

# **Введение в Big Data**

**Как работают и где находятся большие данные?**

**Лекция 1**

**Кирилл Сысоев**

# Откуда пришла Big Data

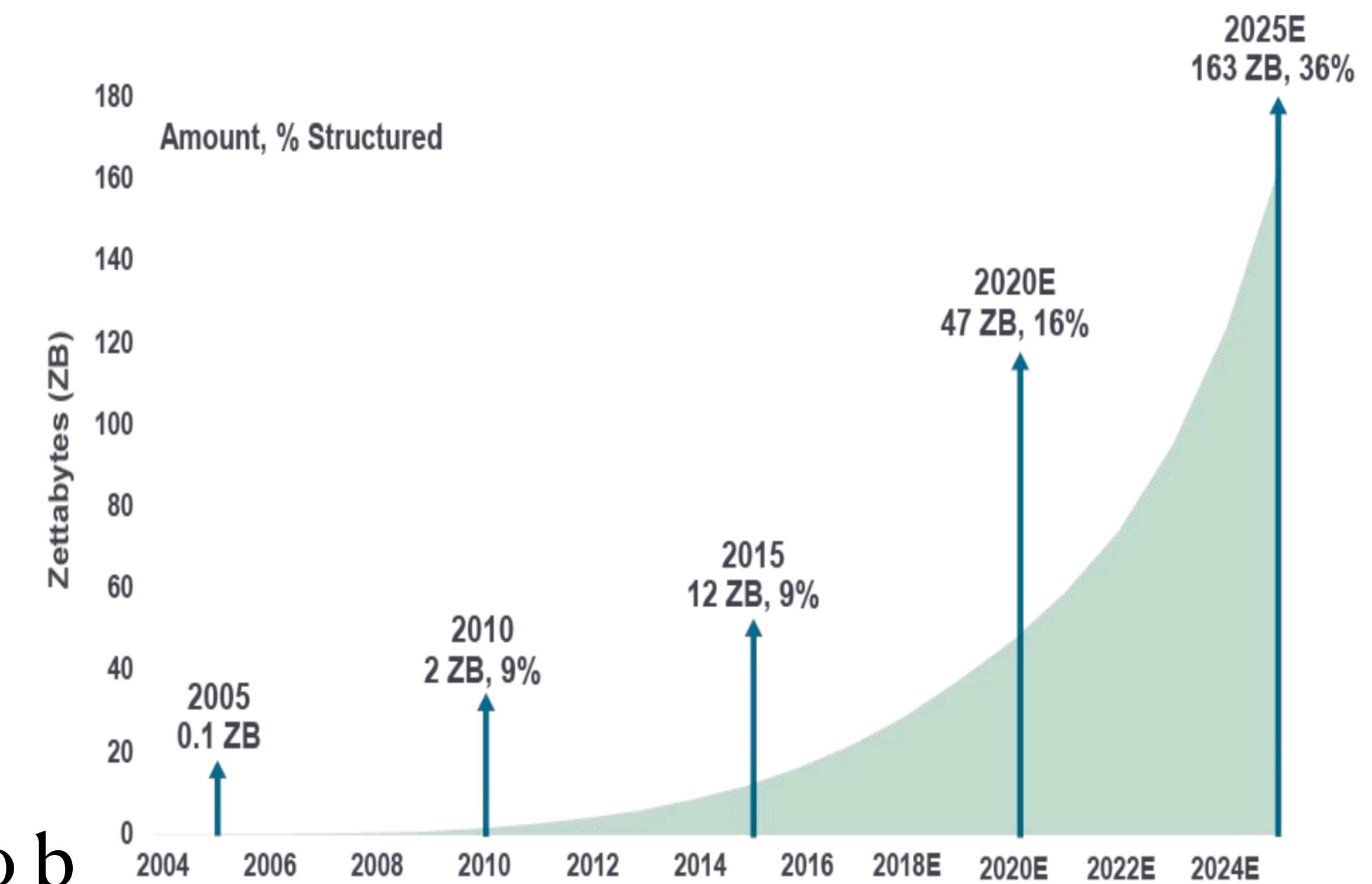
Сфера:

- Телеком
- Банки
- Социальные сети
- Медиа
- Промышленность
- Биоинформатика
- Интернет вещей



