

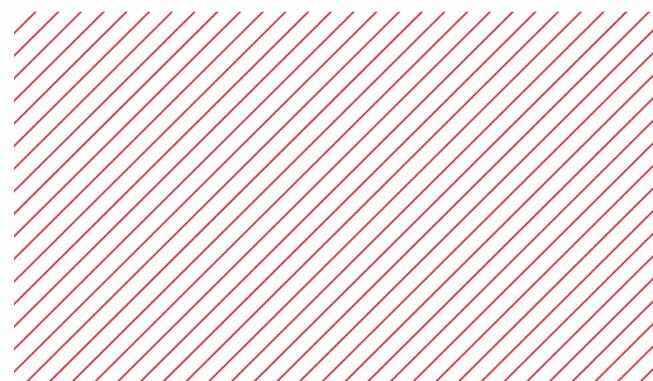
академия  
больших  
данных



# Введение в большие данные

Андрей Кузнецов

18.09.2021



# Знакомство - Андрей Кузнецов

---



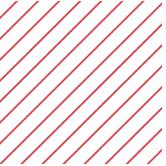
- **Machine Learning Engineer** в проекте Одноклассники, Mail.Ru Group. Занимаюсь разработкой рекомендательных систем.
- С отличием окончил Академию ФСО России
- 7 лет преподавал ИТ, ИБ и статистику
- кандидат технических наук



## Контакты:



[andrew.kuznetsov@corp.mail.ru](mailto:andrew.kuznetsov@corp.mail.ru)



# План занятия

---

1. Вводная часть: знакомство (задачи, оценки, дедлайны), подробности курса.
2. Что такое большие данные.
3. Эволюция систем хранения данных.
4. Распределенная файловая система HDFS.
5. Workshop.

# 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

# Расписание, задачи и дедлайны

---

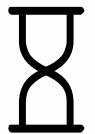


Лекция + семинар (3 астр. часа)  
**по субботам в 11-00**



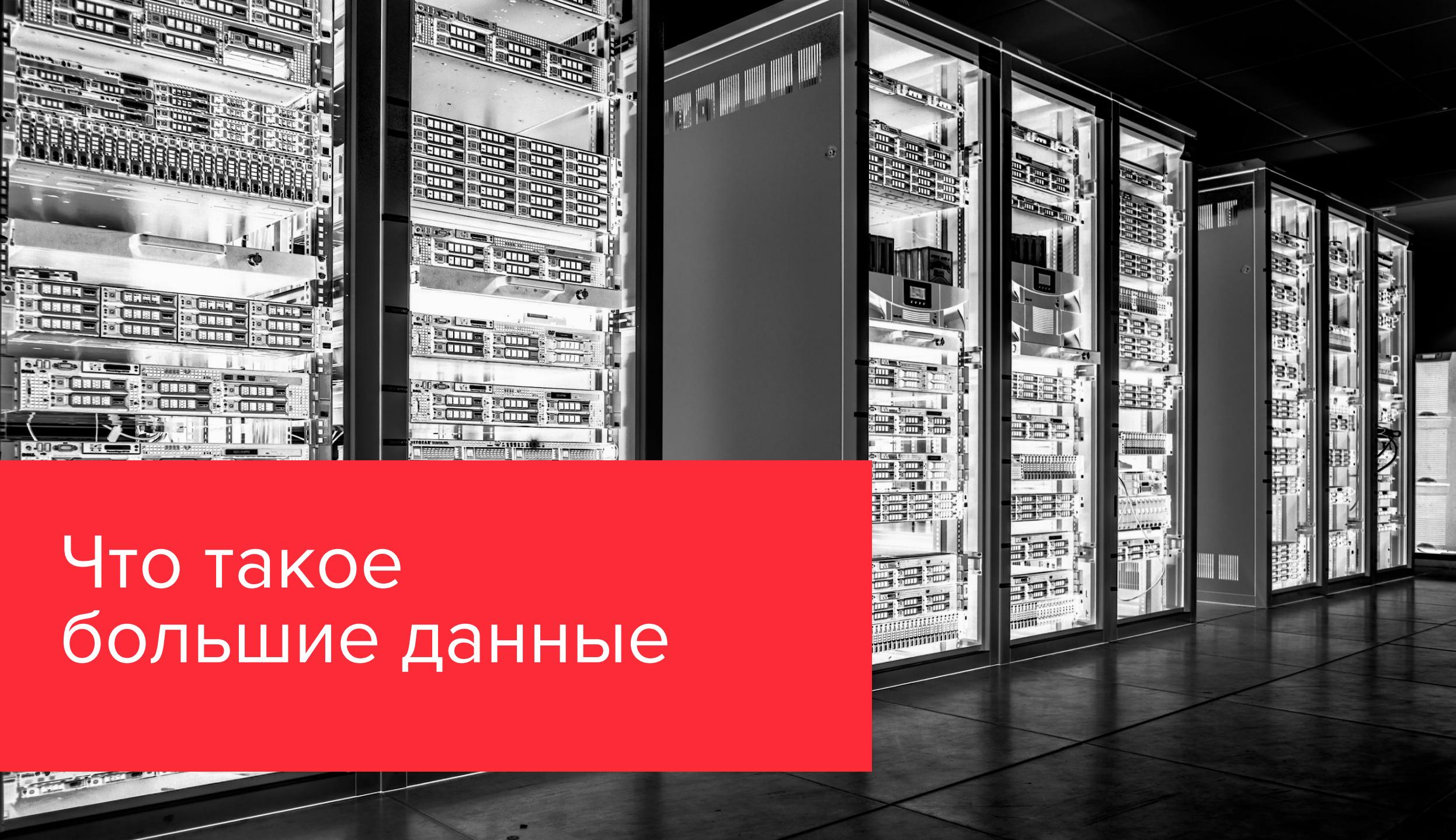
## Штрафы:

- 100% за плагиат
- 30% за сдачу в течение следующей недели после дедлайна



Дедайн сдачи домашек  
**23-59 следующей субботы**

# Что такое большие данные





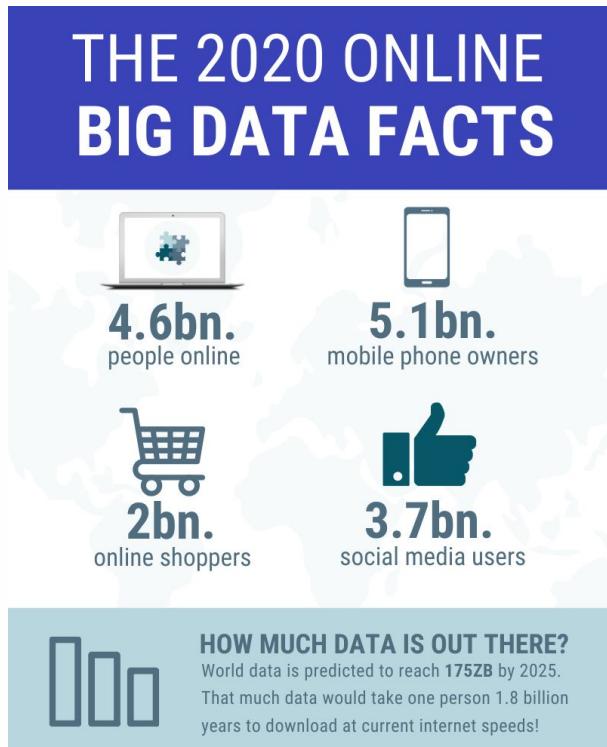
# What is Big Data?

