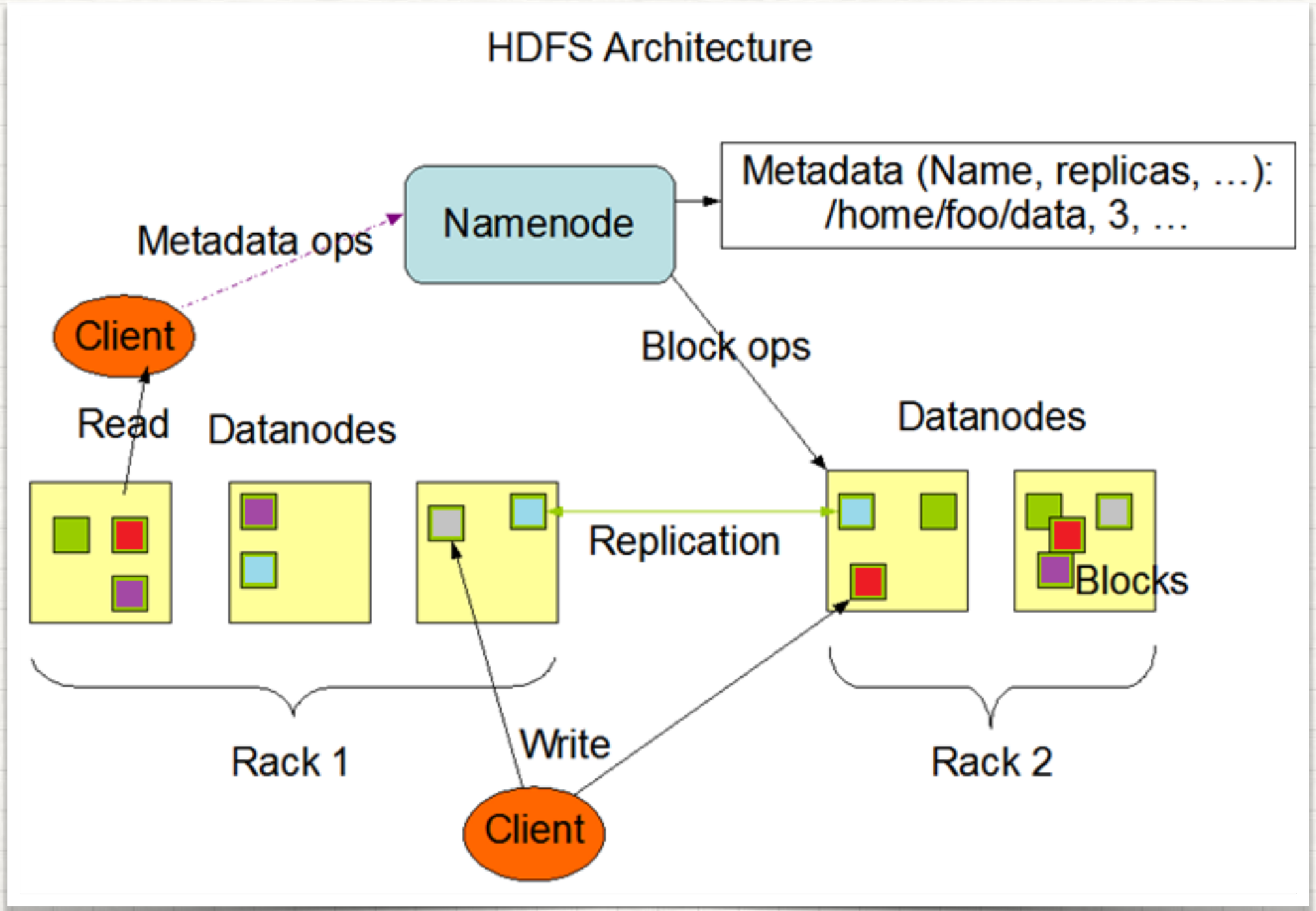
HADOOP BASICS

1.Hadoop Distributed File System (HDFS):