# Рост данных

1 kilobyte	1 000 b
1 megabyte	1 000 000 b
1 gigabyte	1 000 000 000 b
1 terabyte	1 000 000 000 000 b
1 petabyte	1 000 000 000 000 000 b
1 exabyte	1 000 000 000 000 000 000 b
1 zettabyte	1 000 000 000 000 000 000 000 b



# Количество обрабатываемых данных



Обрабатывает ежедневно 40 exabyte



Содержит 300 petabyte пользовательских данных



Хранит 160 zettabyte

# Виды данных

## Структурированные данные

Данные, организованные по **заранее определённой модели или схеме**, обычно представленные в виде таблиц с фиксированными полями и типами данных. Такие данные легко вводить, хранить, искать и анализировать с помощью реляционных баз данных и SQL-запросов

## Полуструктурированные данные

Данные, которые **не соответствуют жёсткой структуре** таблиц, но содержат определённые метки или теги для разделения элементов и иерархии. Примеры включают XML, JSON и HTML файлы. Эти данные имеют некоторую организацию, что облегчает их парсинг и анализ.

## Неструктурированные данные

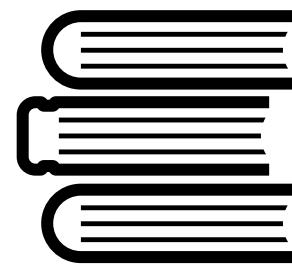
Данные, которые **не имеют предопределённой модели или схемы**. Они могут быть в различных форматах, таких как текстовые документы, изображения, аудио и видео файлы. Анализ таких данных требует специальных методов обработки, включая обработку естественного языка и машинное обучение.

# Что такое Big Data

Big Data — это термин, который используется для описания **очень больших объемов данных**, которые **сложно или невозможно обрабатывать традиционными методами и инструментами.**

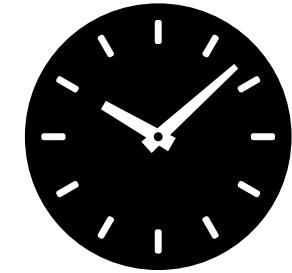


# Характеристики Big Data (5V)



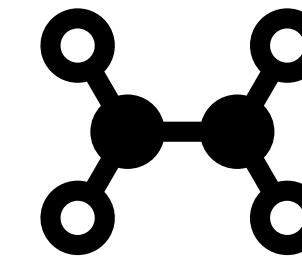
## Volume (Объем)

- Петабайты
- Файлы/Таблицы



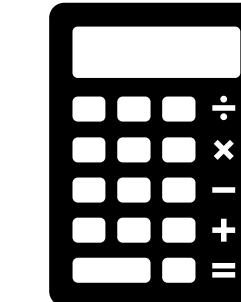
## Velocity (Скорость)

- Высокая частота генерации
- Онлайн и оффлайн данные



## Variety (Разнообразие)

- Структурированные
- Неструктурированные
- Полуструктурированный

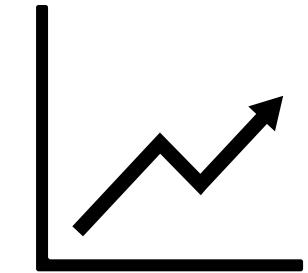


## Value (Ценностъ)

- Не все данные полезные

## Veracity (Достоверность)

- Качество
- Точность



# Применение Big Data

## Технологии и инструменты для обработки Big Data:

1. Apache Hadoop: Одна из первых и наиболее известных технологий для распределенной обработки больших данных с использованием параллельных вычислений.
2. Apache Spark: Быстрее Hadoop и используется для распределенной обработки данных в памяти, что делает его популярным для аналитики в реальном времени.
3. NoSQL базы данных: СУБД, такие как MongoDB, Cassandra, которые хорошо подходят для работы с неструктуризованными данными.
4. Apache Kafka: Платформа для обработки данных в реальном времени, позволяющая собирать, хранить и обрабатывать потоки событий.
5. Elasticsearch: Мощная поисковая система, которая может обрабатывать и индексировать большие объемы данных.

