

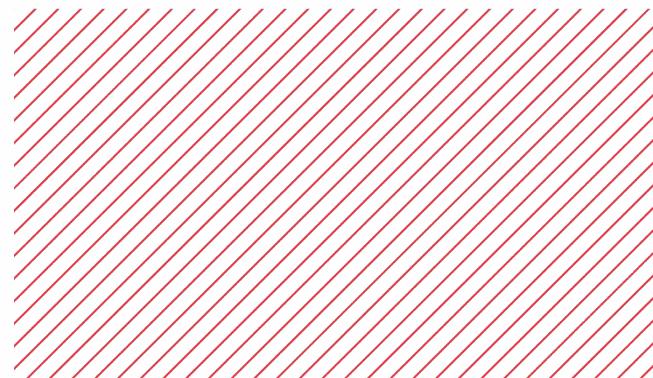
академия  
больших  
данных



# Hadoop экосистема и MapReduce

Андрей Кузнецов

26.09.2020



# Структура курса

---

1. Введение в Большие Данные
2. Hadoop экосистема и MapReduce ←
3. SQL поверх больших данных
4. Инструменты визуализации при работе с Большими Данными
5. Введение в Scala
6. Модель вычислений Spark: RDD
7. Распараллеливание алгоритмов ML
8. Spark Pipelines
9. Approximate алгоритмы для больших данных
10. Spark для оптимизации гиперпараметров
11. Потоковая обработка данных (Kafka, Spark Streaming, Flink)
12. Архитектуры в продакшене



# План занятия

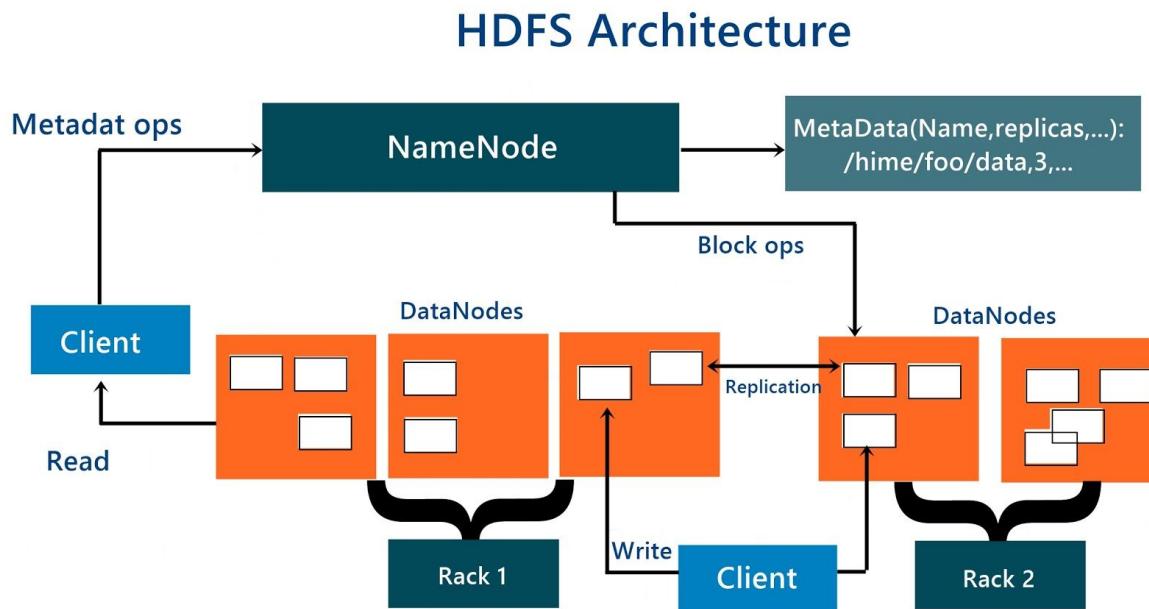
---

1. Концепция MapReduce
2. Реализация MapReduce
3. Менеджер ресурсов YARN
4. MapReduce API and Hadoop Streaming

# Концепция MapReduce



# Quick recap. Hadoop Architecture



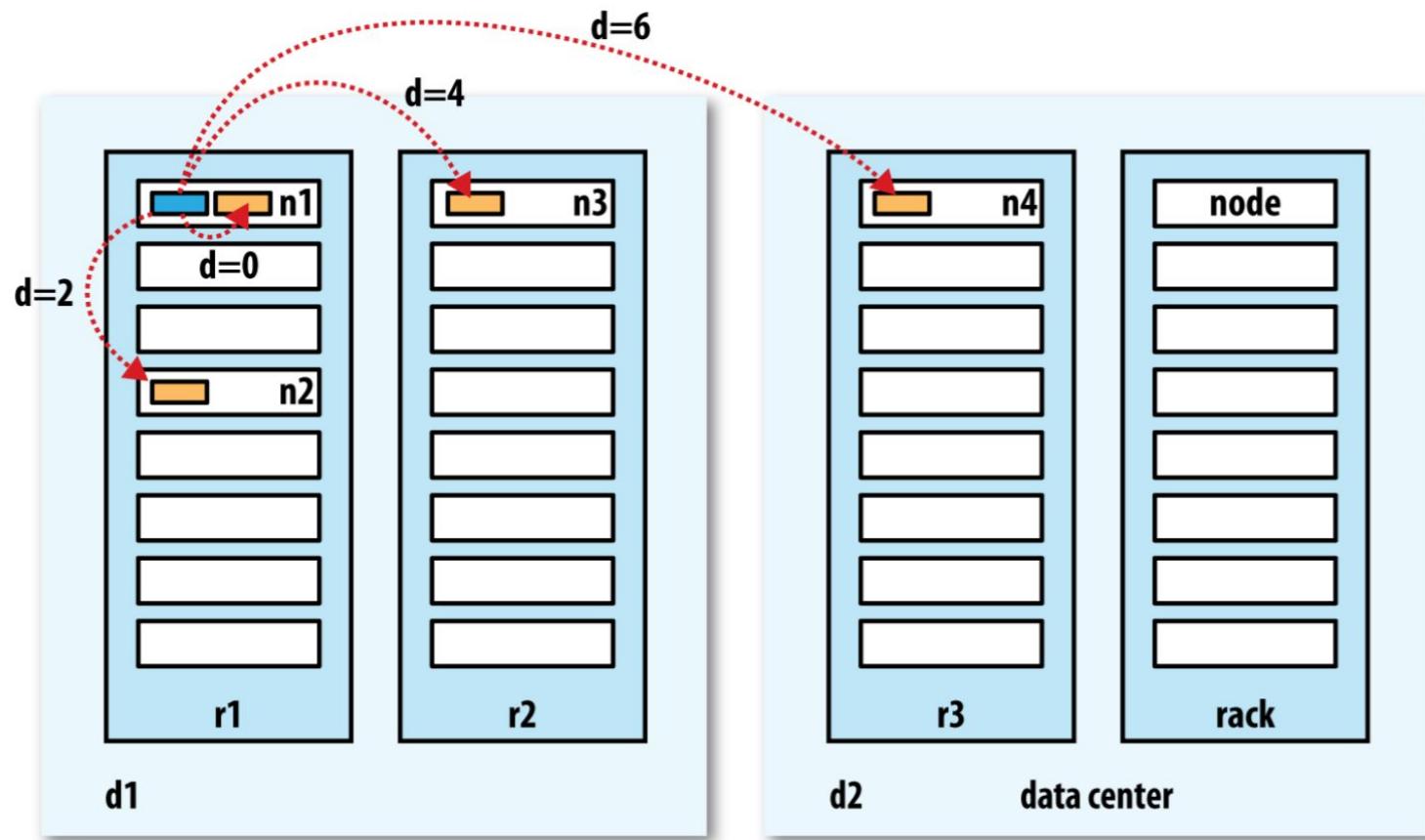
## Namenode:

- хранит пространства имен
- хранит расположение блоков файлов
- ведет журнал изменений на диске
- SPoF

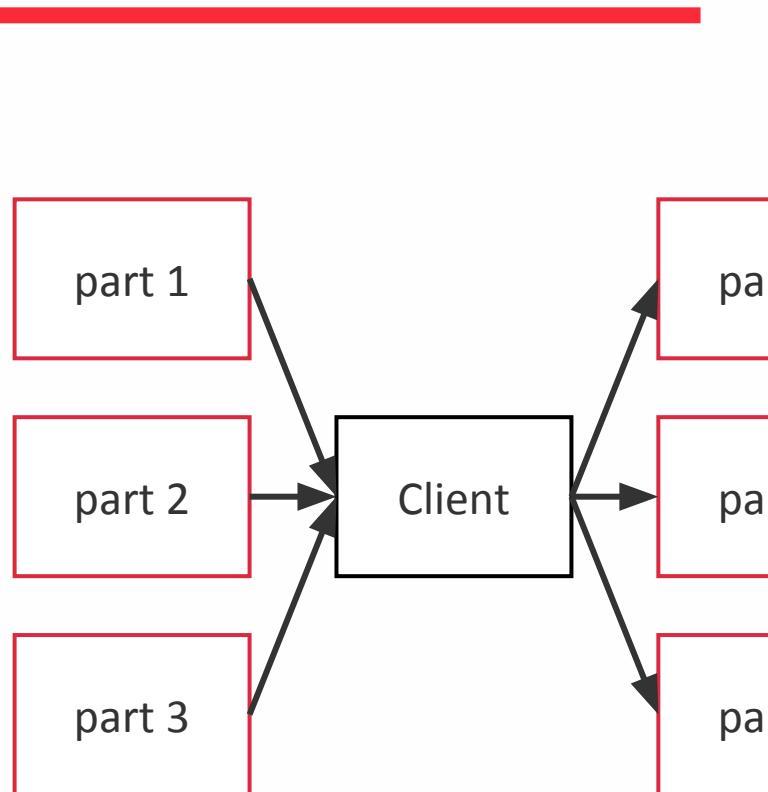
## Datanode:

- непосредственно хранит данные
- отчитывается о своем состоянии Namenode

# Quick recap. Locality & distances

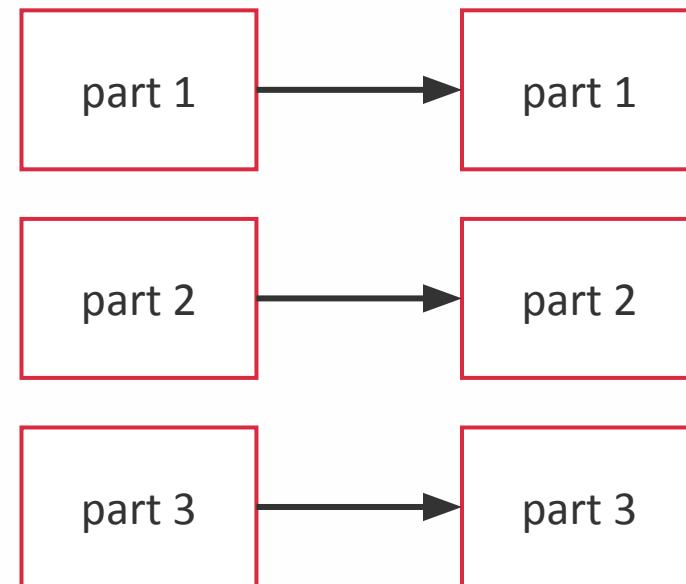


# DistCp vs cp in real life



hdfs dfs -cp

**MapReduce**



hdfs distcp



# MapReduce. Definitions

---

**MapReduce** - модель распределенных вычислений.

Реализация Hadoop MapReduce на Java. Сегодня напрямую используется редко, в основном это поддержка легаси.