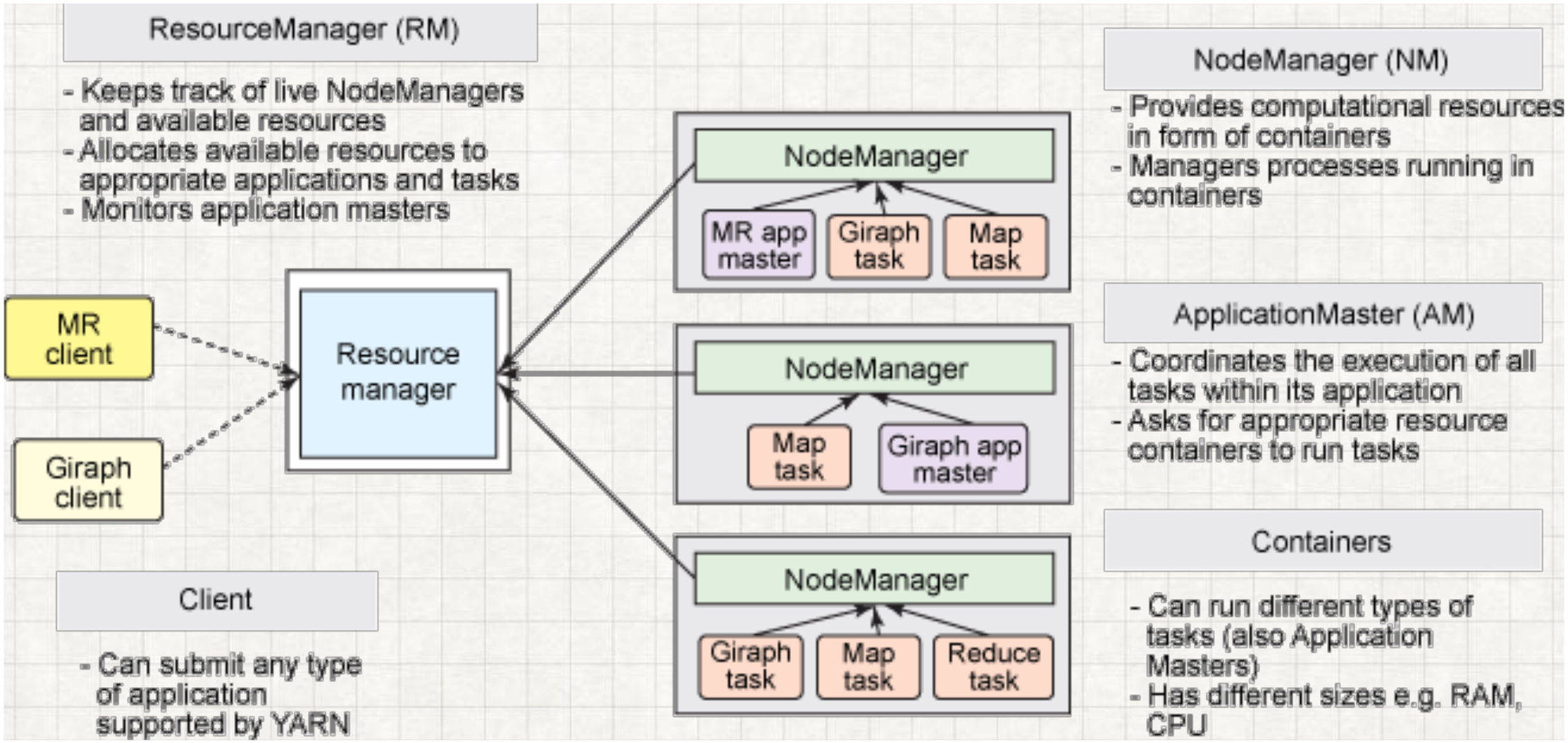
2.Hadoop YARN

3.Hadoop MapReduce

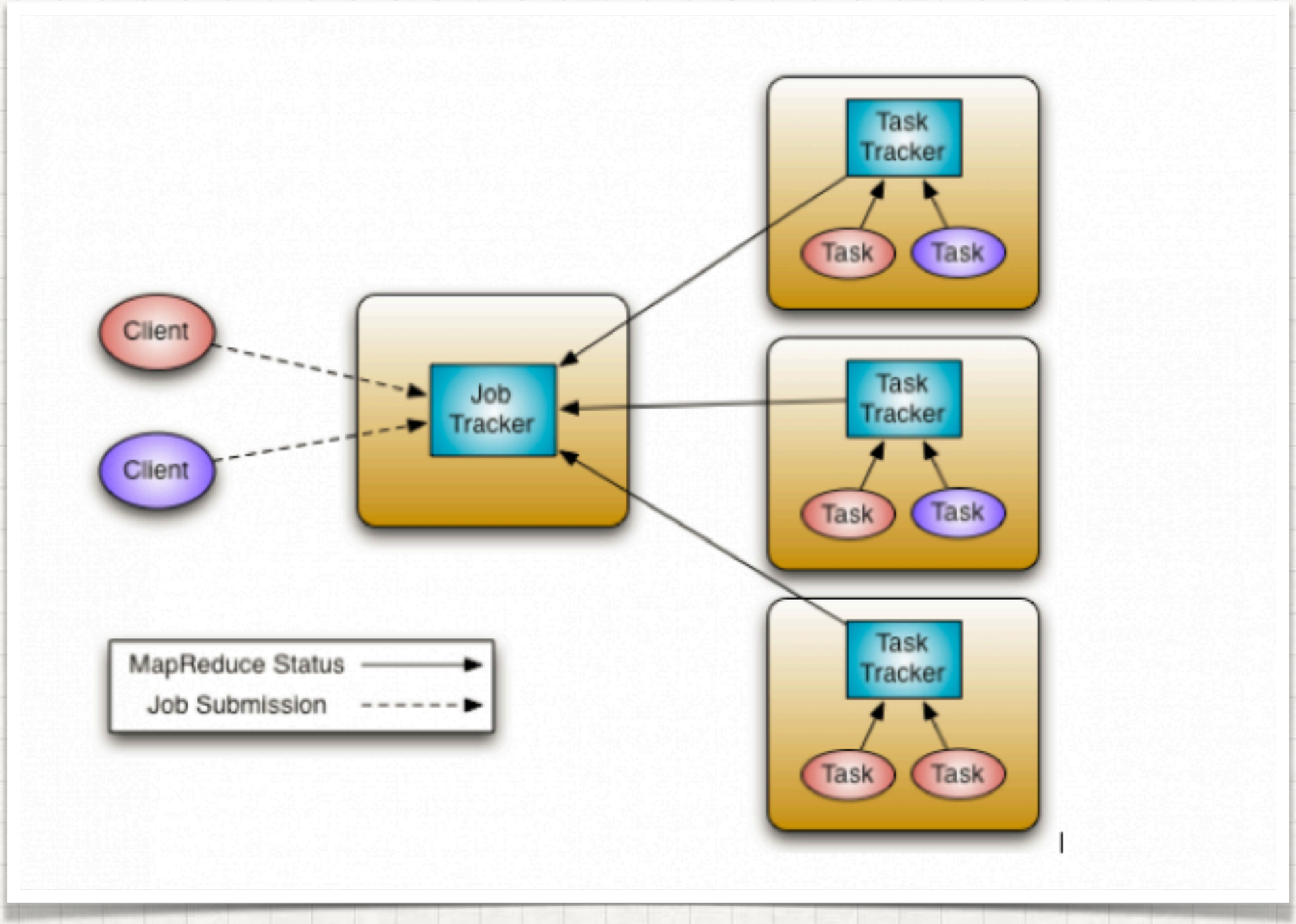
HDFS BASICS


HDFS Architecture
Namenode
Metadata (Name, replicas, …):
/home/foo/data, 3, …
Metadata ops
Client
Block ops
Read
Datanodes
Datanodes
Replication
Blocks
Rack 1
Write
Client
Rack 2