---

**Big data** это:

- Сами массивы данных
- Инструменты
- Подходы и методы их обработки

# Big Data. Current state



## WHAT HAPPENS ONLINE EVERY MINUTE?



Year	2020	2017
Emails sent*	200 million	150 million
Google searches**	4.2 million	3.8 million
Tweets constructed**	480,000	448,800
Instagram images uploaded**	60,000	66,000
YouTube videos viewed**	4.7 million	4.2 million
Facebook new users***	400	360

Source: \*lifewire.com , \*\*internetlivestats.com , \*\*\*omnicoreagency.com

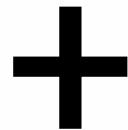
# Main Big Data generators and owners

---

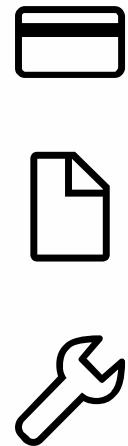
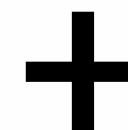


IoT

The number of IoT devices could rise to 41.6 billion by 2025.



Public Internet



Transactional Data

# Why Big Data?

---



Корпорации  
Банки  
Государства  
Люди?



# Ключевые концепты Big Data. 5V

---

1. Volume (большой объем и разные формы)
  2. Variety (разные виды данных)
  3. Velocity (время относительно доставки и обработки)
- 
4. Value (ценность данных?)
  5. Veracity (надежность + достоверность и этика?)

# Эволюция систем хранения данных

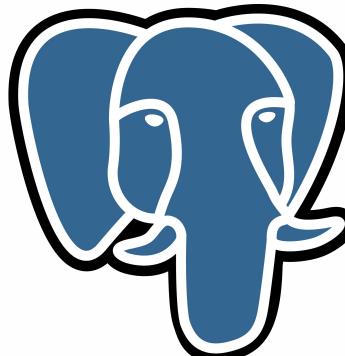


# RDBMS

---

## **RDBMS (Relational Database Management System)**

- Храним и читаем данные построчно
- Жесткая структура хранимых данных
- ACID (Atomicity — Атомарность, Consistency — Согласованность, Isolation — Изолированность, Durability — Стойкость)



# Column-oriented DBMS

---

## Column-oriented DBMS

- Храним и читаем данные в колонках
- Эффективнее работаем с большими таблицами
- Эффективное сжатие
- Медленнее пишем
- Лучше чем classic RDBMS подходит для аналитики



# NoSQL

---

**“Not Only SQL”**

Фокус на горизонтальную масштабируемость. Репликация и шардинг.

**Семейства:**

- **Key-value:** MemcacheDB, Redis, Riak, Amazon DynamoDB.
- **Column:** Apache HBase, Apache Cassandra, ScyllaDB.
- **Document:** СУБД CouchDB, Couchbase, MongoDB, Berkeley DB XML.
- **Graph:** Neo4j, OrientDB, AllegroGraph, Blazegraph, InfiniteGraph.





# NoSQL. BASE & CAP

---

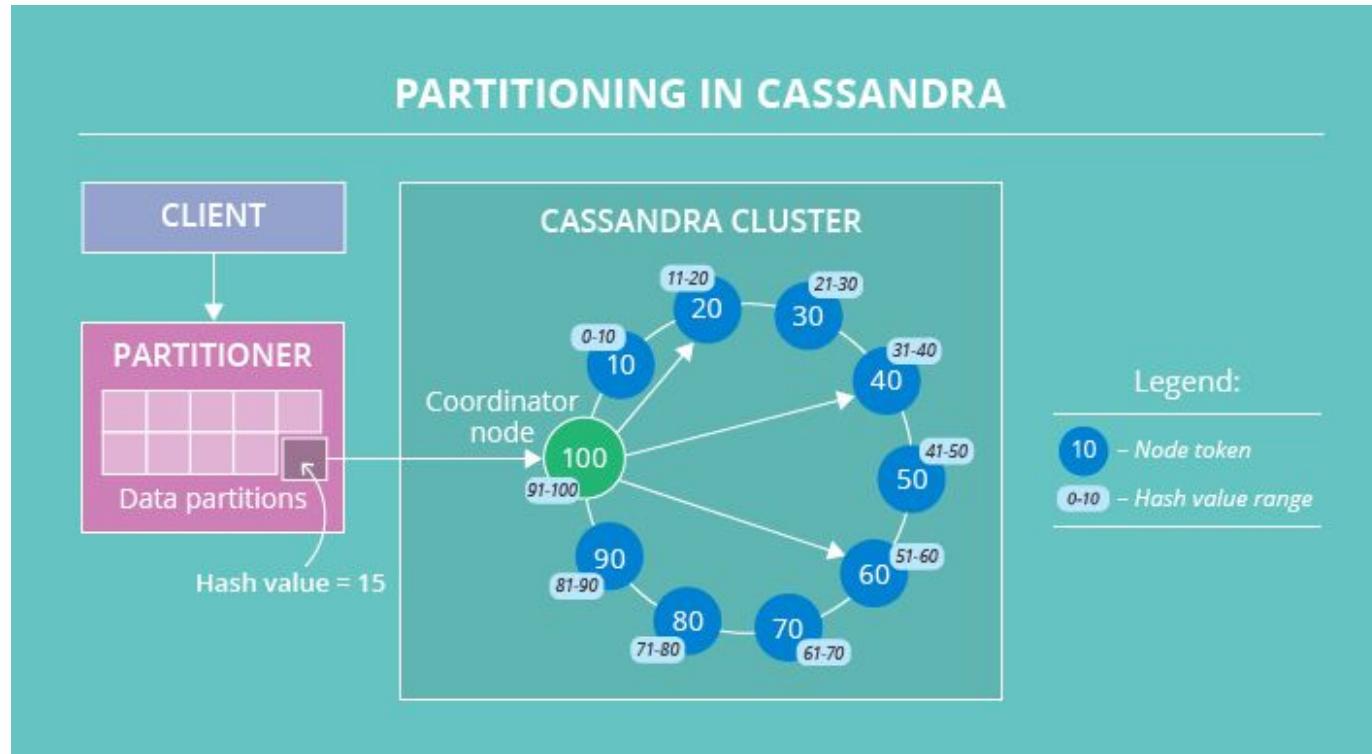
**Теорема CAP** — в любой реализации распределённых вычислений возможно обеспечить **не более двух** из трёх свойств:

- согласованность данных (**consistency**) — во всех вычислительных узлах в один момент времени данные не противоречат друг другу;
- доступность (**availability**) — любой запрос к распределённой системе завершается корректным откликом, однако без гарантии, что ответы всех узлов системы совпадают;
- устойчивость к разделению (**partition tolerance**) — расщепление распределённой системы на несколько изолированных секций не приводит к некорректности отклика от каждой из секций.