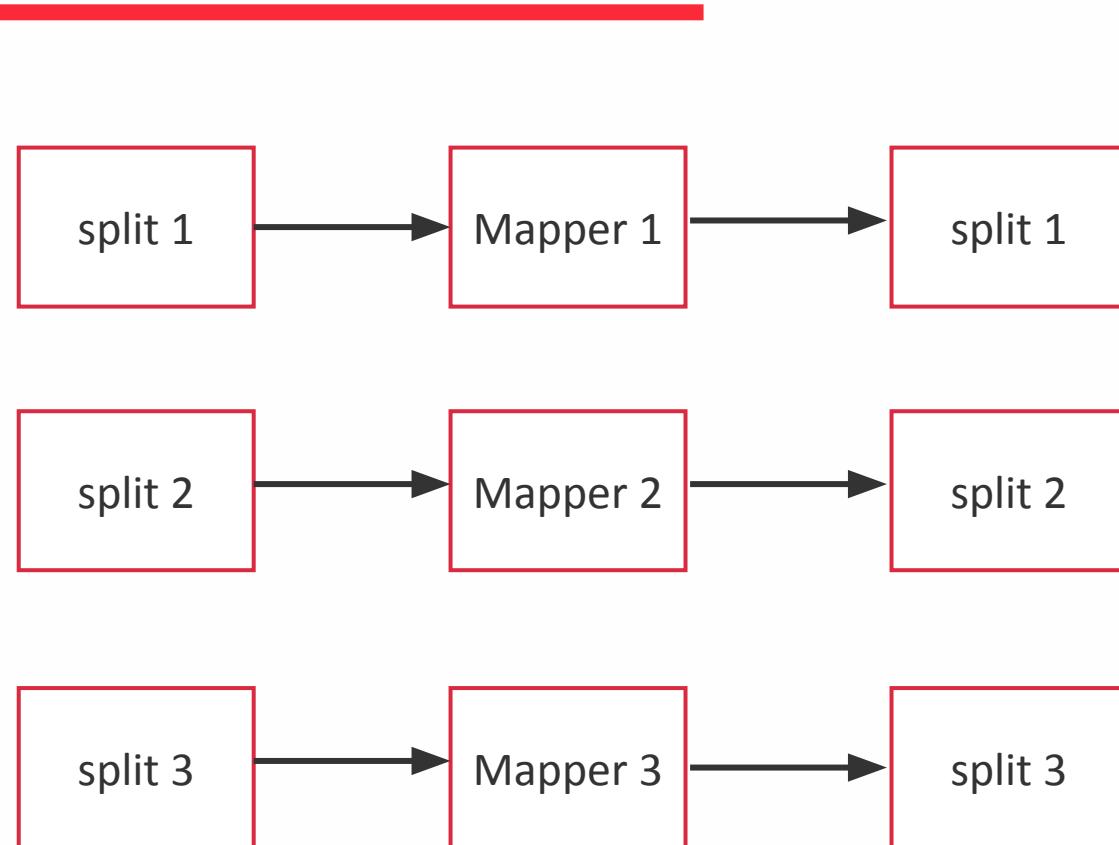
**Map** - обработка данных → Mapper

**Reduce** - свертка данных → Reducer

**Split** - независимая (!!!) часть для обработки

Работаем с **Key-Value** данными

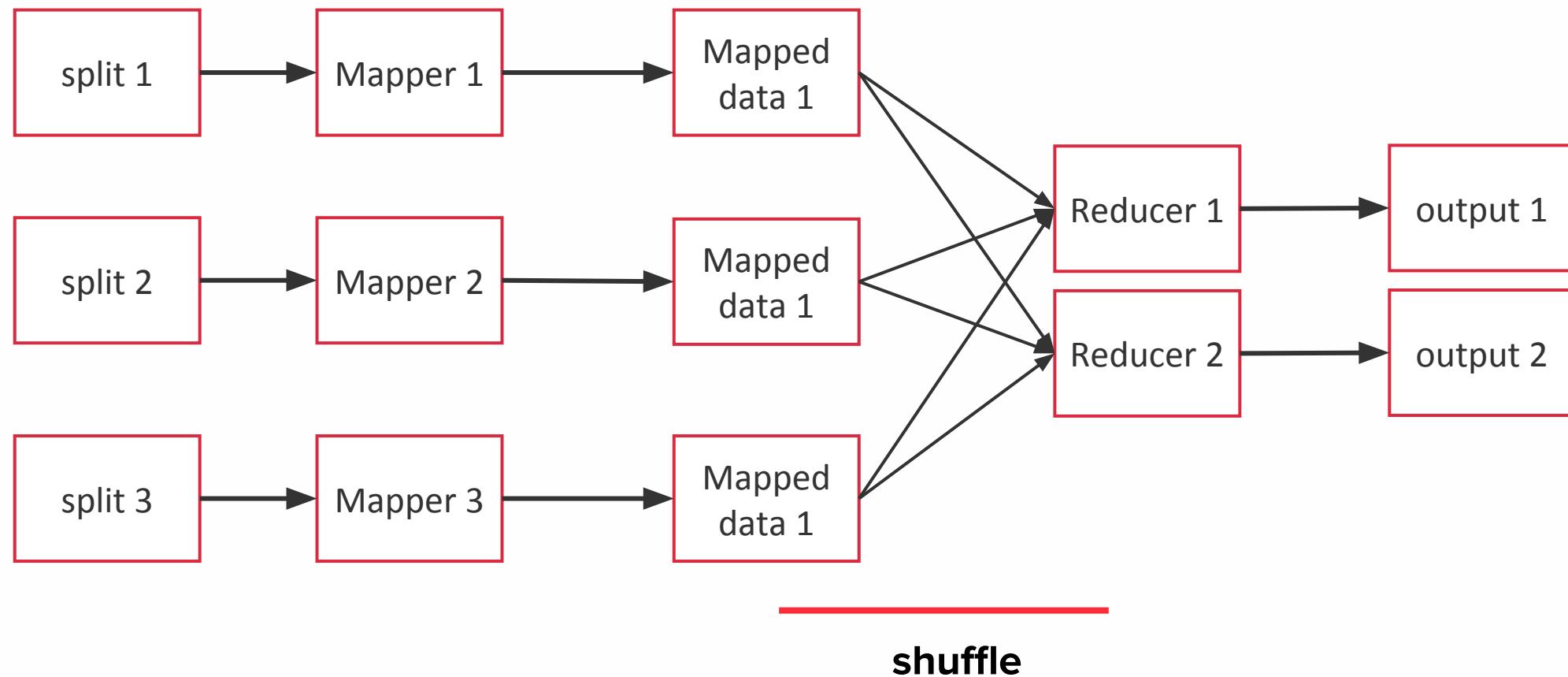
# MapReduce. No reduce



Зачем может быть такое нужно:

1. Просто скопировать данные
2. Отфильтровать данные
3. Сконвертировать данные

# MapReduce. Whole concept



# Реализация MapReduce



# MapReduce. Input data

split 1

split 2

split 3

Класс **InputFormat** ответственен за разбиение входных данных на сплиты.

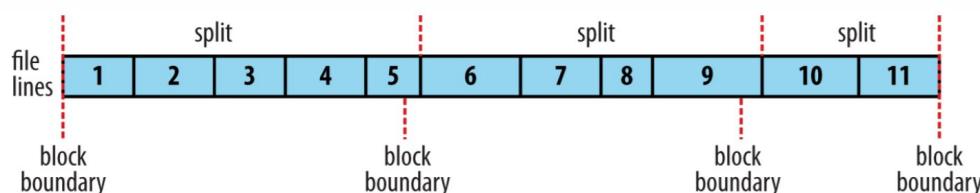
Размер сплита считается по формуле:

$$\max(\minSize, \min(\maxSize, \text{blockSize}))$$

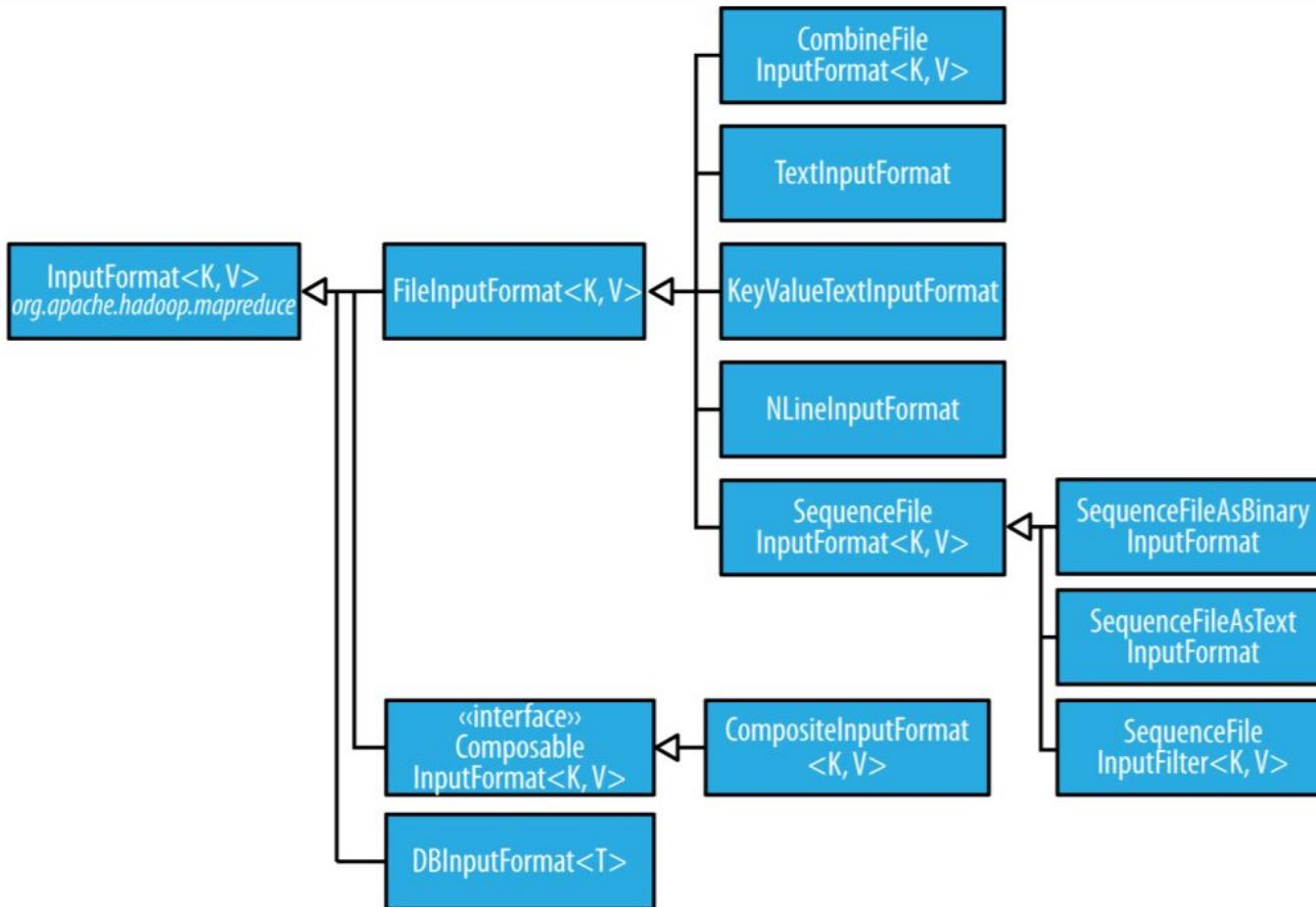
$\maxSize \rightarrow \text{mapred.max.split.size}$