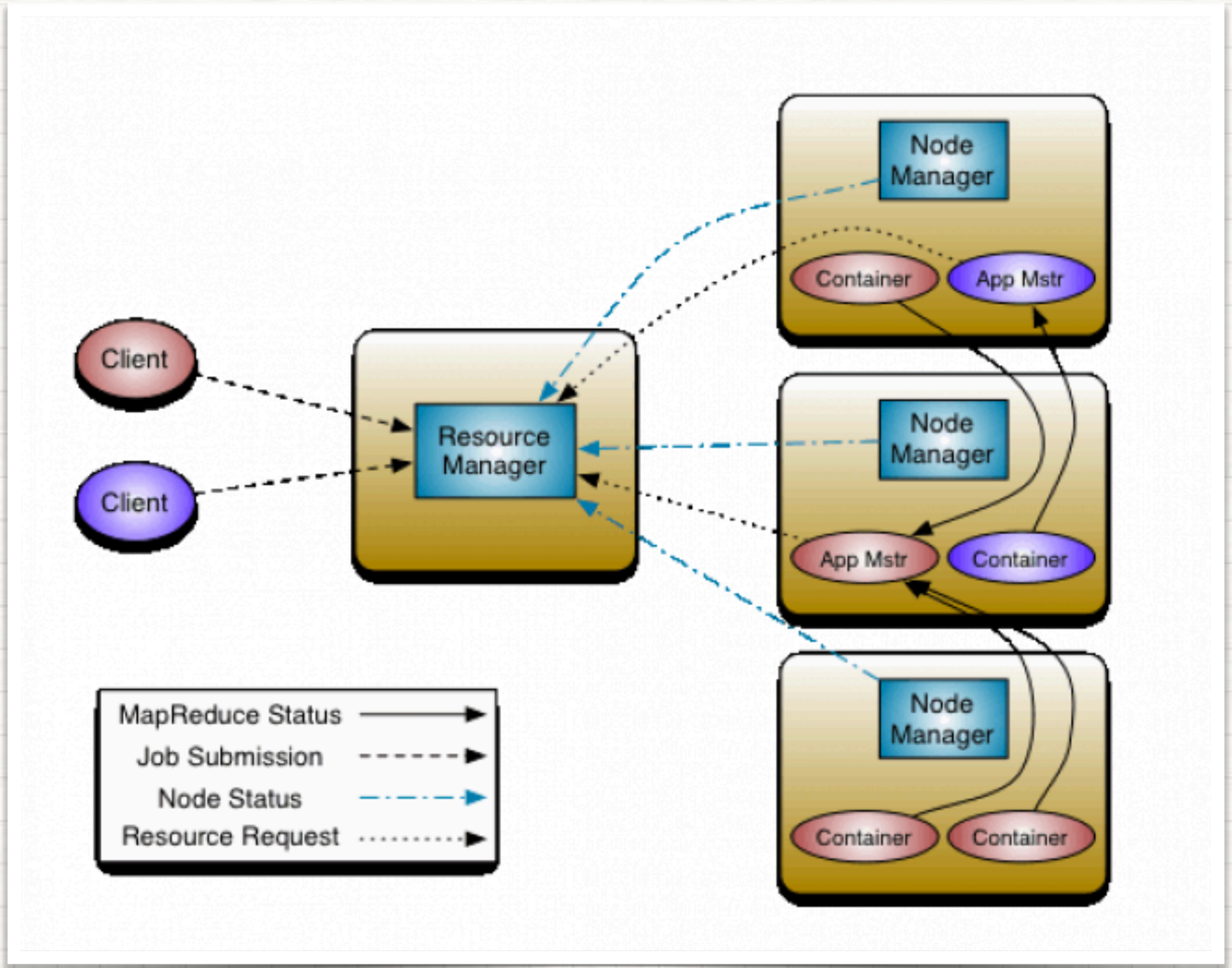
YARN BASICS


ResourceManager (RM)
- Keeps track of live NodeManagers and available resources
- Allocates available resources to appropriate applications and tasks
- Monitors application masters
MR client
Giraph client
Resource manager
Client
- Can submit any type of application supported by YARN
NodeManager
MR app master
Giraph task
Map task
NodeManager
Map task
Giraph app master
NodeManager
Giraph task
Map task
Reduce task
NodeManager (NM)
- Provides computational resources in form of containers
- Managers processes running in containers
ApplicationMaster (AM)
- Coordinates the execution of all tasks within its application
- Asks for appropriate resource containers to run tasks
Containers
- Can run different types of tasks (also Application Masters)
- Has different sizes e.g. RAM, CPU

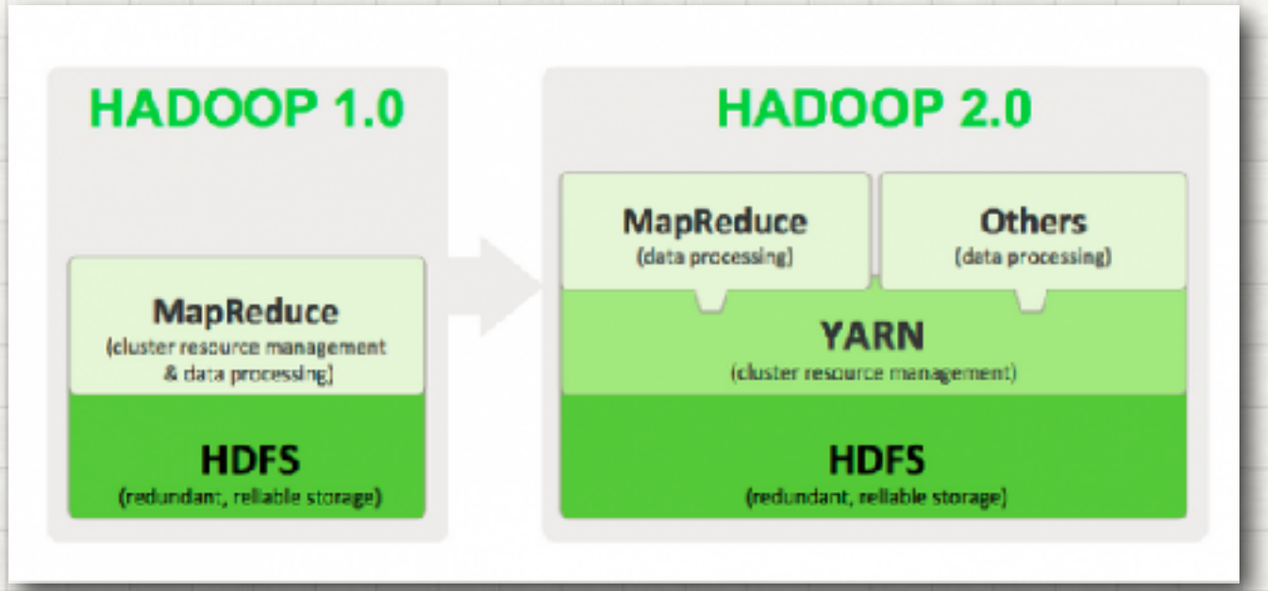
MR 1


Client
Client
Job Tracker
Task Tracker
Task
Task
Task Tracker
Task
Task
Task Tracker
Task
Task
MapReduce Status
Job Submission