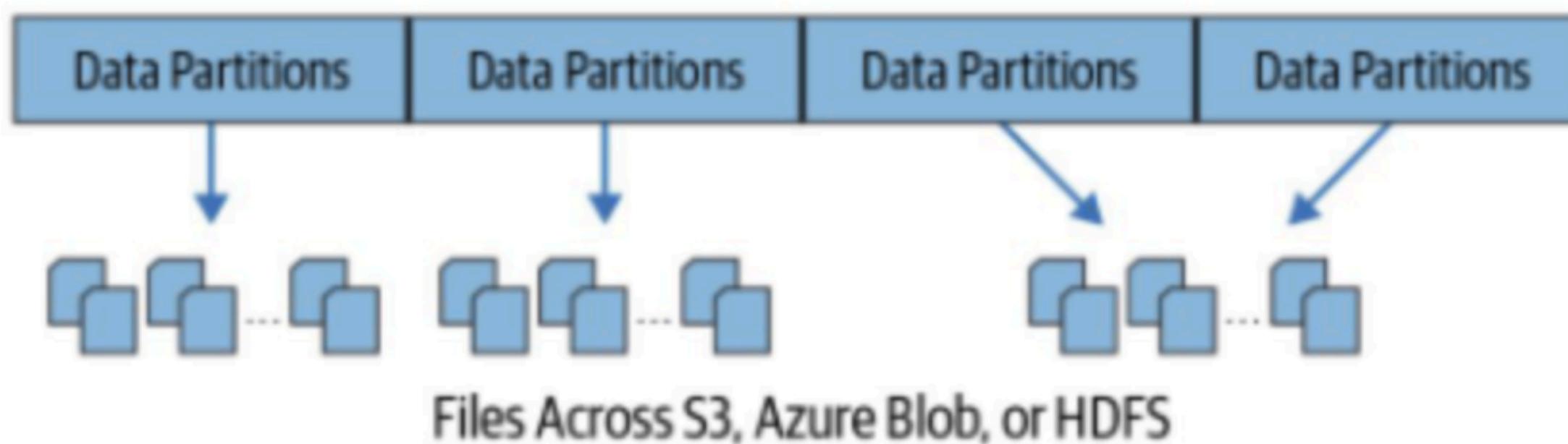
## Применение Big Data:

1. Бизнес-аналитика и маркетинг: Использование данных для понимания поведения клиентов, анализа трендов, персонализации предложений и оптимизации маркетинговых стратегий.
2. Интернет и социальные сети: Анализ пользовательской активности, социальных трендов и предпочтений для улучшения пользовательского опыта.
3. Медицина: Анализ геномных данных, медицинских изображений, данных пациентов для улучшения диагностики и персонализированной медицины.
4. Финансовый сектор: Анализ транзакций для выявления мошенничества, прогнозирования рисков и улучшения услуг для клиентов.
5. Производство и логистика: Прогнозирование поломок оборудования, оптимизация цепочек поставок, управление производственными процессами.