## BASE (ACID in NoSQL manner)

- базовая доступность (**basic availability**) — каждый запрос гарантированно завершается (успешно или безуспешно).
- гибкое состояние (**soft state**) — состояние системы может изменяться со временем, даже без ввода новых данных, для достижения согласования данных.
- согласованность в конечном счёте (**eventual consistency**)

# NoSQL. Apache Cassandra

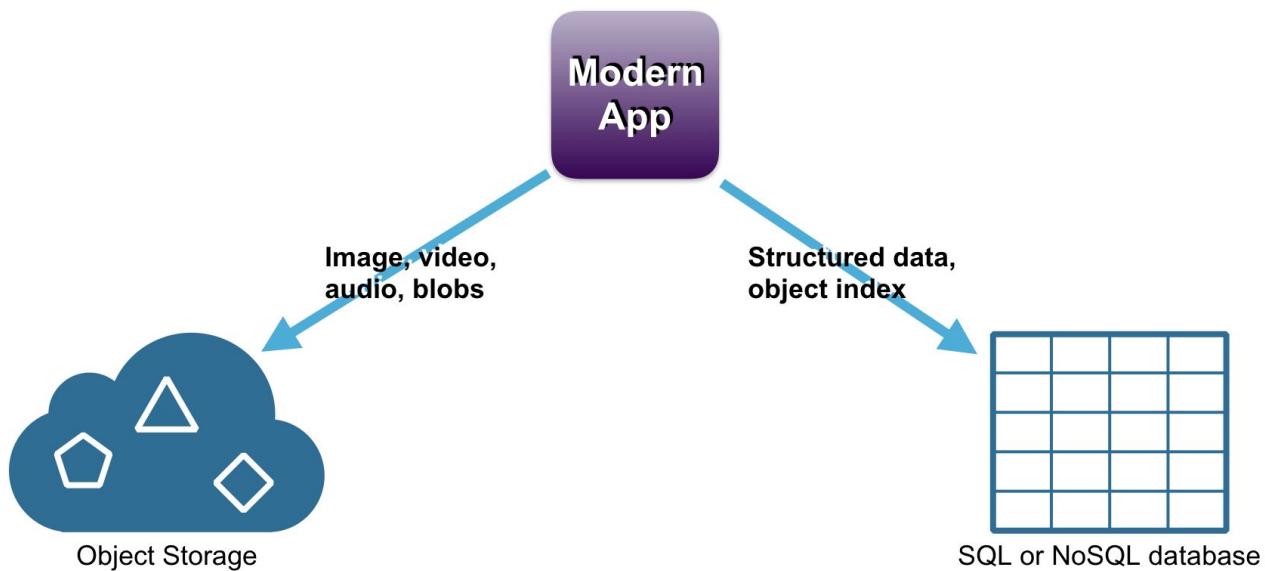


Промышленные решения на базе **Cassandra** развёрнуты у Cisco, IBM, Reddit, Apple, Twitter, Spotify и в Одноклассниках.

## Особенности:

- высокая скорость записи (около 80-360 МБ/с на узел) – данные хранятся в оперативной памяти ответственного узла, и любые обновления сперва выполняются в памяти, а только потом в файловой системе.
- гибкая масштабируемость – можно построить кластер даже на сотню узлов, способный обрабатывать петабайты данных.

# Object storage



## Особенности:

- Архитектурно неограниченный объём хранения (петабайты)
- Высокая доступность данных
- Высокая надёжность хранения данных
- Возможность многомерного масштабирования платформы: и вертикального, и горизонтального

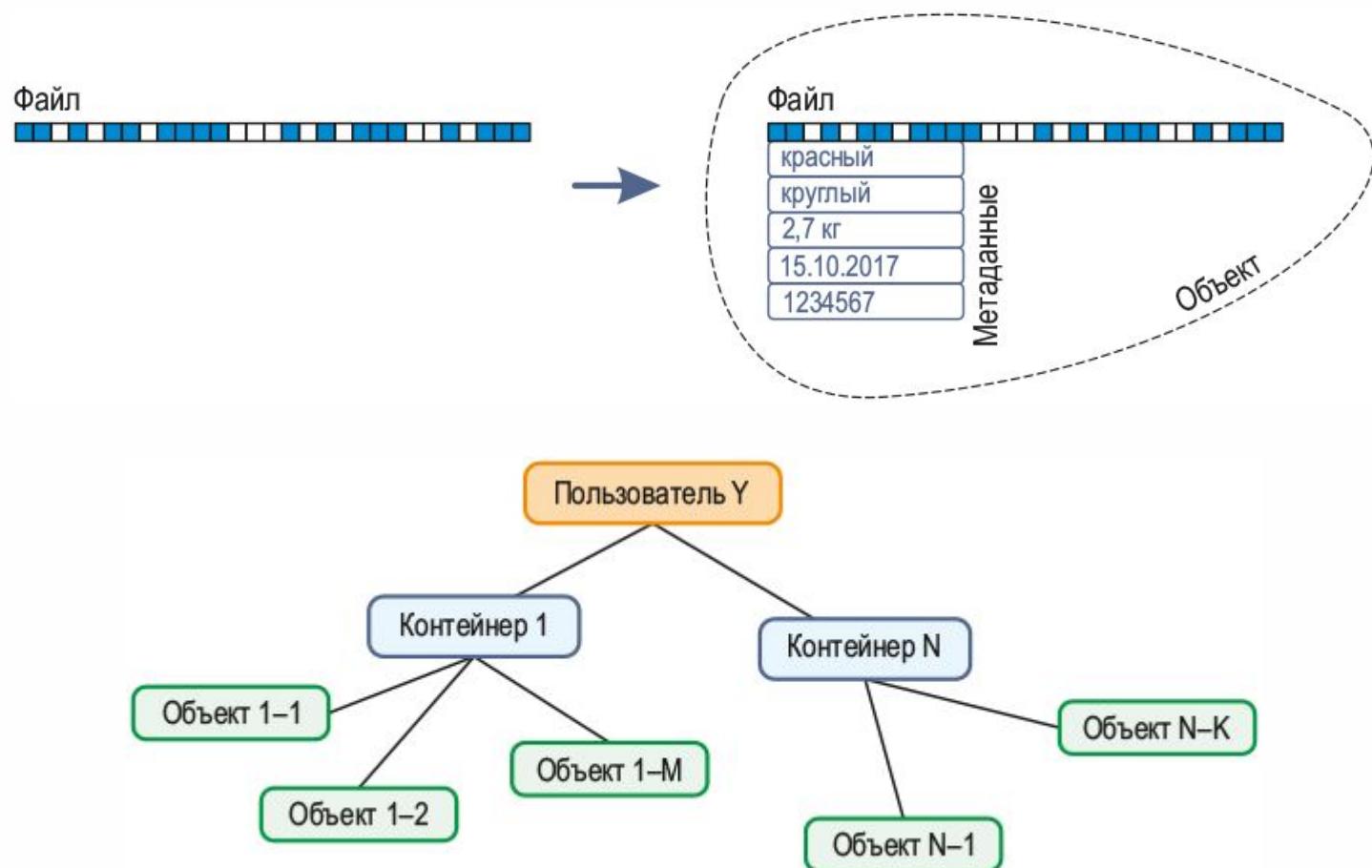
## Подходит для хранения:

- текста, фото-, видео-, аудио-
- логов/отчетов
- статических компонентов сайтов

# Object storage

## Примеры объектных хранилищ:

- Amazon S3
- Google Cloud Storage
- Azure Blob Storage
- Minio
- Ceph

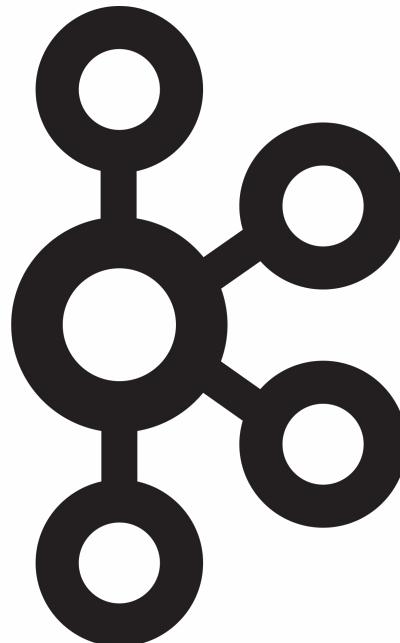


# Message broker

---

Временное хранение данных.  
Репликация и шардинг.

- Apache Kafka
- Apache Pulsar
- RabbitMQ
- Amazon Web Services (AWS)  
Kinesis



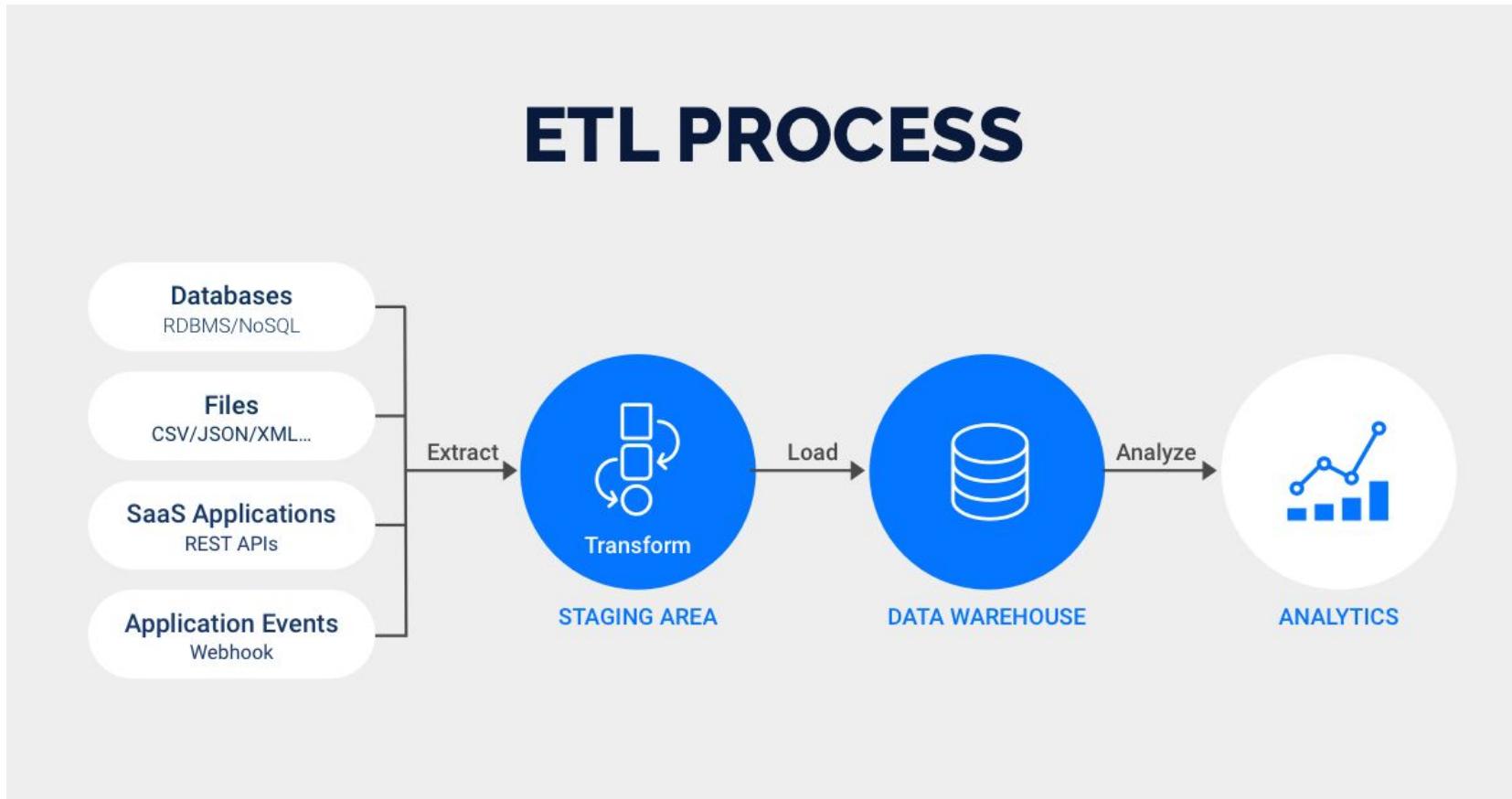
## Apache Kafka подойдет для:

- логирование поведения пользователя на сайте
- обработка потоков информации с конечных устройств IoT и IIoT («сырые данные»)
- журналирование событий

Состоит из брокеров, консьюмеров и продьюсеров.

Данные разделяются по топикам.

# ETL



Что?

Extract-Transform-Load

Зачем?

1. Аналитика
2. Машинное обучение

# Feature storage

Platform	Open-Source	Offline	Online	Metadata	Feature Engineering
<a href="#">Hopsworks</a>	AGPL-V3	Hudi/Hive	MySQL Cluster	DB Tables, Elasticsearch	(Py)Spark, Python
<a href="#">Michelangelo</a>	N/A	Hive	Cassandra	Content	Spark, DSL
<a href="#">Feast</a>	Apache V2	BigQuery	BigTable/Redis	DB Tables	Beam, Python
<a href="#">Conde Nast</a>	N/A	Kafka/Cassandra	Kafka/ Cassandra	Protocol Buffers	Shared libraries
<a href="#">Zipline</a>	N/A	Hive	KV Store	KV Entries	Flink, Spark, DSL
<a href="#">Comcast</a>	N/A	HDFS, Cassandra	Kafka / Redis	Github	Flink, Spark
<a href="#">Netflix Metaflow</a>	N/A	Kafka & S3	Kafka & Microservices	Protobufs	Spark, shared libraries
<a href="#">Twitter</a>	N/A	HDFS	Strato / Manhatten	Scala shared feature libraries	Scala DSL, Scalding, shared libraries
<a href="#">Facebook FBLearnner</a>	N/A	?	Yes, no details	Yes, no details	?
<a href="#">Pinterest Galaxy</a>	Content	S3/Hive	Yes, no details	Yes, no details	DSL (Linchpin), Spark

Что?