YARN (MR 2)

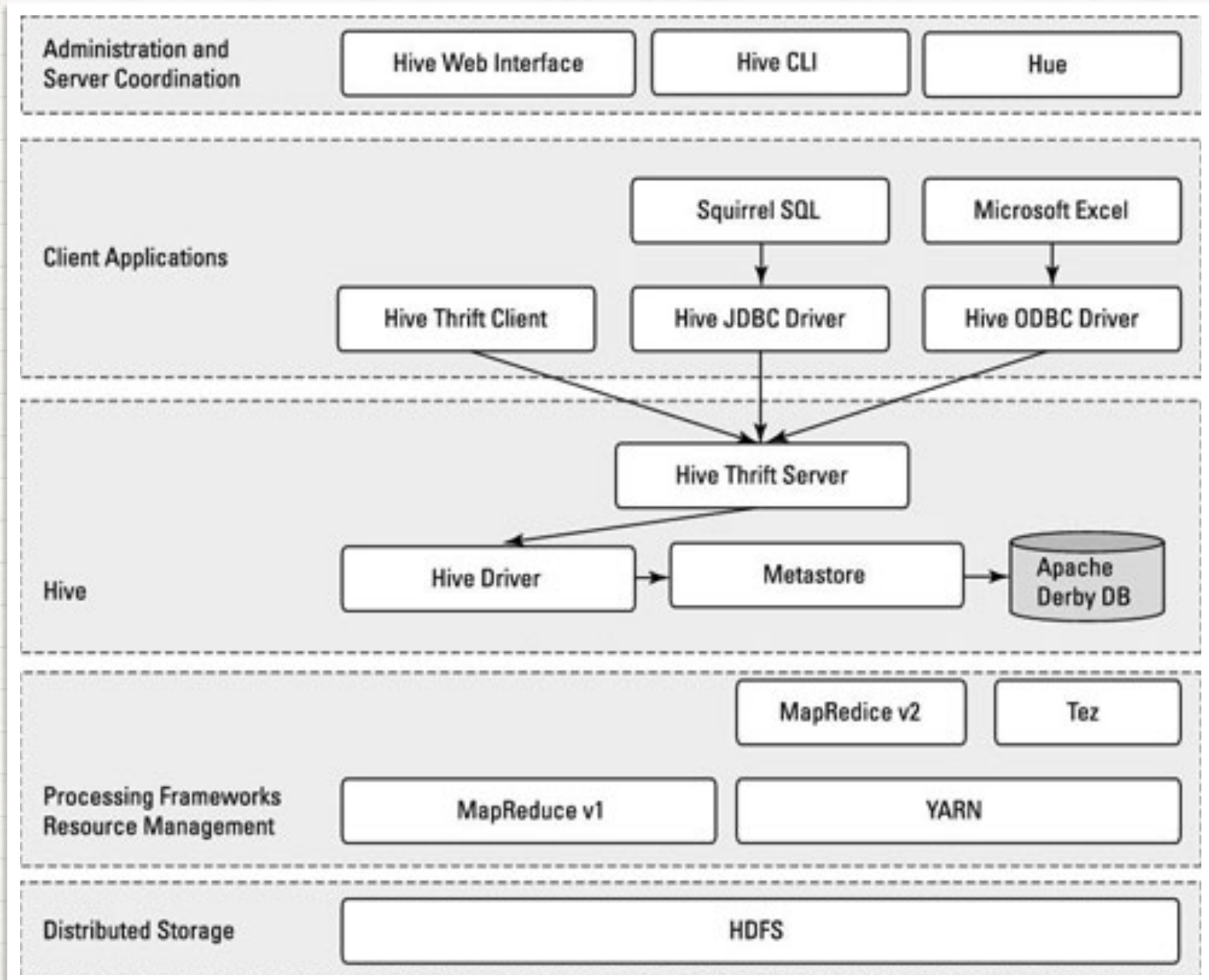

Client
Client
Resource Manager
Node Manager
Container
App Mstr
Node Manager
App Mstr
Container
Node Manager
Container
Container
MapReduce Status
Job Submission
Node Status
Resource Request