# BigData

- не влезают на одну машину

Распределенное хранилище

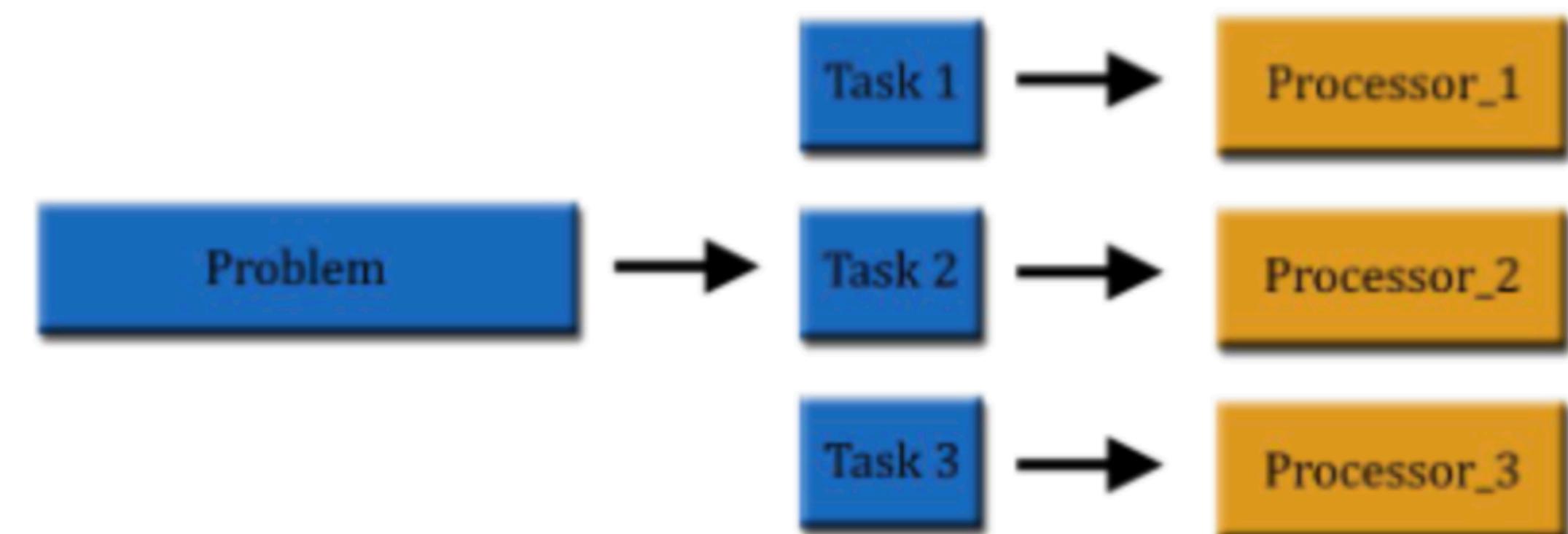


Параллельные вычисления

Serial Computing



Parallel Computing



# **2 ключевых вопроса в Big Data**

- 1. Как хранить большие данные? (Storage Layer)**
- 2. Как обрабатывать большие данные? (Processing layer)**
- 3. ... (Как управлять ресурсами кластера – Resource management layer)**

**Сначала ознакомимся с несколькими ключевыми терминами в BigData**

# OLTP VS OLAP

## Online Transactional Processing

## Online Analytical Processing

Критерии	OLAP	OLTP
Цель	OLAP помогает анализировать большие объемы данных для обеспечения поддержки принятия решений.	OLTP помогает управлять транзакциями в реальном времени и обрабатывать их.
Источник данных	OLAP использует исторические и объединенные данные из нескольких источников.	OLTP использует транзакционные данные в реальном времени из одного источника.
Структура данных	OLAP использует многомерные (в формате кубов) или реляционные базы данных.	OLTP использует реляционные базы данных.
Модель данных	OLAP использует схему «звезда», «снежинка» или другие аналитические модели.	В OLTP используются нормализованные или денормализованные модели.
Объем данных	OLAP предъявляет большие требования к хранению данных. Используйте терабайты (ТБ) и петабайты (ПБ).	OLTP предъявляет сравнительно небольшие требования к хранению данных. Используйте гигабайты (ГБ).
Время ответа	Время отклика OLAP больше, обычно оно исчисляется секундами или минутами.	Время отклика OLTP короче, обычно исчисляется миллисекундами.
Образцы приложений	OLAP хорошо подходит для анализа тенденций, прогнозирования поведения клиентов и определения прибыльности.	OLTP подходит для обработки платежей, заказов и управления данными клиентов.

# OLTP vs OLAP

--Еще примерно 150+ строк кода

...