Хранить и готовить фичи  
для моделей

Зачем?

Переиспользование  
фичей и быстрая выкатка  
НОВЫХ моделей



# Intermediate summary

---

1. **RDMBS** хороши для приложений, но тяжело масштабировать и использовать для аналитики.
2. **Column-oriented DBMS** можно использовать для аналитики
3. **NoSQL** хорошо масштабируются и могут решать расширенный набор задач. Нет ACID, зато есть более мягкий BASE.
4. **Object storages** - важный вид хранилищ слабо структурированных данных.  
Must-have для современных приложений
5. **Message brokers** - временные хранилища слабо структурированных потоковых данных.
6. Все эти системы могут быть источником данных в схеме ETL

# Распределенная файловая система HDFS



# MPP & Distributed computing

## Параллельные вычисления:

- Shared memory
- Fail-stop
- Sync
- Scientific tasks



## Распределенные вычисления:

- Distributed memory
- Fail tolerance
- Async
- Cost / performance balance



# GFS → HDFS

## The Google File System

Sanjay Ghemawat, Howard Gobioff, and Shun-Tak Leung  
Google\*

### ABSTRACT

We have designed and implemented the Google File System, a scalable distributed file system for large distributed data-intensive applications. It provides fault tolerance while running on inexpensive commodity hardware, and it delivers high aggregate performance to a large number of clients.

While sharing many of the same goals as previous distributed file systems, our design has been driven by observations of our application workloads and technological environment, both current and anticipated, that reflect a marked departure from some earlier file system assumptions. This

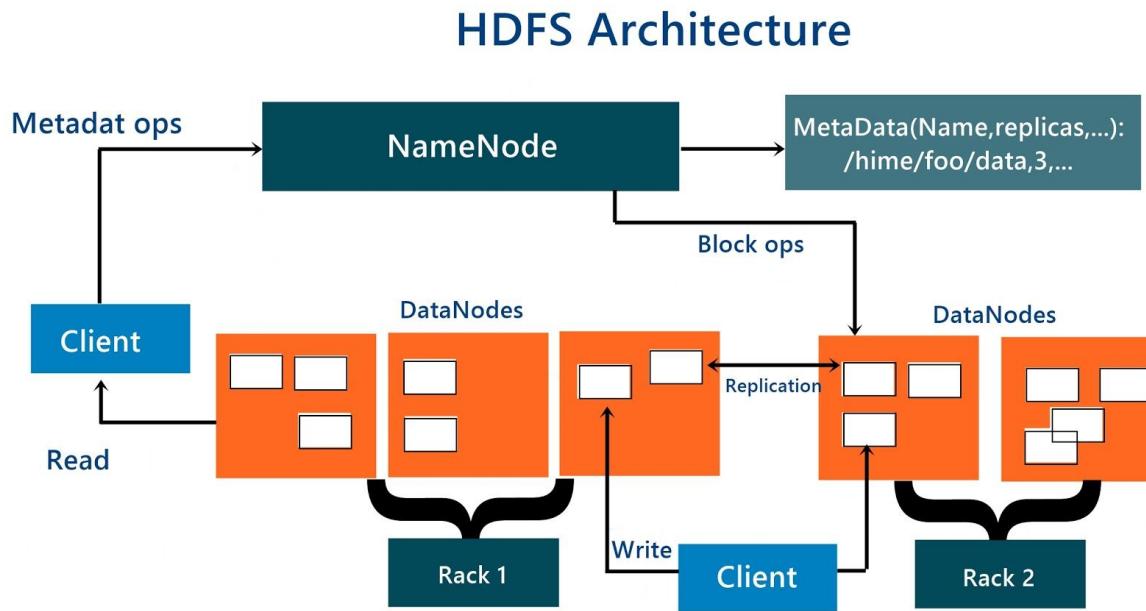
### 1. INTRODUCTION

We have designed and implemented the Google File System (GFS) to meet the rapidly growing demands of Google's data processing needs. GFS shares many of the same goals as previous distributed file systems such as performance, scalability, reliability, and availability. However, its design has been driven by key observations of our application workloads and technological environment, both current and anticipated, that reflect a marked departure from some earlier file system design assumptions. We have reexamined traditional choices and explored radically different points in the

- **2003** GFS paper published
- **2006** Hadoop first version released
- **2010** Facebook build the largest Hadoop cluster with 21 PB of storage.
- **2012** Facebook extended cluster to 100 PB
- **2013** More than 50 companies uses Hadoop



# Hadoop Architecture



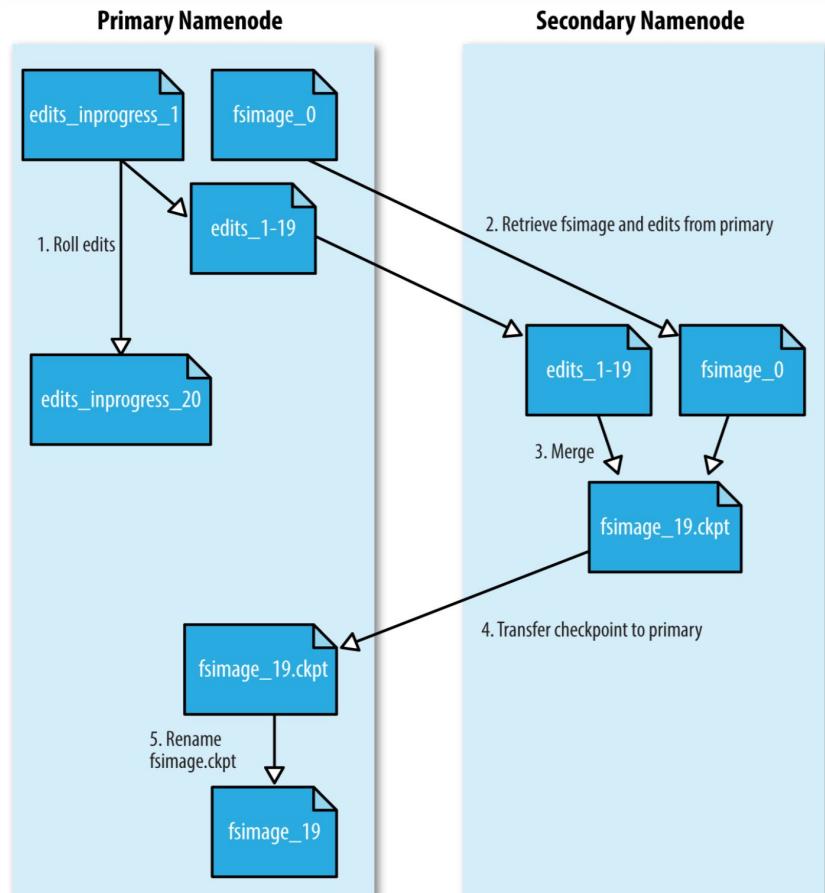
## Namenode:

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

## Datanode:

- непосредственно хранит данные
- отчитывается о своем состоянии Namenode

# Hadoop. Secondary Namenode



В обычном режиме **не дублирует**, а занимается мерджем журнала изменения с `fsimage` для быстрого восстановления Namenode после ребута.