YARN

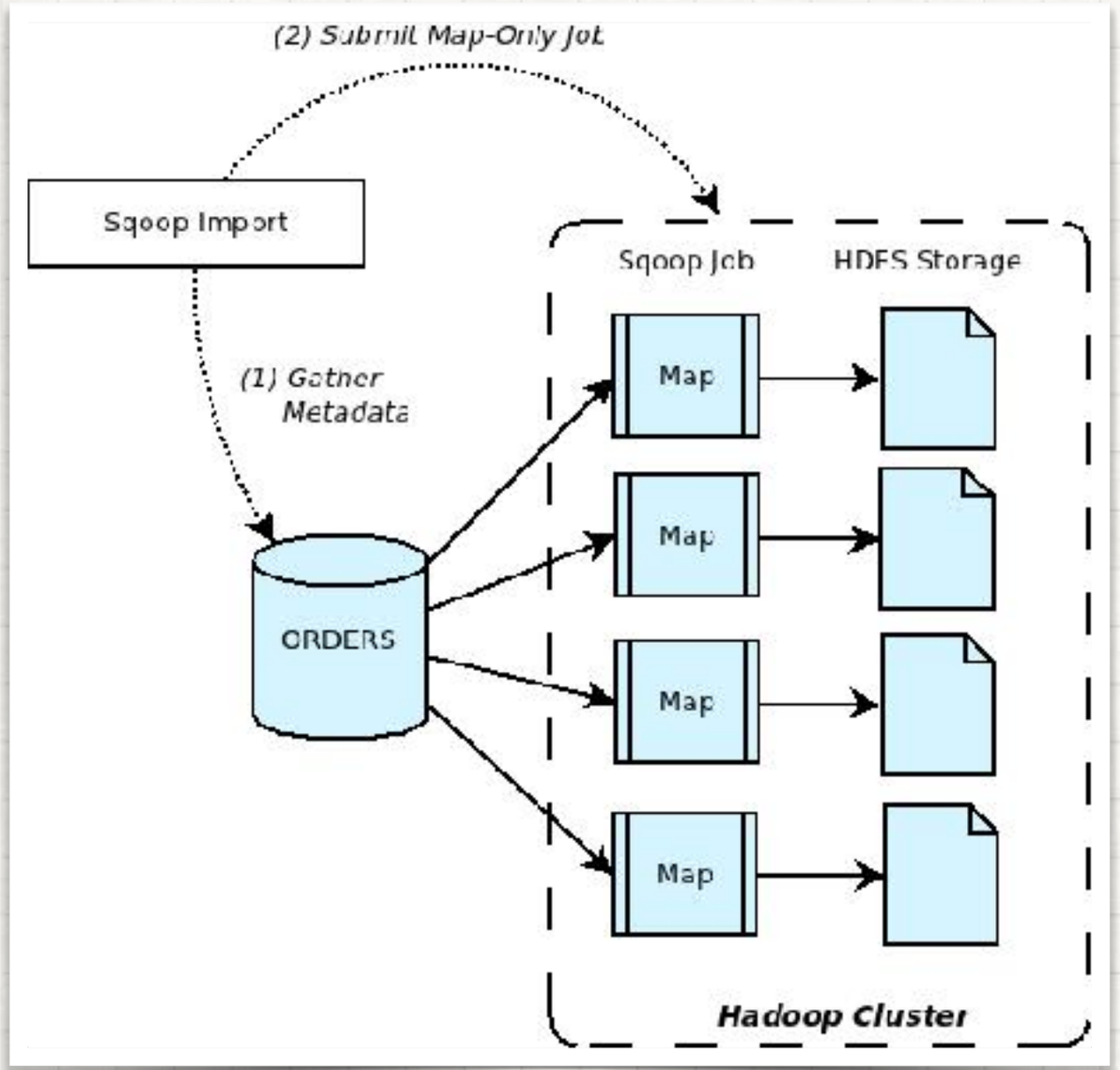

HADOOP 1.0
MapReduce
(cluster resource management & data processing)
HDFS
(redundant, reliable storage)
HADOOP 2.0
MapReduce
(data processing)
Others
(data processing)
YARN
(cluster resource management)
HDFS
(redundant, reliable storage)

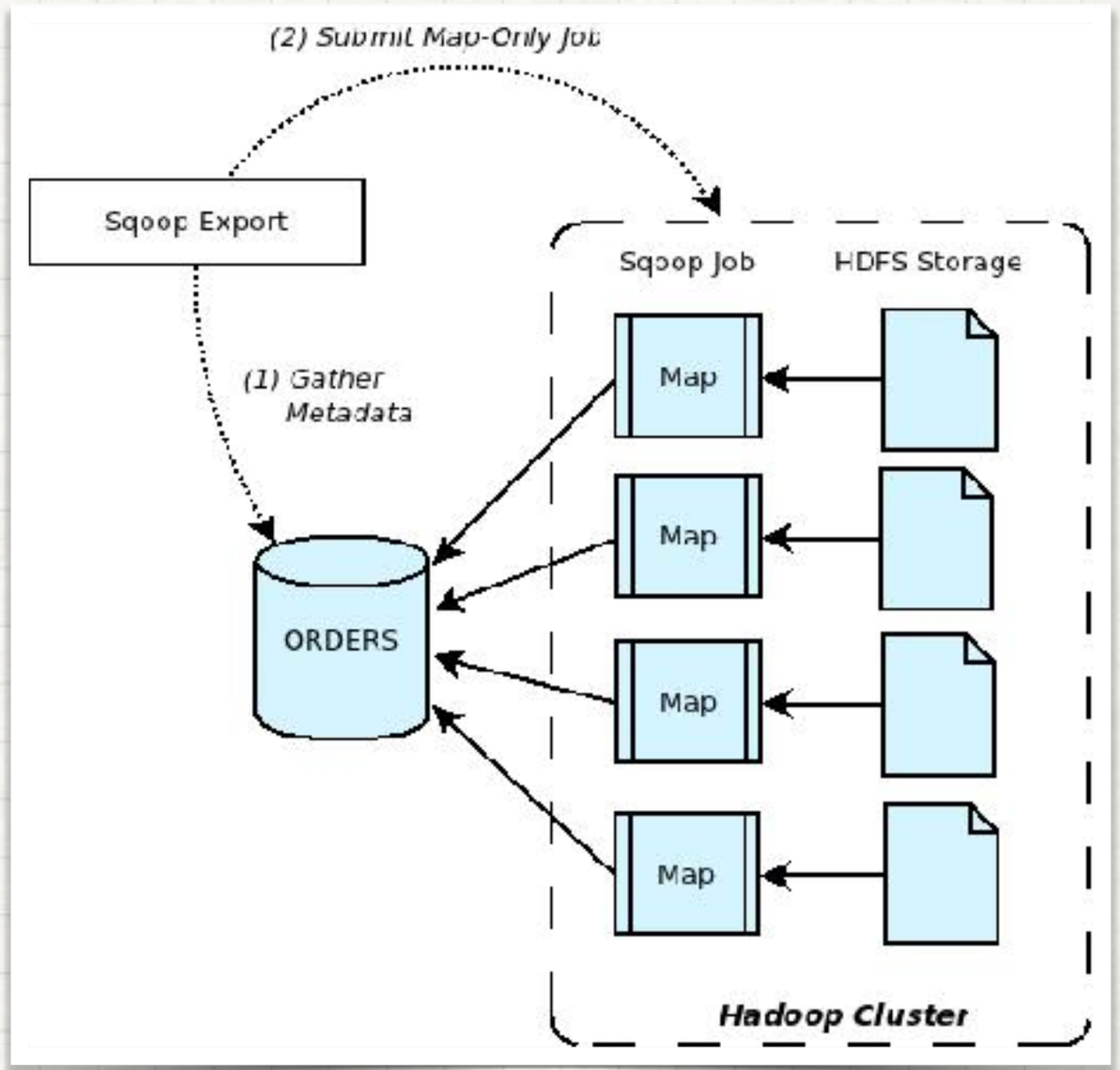

Applications Run Natively IN Hadoop
BATCH (MapReduce)
INTERACTIVE (Tez)
ONLINE (HBase)
STREAMING (Storm, S4,...)
GRAPH (Giraph)
IN-MEMORY (Spark)
HPC MPI (OpenMPI)
OTHER (Search) (Weave...)
YARN (Cluster Resource Management)
HDFS2 (Redundant, Reliable Storage)