$\minSize \rightarrow \text{mapred.min.split.size}$

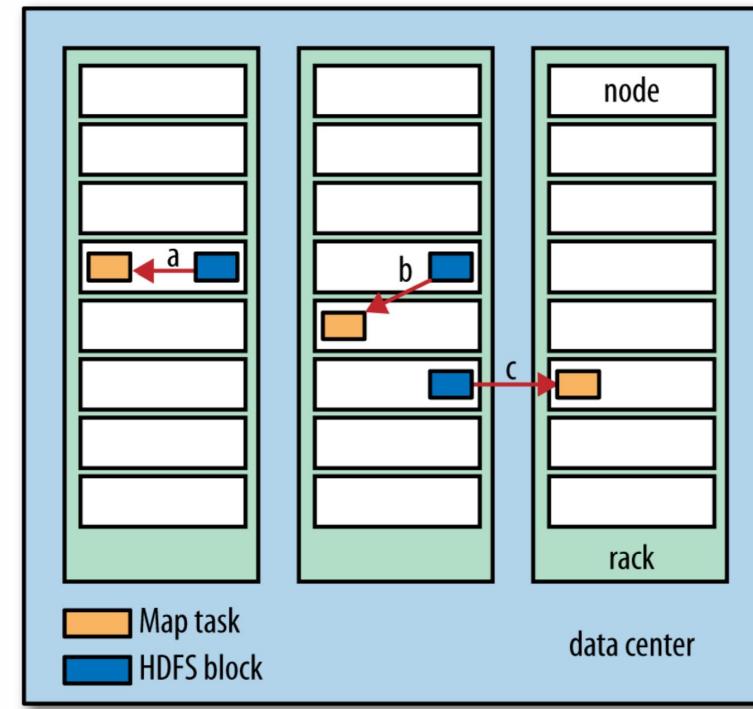
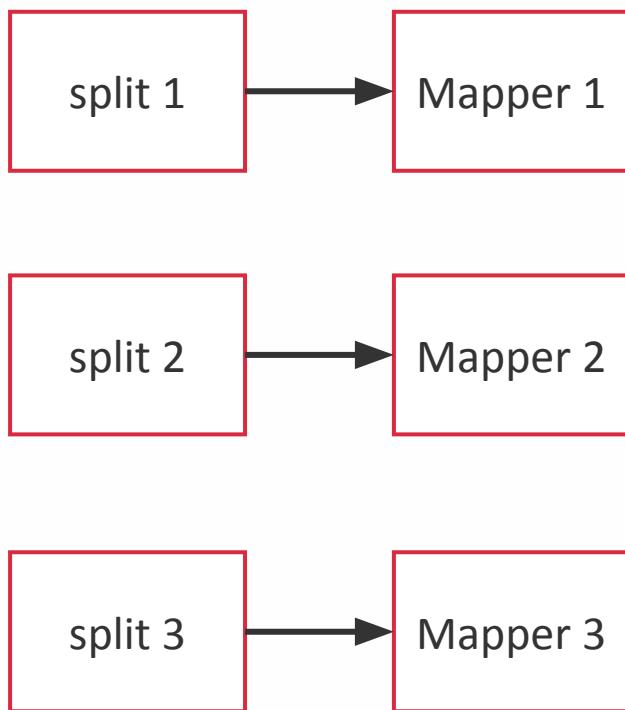
$\text{blockSize} \rightarrow \text{HDFS block size}$