В режиме **HA (High Availability)** может быть два режима:

1. **active** - текущая ведущая нода
2. **standby** - нода станет ведущей при выходе из строя активной

**Namenode Federation** - горизонтальное масштабирование Namenode



# Hadoop Architecture and limitations

---

**Файл в HDFS** - запись в метадате Namenode. Содержимое файла хранится в нескольких блоках одинакового размера.

**Блок** - базовая и минимальная единица памяти для HDFS. Обычно 128 Мб.

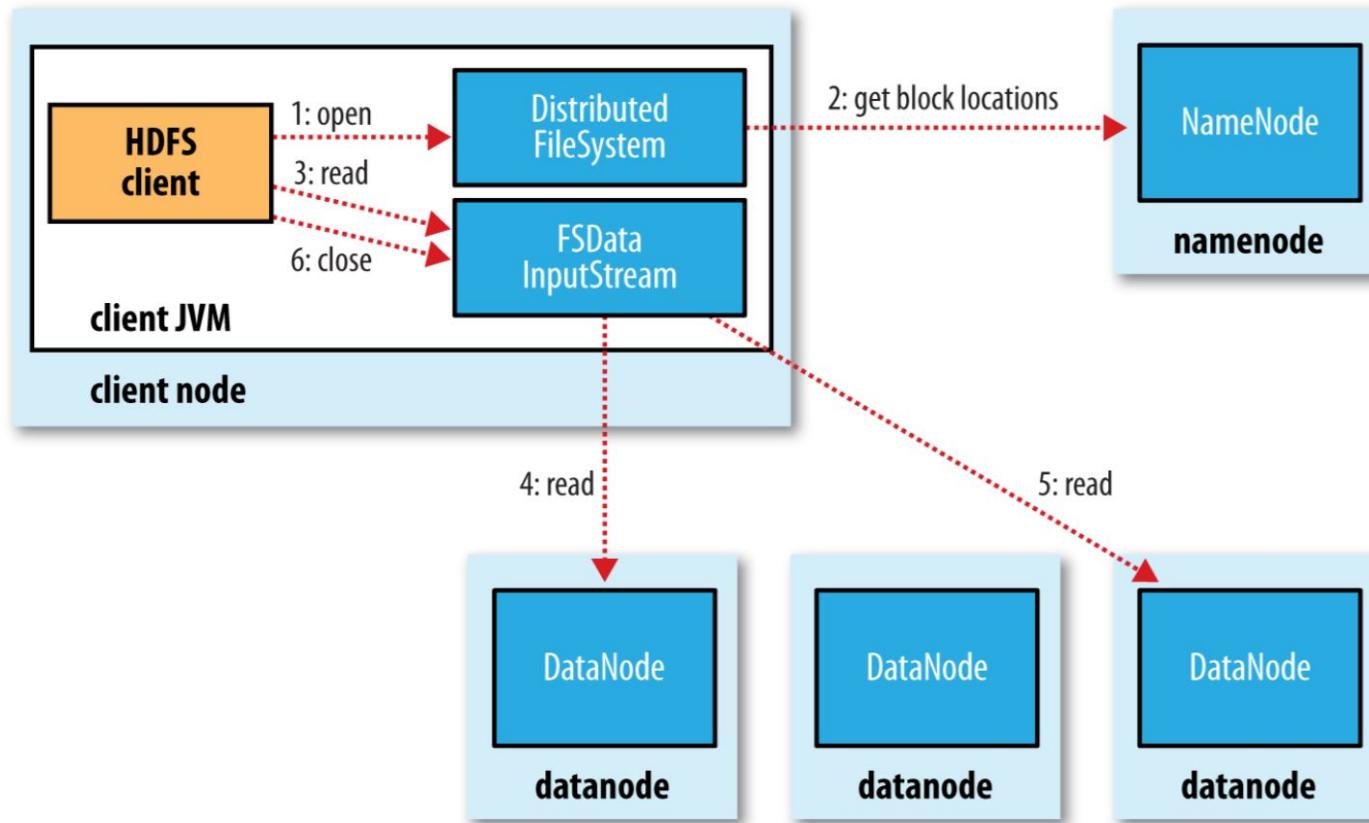
## Плюсы:

- Datanode можно запускать на слабом железе

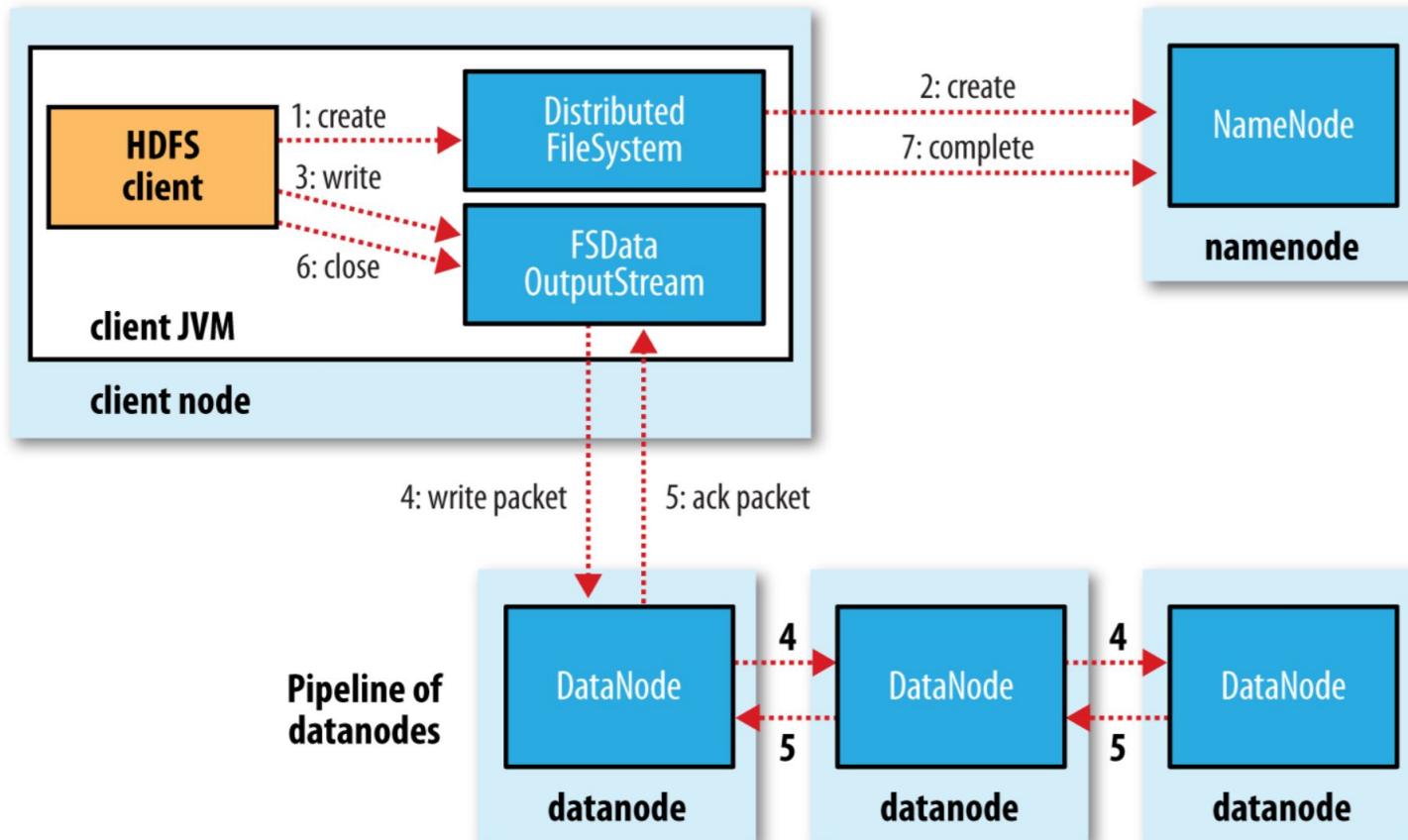
## Ограничения:

- Нельзя хранить много файлов (миллиарды уже проблема), так как описание каждого файла, директории или блока (150 байт) хранится в ОЗУ Namenode.
- Нельзя читать быстро (low-latency)
- Нельзя писать многопоточно

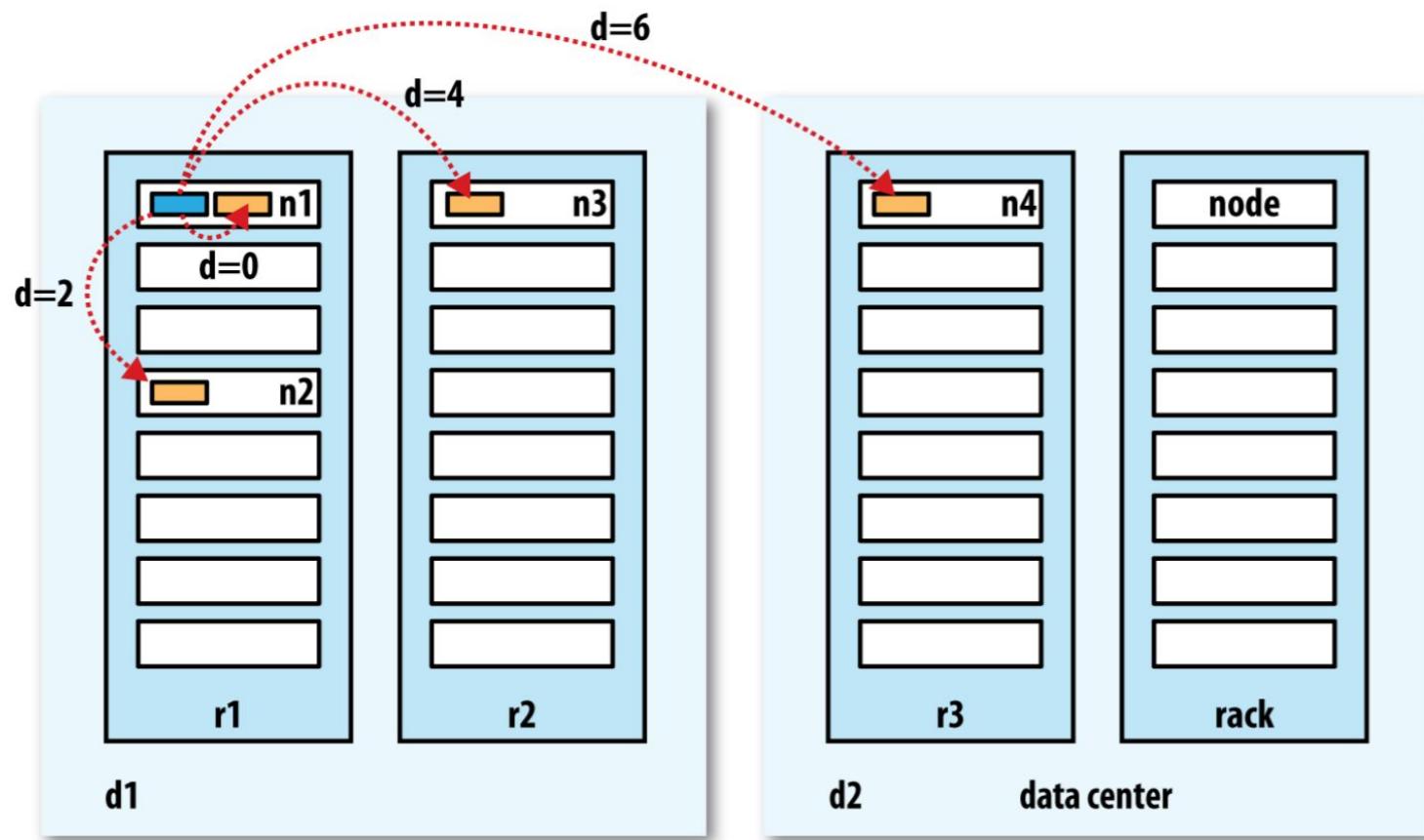
# Hadoop. Reading



# Hadoop. Writing



# Hadoop. Locality & distances

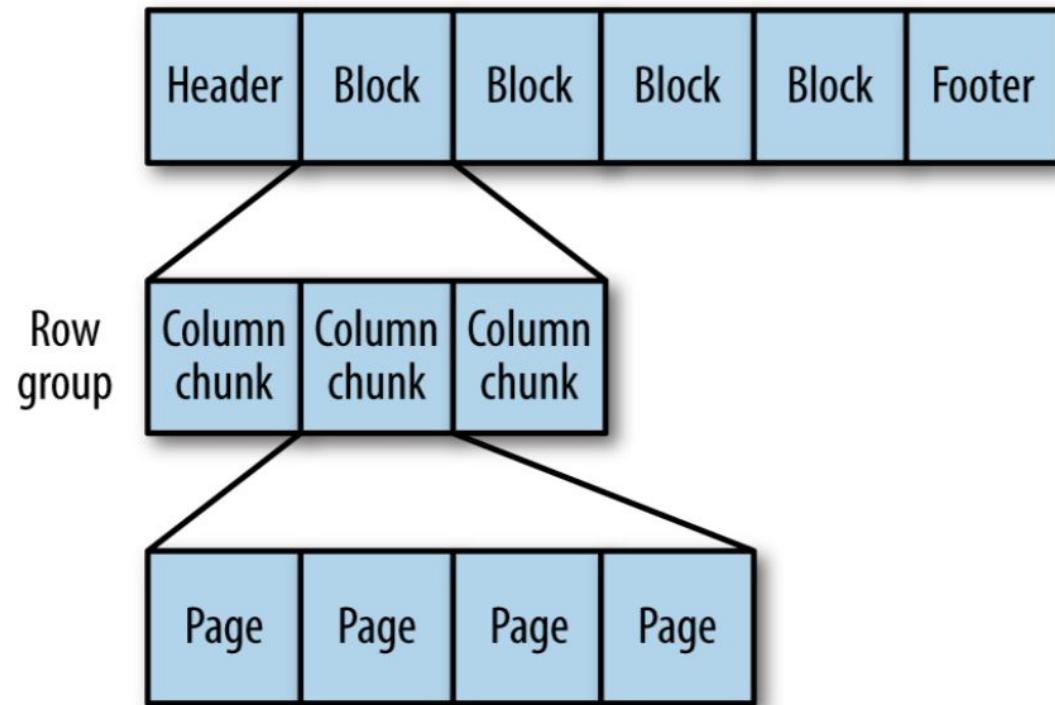


# Hadoop. Erasure coding



	Data Durability	Storage Efficiency
Single replica	0	100%
Three-way replication	2	33%
XOR with six data cells	1	86%
RS(6,3)	3	67%
RS(10,4)	4	71%