HIVE BASICS

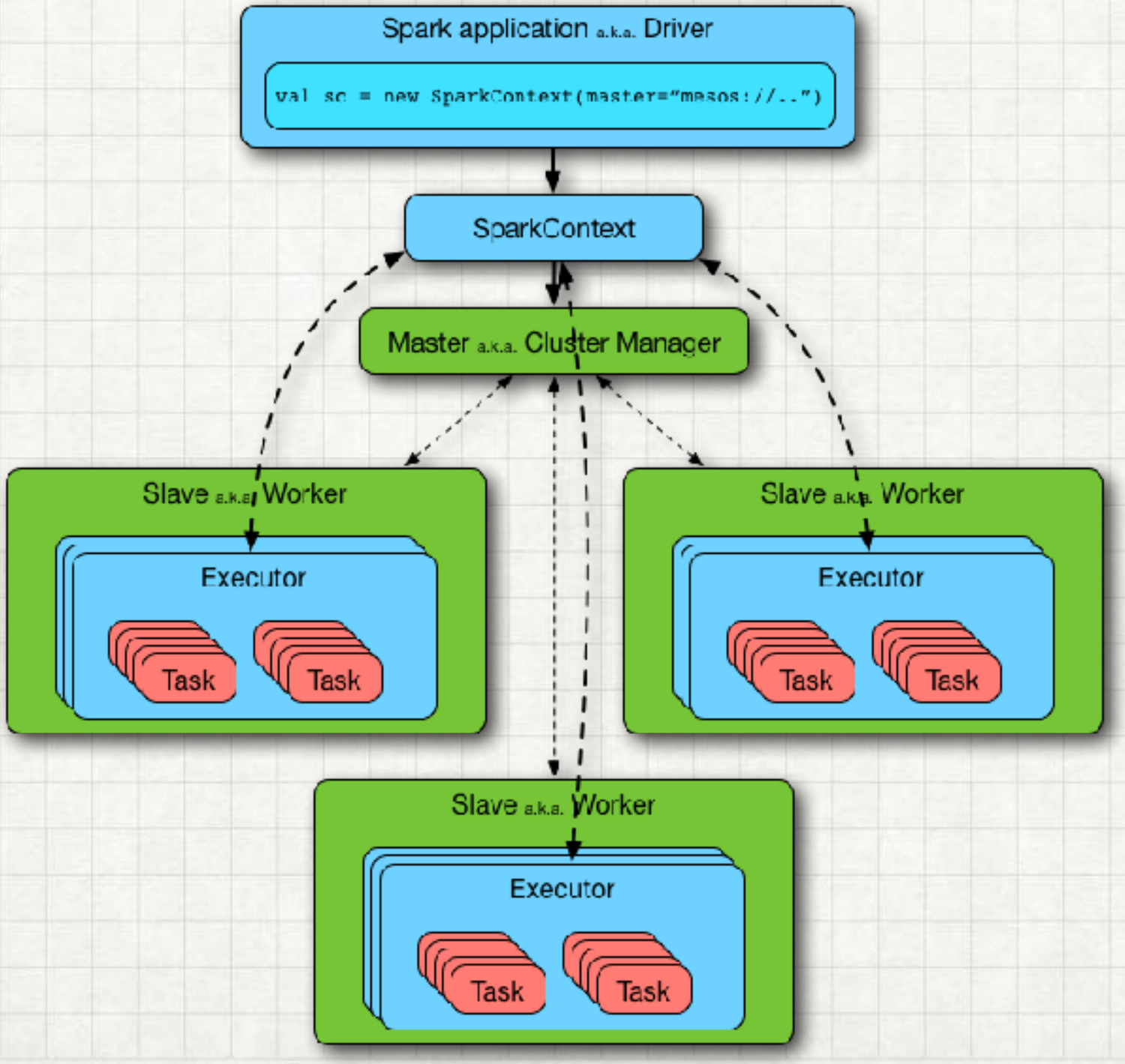

Administration and Server Coordination
Hive Web Interface
Hive CLI
Hue
Client Applications
Squirrel SQL
Microsoft Excel
Hive Thrift Client
Hive JDBC Driver
Hive ODBC Driver
Hive Thrift Server
Hive
Hive Driver
Metastore
Apache Derby DB
MapRedice v2
Tez
Processing Frameworks Resource Management
MapReduce v1
YARN
Distributed Storage
HDFS