# MapReduce. Input data

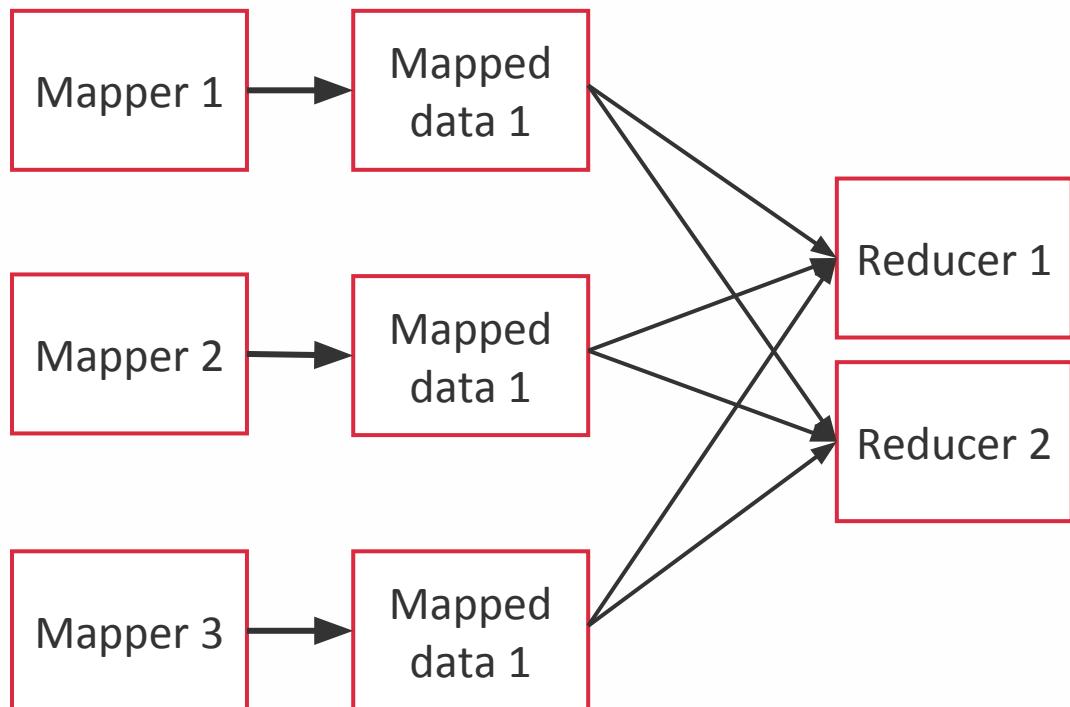


# MapReduce. Map phase



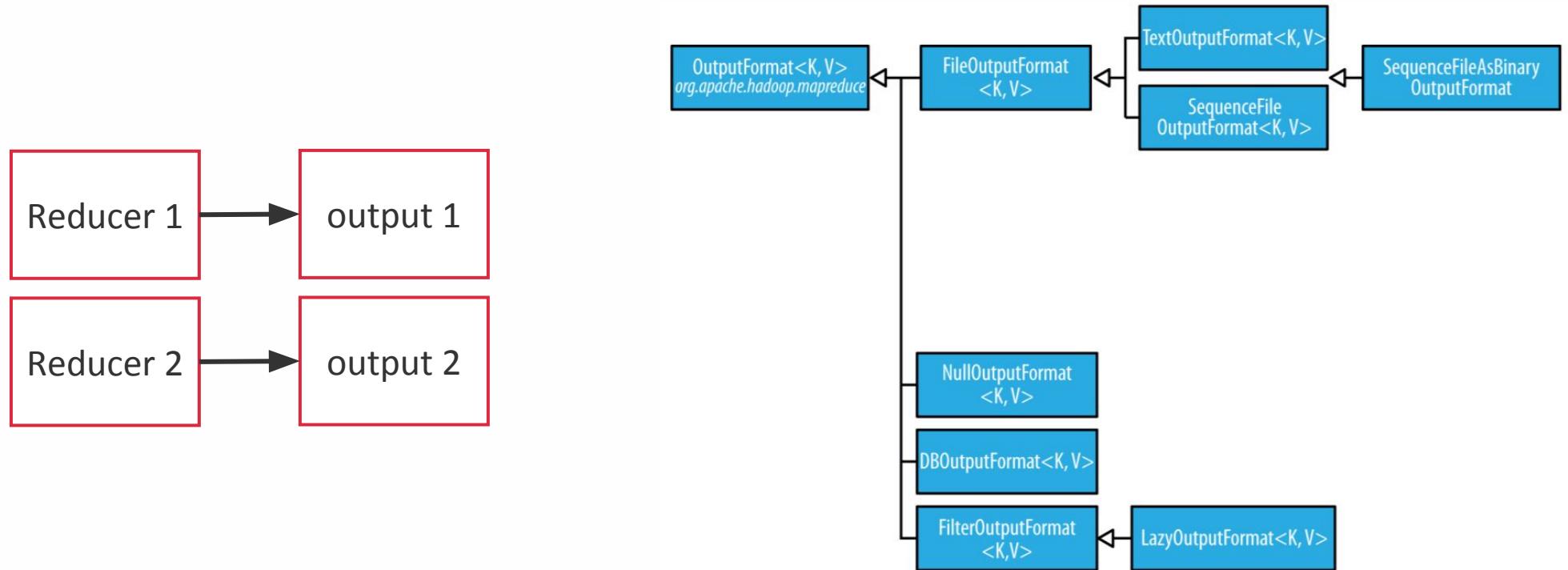
Мапперы оптимально располагать как можно ближе по data locality к самим данным

# MapReduce. Mapper results and shuffle



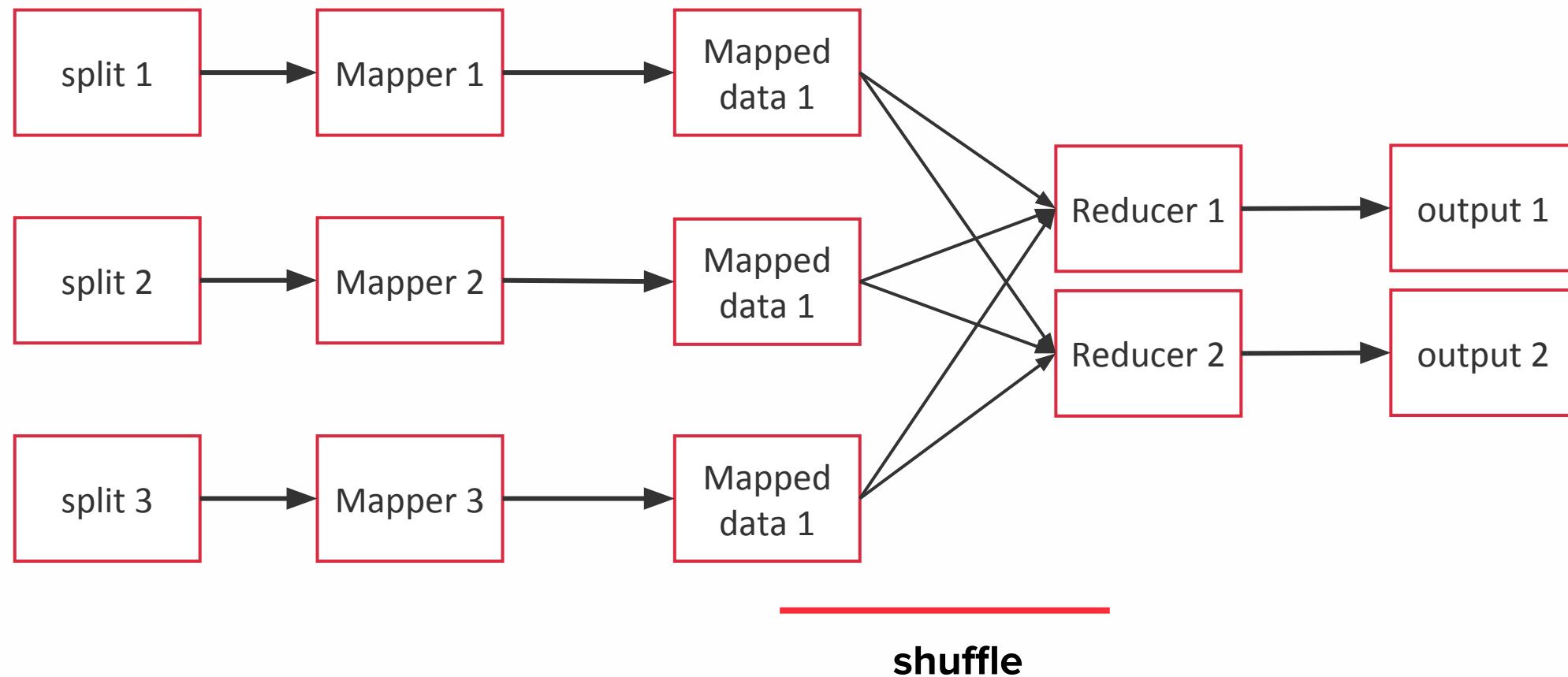
- Итоговые данные работы маппера - это key-value и записываются на локальный диск ноды.
- Для каждого редюсера маппер готовит свой файл для шаффла по ключу
- Данные с одним ключом попадают на один редьюсер