# Apache Parquet



## Features:

- Один из самых популярных форматов для записи Больших Данных
- Колоночный тип
- Поддерживает вложенность без потери эффективности
- В футер пишет схему
- Использует дельта-кодирование, run-length encoding, кодирование по словарю

# HDFS API

---

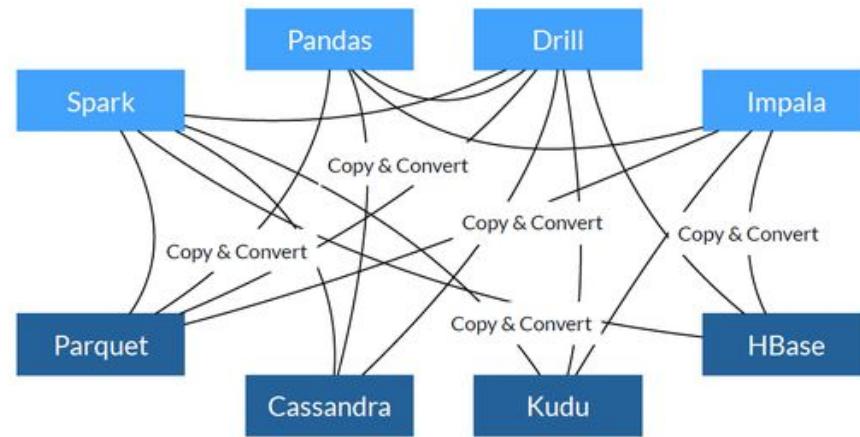
У HDFS есть несколько API:

- libhdfs - C API
- FileSystem - Java API
- WebHDFS/HttpFS - REST API
- HDFS CLI

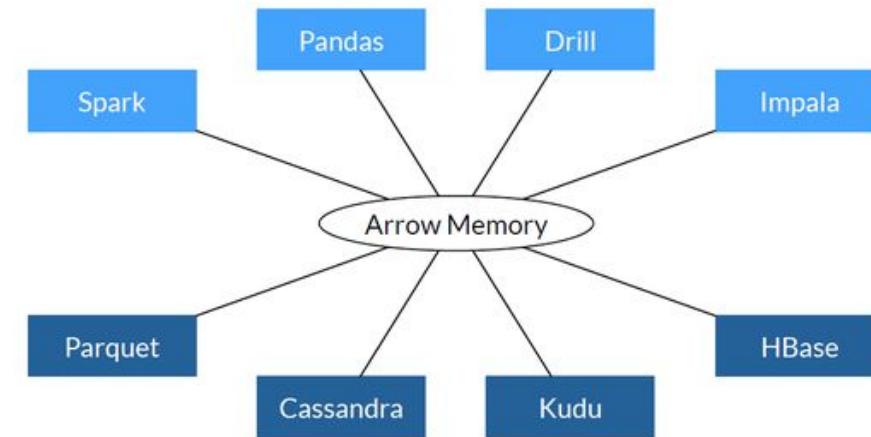


Для Python очень удобен **Apache Arrow**

# Apache Arrow



Without Arrow



With Arrow



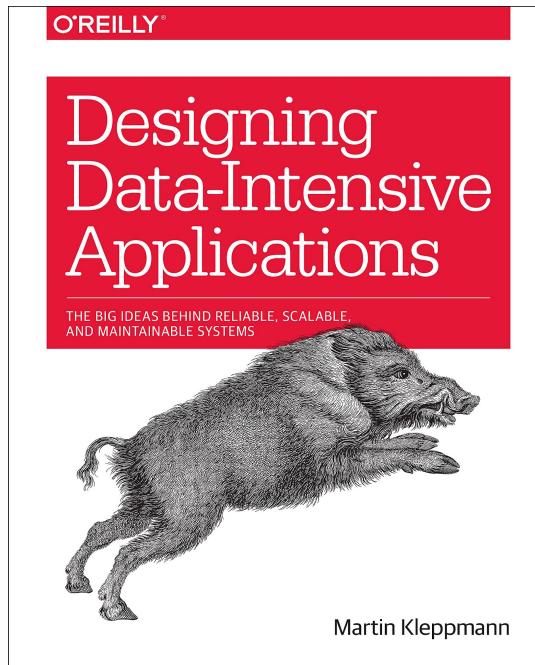
# Lecture summary

---

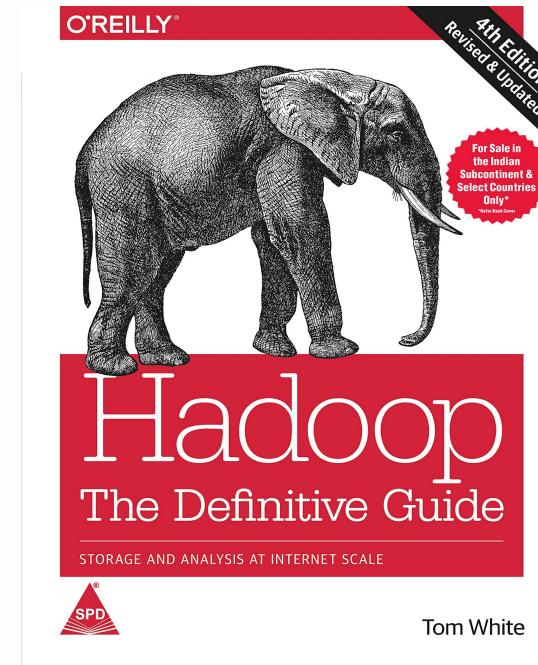
1. Big Data - это большие массивы данных, а также методы и инструменты их обработки.
2. Системы хранения RDBMS, Column DB, NoSQL, Object Storage, DFS имеют свои особенности и позволяют хранить большие данные в разных стратегиях их использования.
3. HDFS - один из главных столпов современных систем распределенных вычислений. Состоит из Namenode и Datanode, устойчив к отказам и имеет ряд ограничений.

# Recommended literature

---



**Kleppman M. Designing  
Data-Intensive Applications: The Big  
Ideas Behind Reliable, Scalable, and  
Maintainable Systems**



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