SQOOP BASICS

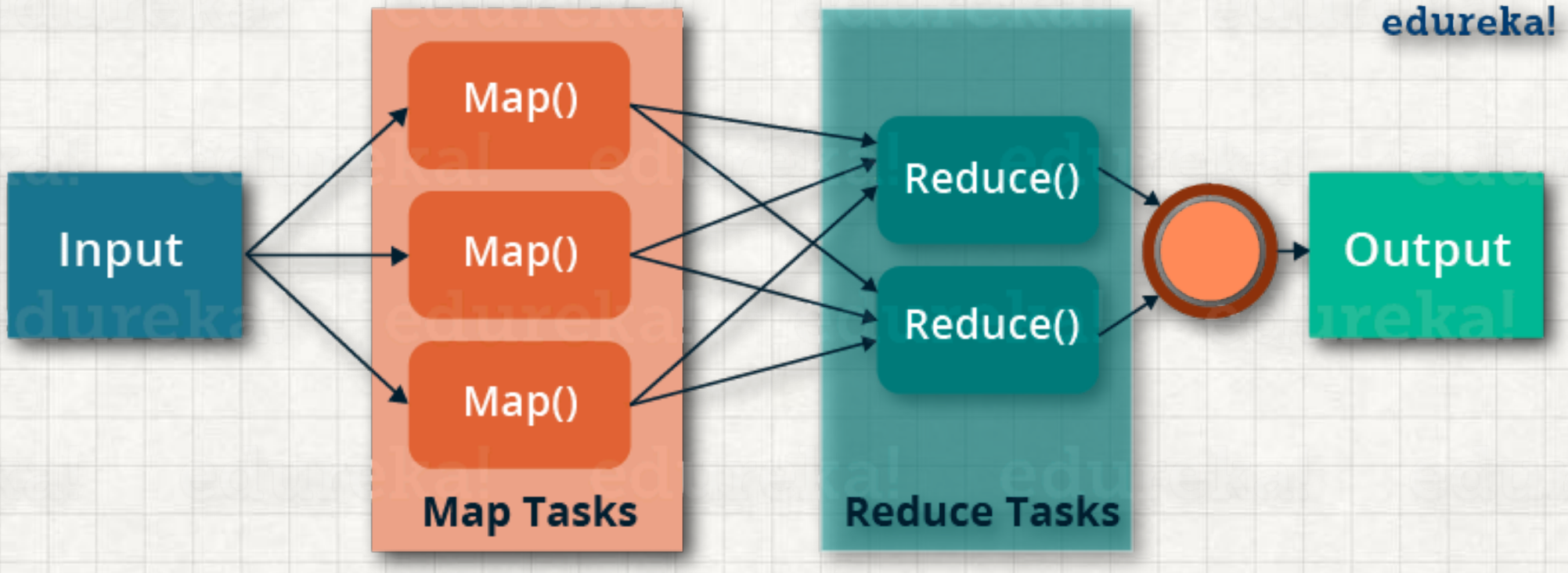


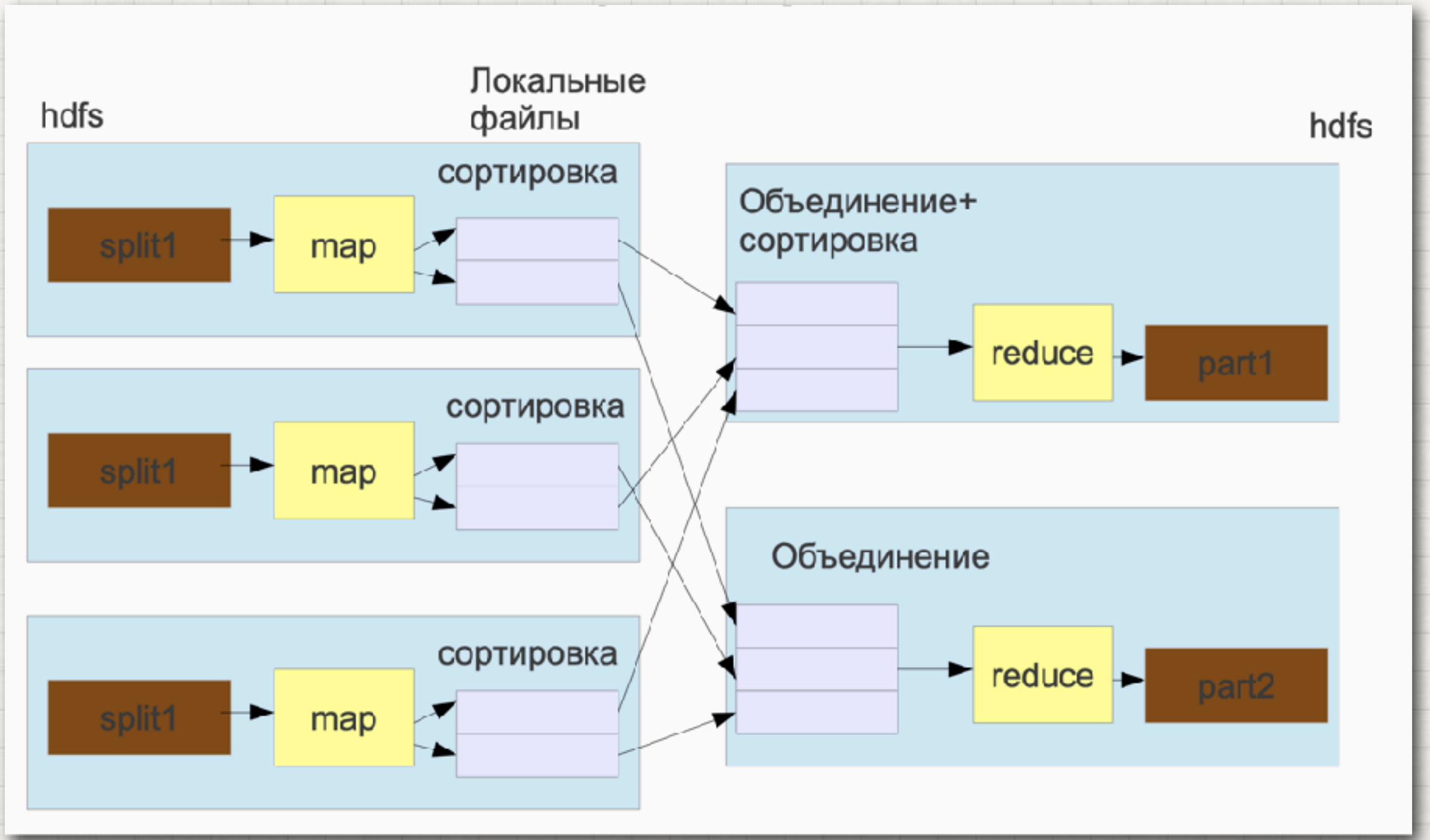


SPARK BASICS



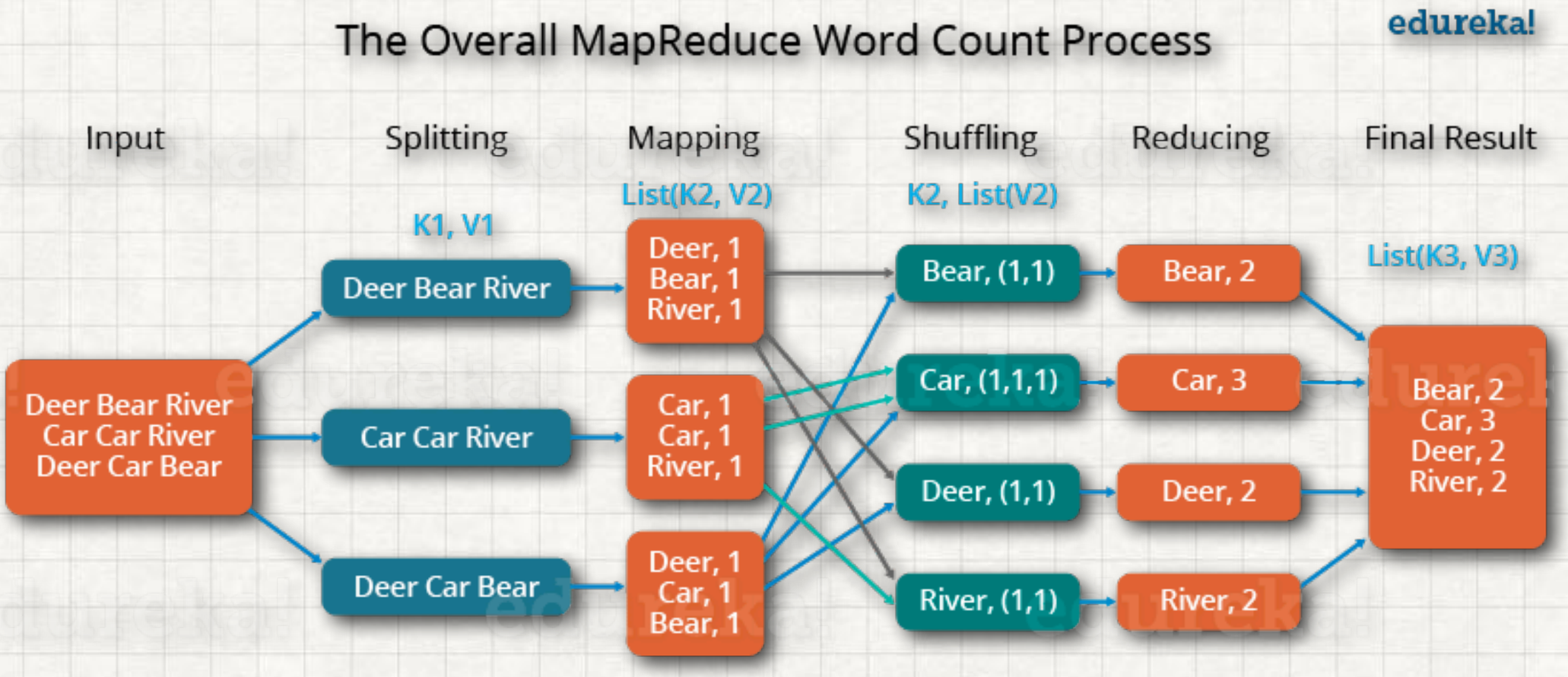
MAP-REDUCE


edureka!
Input
Map()
Map()
Map()
Map Tasks
Reduce()
Reduce()
Reduce Tasks
Output


hdfs
Локальные файлы
сортировка
split1
map
сортировка
split1
map
сортировка
split1
map
hdfs
Объединение+ сортировка
reduce
part1
Объединение
reduce
part2

The Overall MapReduce Word Count Process

edureka!



INPUTFORMAT

- **Основные функции** :
  - **Разбивает исходные данные на** splits
  - **Обеспечивает чтение данных из** split **в процессе работы** mapper
  -

БАЗОВЫЕ КЛАССЫ

```
public abstract class InputFormat {

   public abstract List getSplits(JobContext context)..;

   public abstract RecordReader createRecordReader( InputSplit
   split, TaskAttemptContext context) ..

   }
```

БАЗОВЫЕ КЛАССЫ

```
public abstract class InputSplit {

    public abstract long getLength() throws IOException,
    InterruptedException;

    public abstract String[] getLocations() throws …;

}
```

СЕРИАЛИЗАЦИЯ

- **Сериализация — преобразование объекта в байтовый поток данных**
- **Десериализация — преобразование потока байтов в объект.**
- **Базовый интерфейс объекта сериализации в** Hadoop — org.apache.hadoop.io.Writable
- public interface Writable { void write(DataOutput out) throws IOException; void readFields(DataInput in) throws IOException; }

MAPPER

- **Базовый класс для реализации** mapper : org.apache.hadoop.mapreduce.Mapper
- **Базовый метод для перегрузки**
- protected void map(KEYIN key, VALUEIN value, Mapper.Context context)
- **Для того чтобы добавить результат функции** map **требуется вызвать** context.write(KEYOUT key, VALUEOUT value)

MAPPER

```