# MapReduce. Reduce



Итоговые данные работы редьюсеров сохраняются в HDFS

# MapReduce. Whole concept





# MapReduce. How to run it?

---

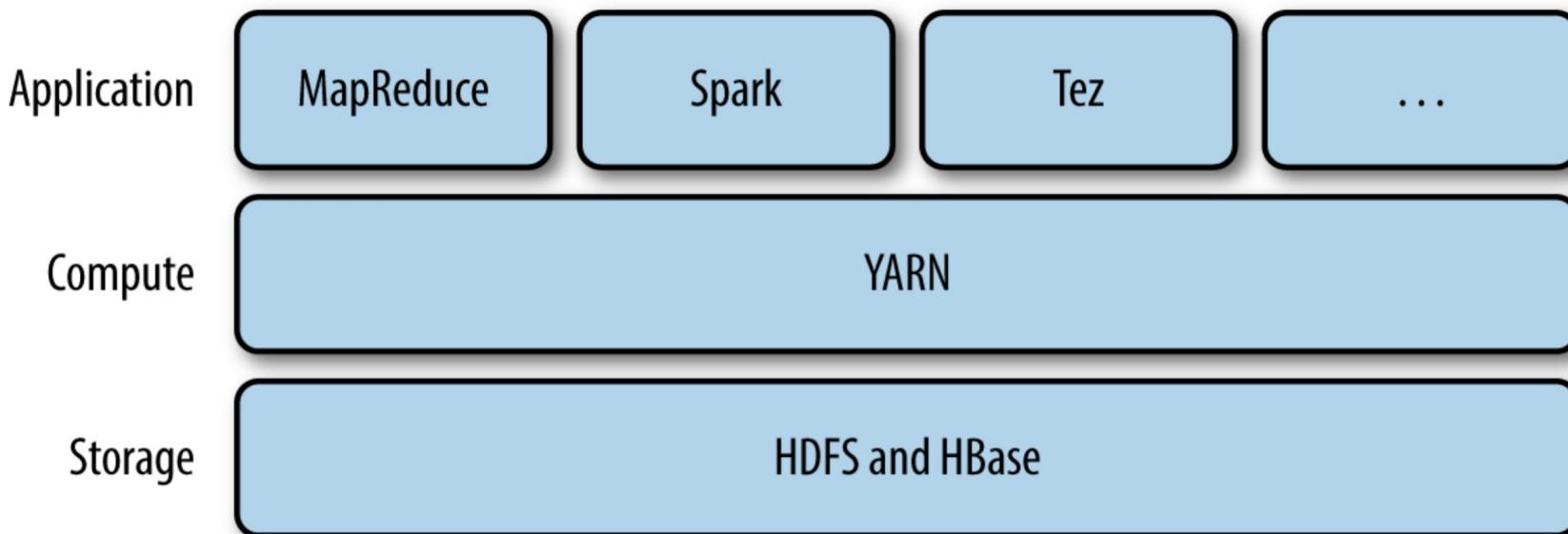
1. Как, сколько и где запустить мапперы и редьюсеры?
2. Какие ресурсы выделить для выполнения задачи?
3. Как следить за исполнением задачи?
4. Как доставить исполняемый код на воркеры?
5. Как залогировать процесс исполнения задачи?
6. Как действовать в случае отказов?

# Менеджер ресурсов YARN

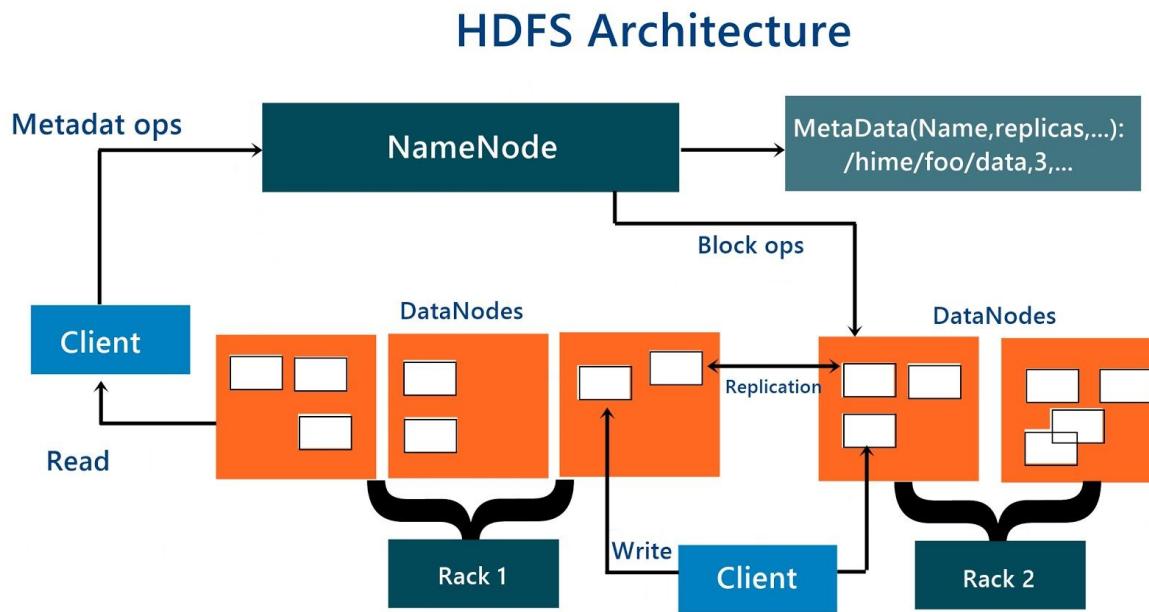


# YARN

- Управляет ресурсами кластера (CPU, GPU, RAM)
- Позволяет запускать разные типы задач на кластере



# Quick recap. Hadoop Architecture



## Namenode:

- хранит пространства имен
- хранит расположение блоков файлов
- ведет журнал изменений на диске
- SPoF

## Datanode:

- непосредственно хранит данные
- отчитывается о своем состоянии Namenode

# YARN. Architecture

## Resource Manager (RM):

- Работает на отдельном сервере
- Знает какие ресурсы есть на каждой ноде в кластере
- Распределяет воркеры по кластеру для приложений
- Реализует логику планировщика запуска