public class WordMapper extends Mapper {

    @Override protected void map(LongWritable key, Text value, Context context)

        throws IOException, InterruptedException {

        String line = value.toString();

        String[] words = StringTools.split(StringTools.removeAllNonSymbols(line).toLowerCase());

        for (String word : words) {

            context.write(new Text(word), new IntWritable(1));

        }

    }

}
```

REDUCER

- **Базовый класс для реализации** reducer : org.apache.hadoop.mapreduce.Reducer
- **Базовый метод для перегрузки**
- protected void reduce(KEYIN key, Iterable values, Reducer.Context context)
- **Для того чтобы добавить результат функции** map **требуется вызвать** context.write(KEYOUT key, VALUEOUT value)

ПРИМЕР

```
public class WordReducer extends Reducer {

    @Override protected void reduce(Text key, Iterable values, Context context) throws IOException,
    InterruptedException {

        long count=0;

        Iterator iter = values.iterator();

    while(iter.hasNext()) {

        iter.next();

        count++;

    }

    context.write(key, new LongWritable(count)); } }
```

DRIVER

```
public class WordCountApp {
    public static void main(String[] args) throws Exception {
    if (args.length != 2) {
        System.err.println("Usage: WordCountApp "); System.exit(-1); }
    Job job = Job.getInstance();
    job.setJarByClass(WordCountApp.class);
    job.setJobName("Word count");
    FileInputFormat.addInputPath(job, new Path(args[0]));
    FileOutputFormat.setOutputPath(job, new Path(args[1]));
    job.setMapperClass(WordMapper.class);
    job.setReducerClass(WordReducer.class);
    job.setOutputKeyClass(Text.class);
    job.setOutputValueClass(IntWritable.class);
```