## Node Manager (NM):

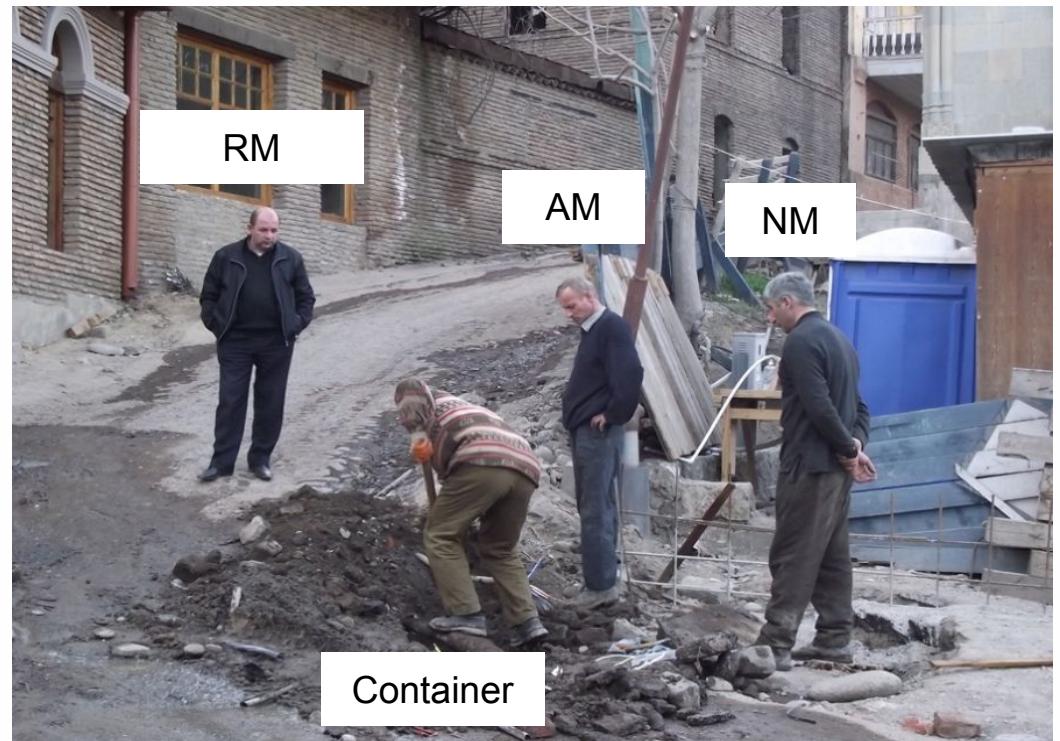
- Работает на каждой ноде
- Работает и следит за Container по команде RM

## Container:

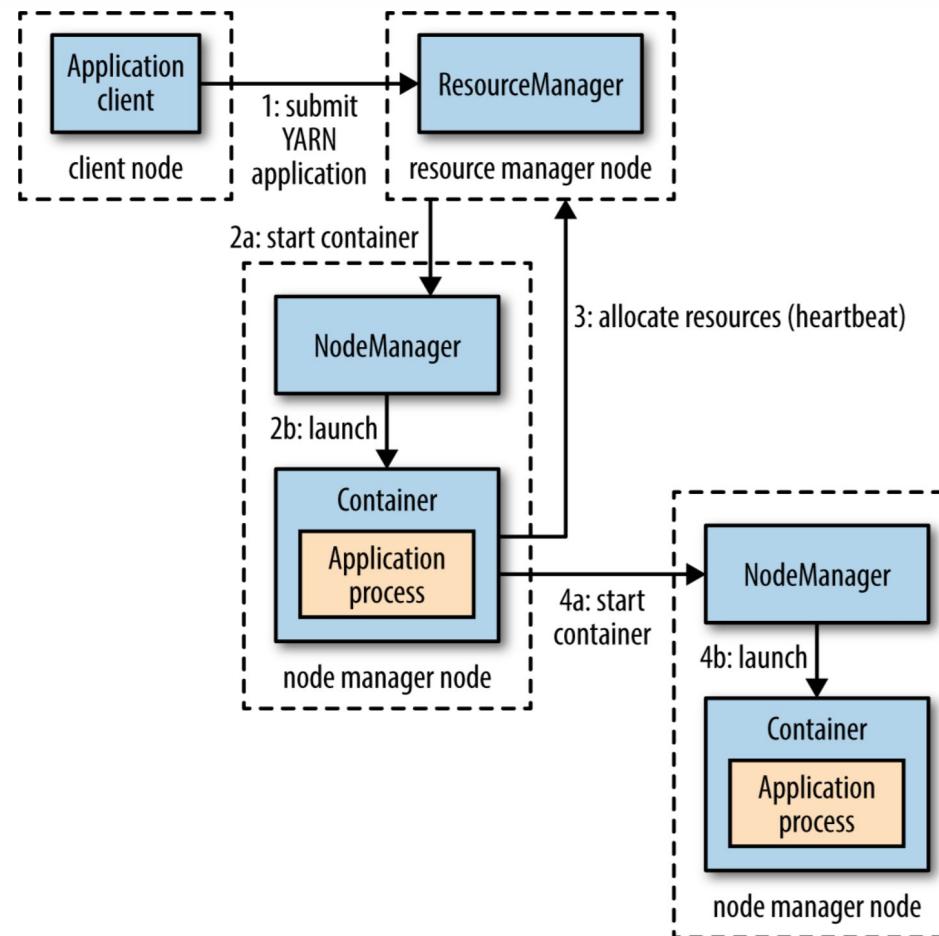
- Создается NM по команде от RM
- Создается с фиксированными ресурсами

## Application Master (AM):

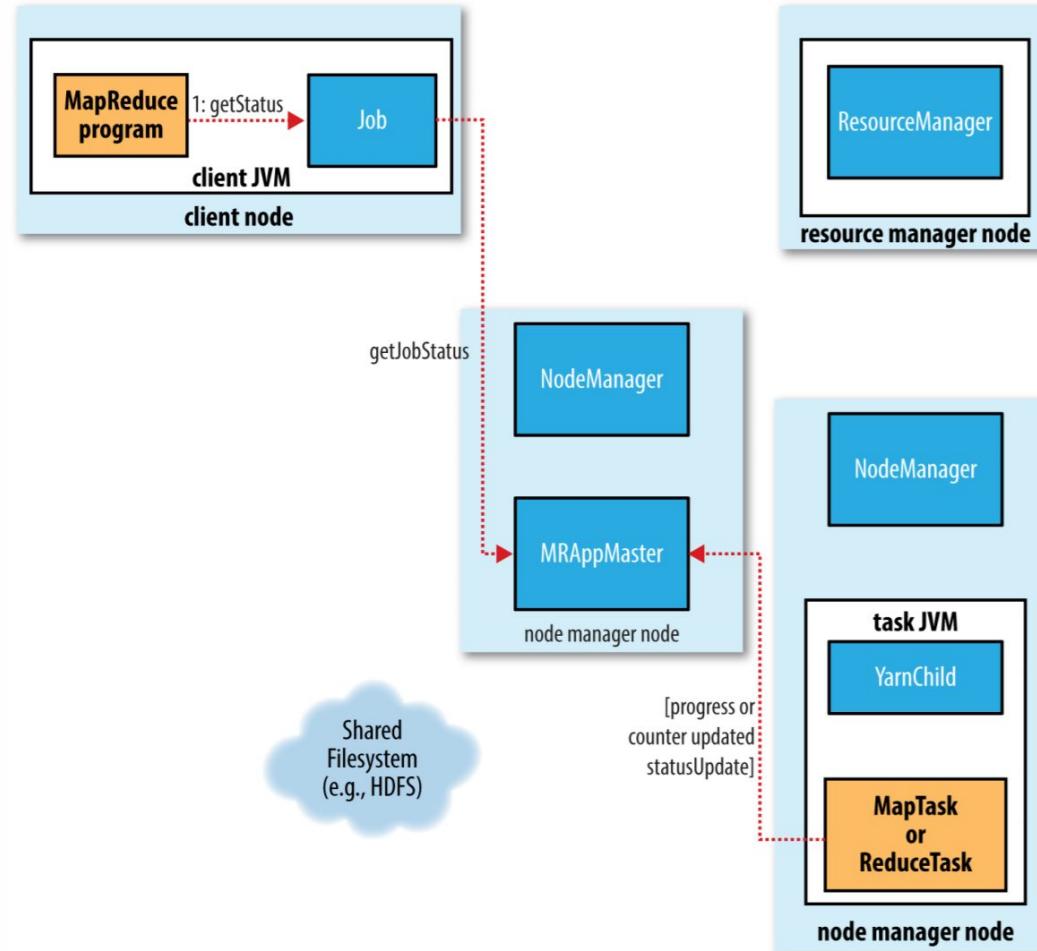
- Запускается в контейнере
- Запрашивает создание контейнеров для решения задачи у RM
- Следит за исполнением задачи и передает результаты



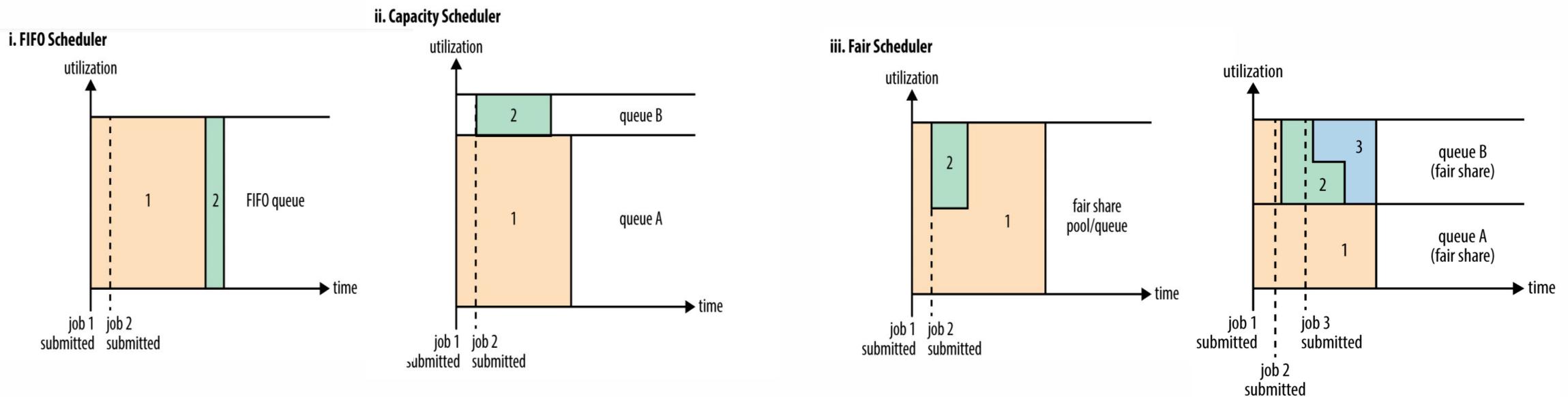
# YARN. Application workflow



# YARN. MapReduce workflow



# YARN. Tasks scheduling



# MapReduce API и Hadoop Streaming





# MapReduce API

---

## **Job**

- Пути input/output данных
- Форматы input/output данных
- Классы mapper, reducer, combiner, partitioner

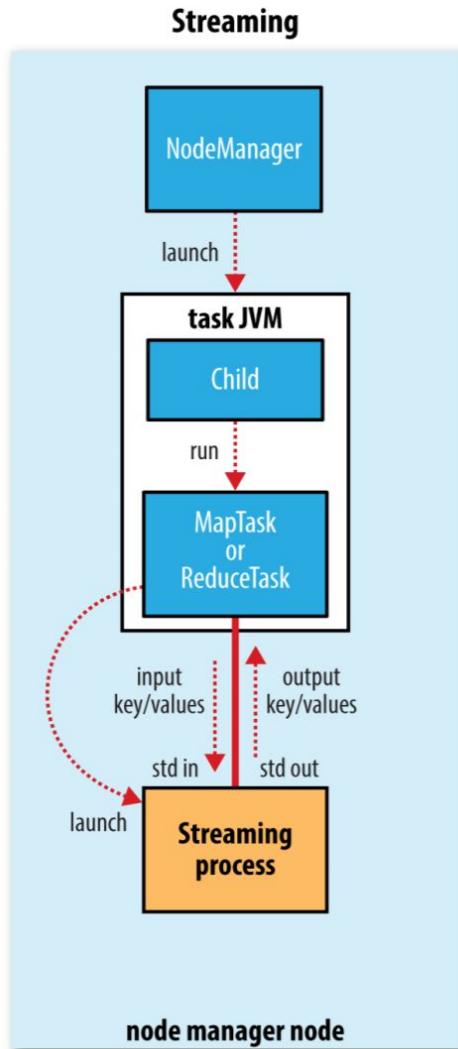
## **Mapper**

- setup
- map - для каждого key-value
- cleanup

## **Reducer**

- setup
- reduce - для каждого key
- cleanup

# Hadoop Streaming



- Пишем свой **map()**, который читает из **stdin** построчно и пишет key-value через */tab* в **stdout**
- Пишем свой **reduce()**, который читает из **stdin** отсортированный по данные по ключам и пишет key-value в **stdout**



# Lecture summary

---

1. **MapReduce** - модель распределенных вычислений. Реализовано на Java, есть возможность стриминга
2. **Map** - обработка, **Reduce** - свертка, **Shuffle** - доставка расчетов между mapper и reducer.
3. На кластере самый распространенный менеджер ресурсов - **YARN**.
4. **RM** - управляет кластером, **NM** - управляет нодой, **AM** - управляет задачей, **Container/Worker** - работает :)

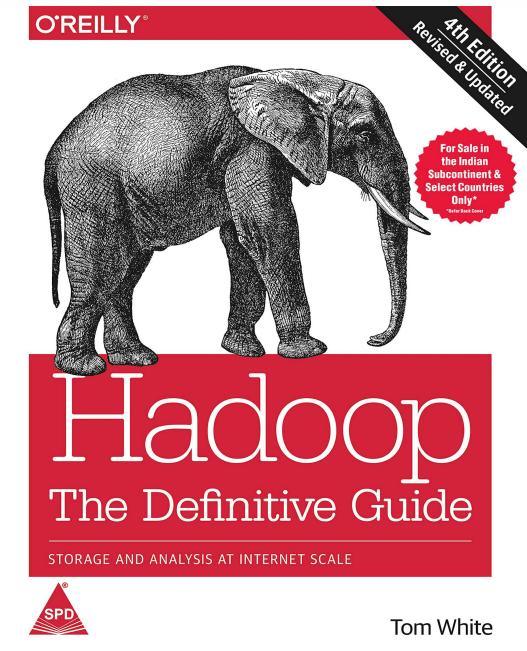


# Additional topics

---

1. Защита и контроль доступа к данным в HDFS: Kerberos, ACL.
2. Восстановление после отказов
3. Логирование задач
4. Планирование задач и построение зависимостей

# Recommended literature



**White T. Hadoop: The definitive guide**

**Java for Python Programmers**

**Bradley N. Miller. Java for Python Programmers**