## Terrifying OLAP

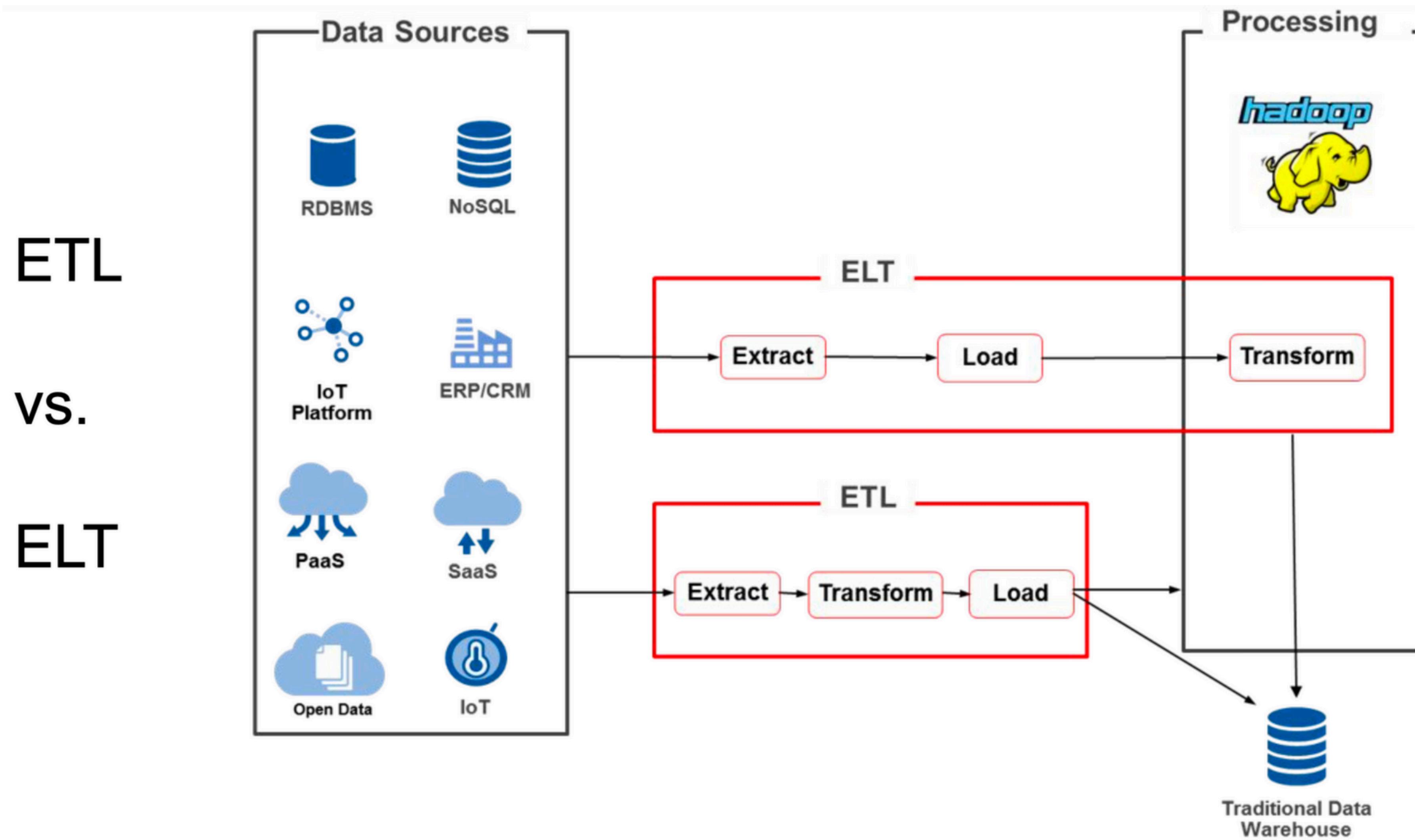
```
WITH EV AS
(
    SELECT
        e.*,
        LAG(IFNULL(e.SESSION_ID, NULL), 1, ARRAY_CONSTRUCT(e.SESSION_ID, e.EVENT_DTM)) IGNORE NULLS
        OVER(
            PARTITION BY e.CLIENT_ID
            ORDER BY e.EVENT_DTM ASC, e.EVENT_ID ASC
        ) AS PREV_SESSION_ROW,
        LEAD(IFNULL(e.SESSION_ID, NULL), 1, ARRAY_CONSTRUCT(e.SESSION_ID, e.EVENT_DTM)) IGNORE NULLS
        OVER(
            PARTITION BY e.CLIENT_ID
            ORDER BY e.EVENT_DTM ASC, e.EVENT_ID ASC
        ) AS NEXT_SESSION_ROW,
        PREV_SESSION_ROW[1] AS PREV_EVENT_DTM,
        NEXT_SESSION_ROW[1] AS NEXT_EVENT_DTM,
        IFF(PREV_EVENT_DTM >= DATEADD(hour, -2, e.EVENT_DTM), PREV_SESSION_ROW[0], NULL) AS PREV_SESSION_ID,
        IFF(NEXT_EVENT_DTM < DATEADD(hour, 2, e.EVENT_DTM), NEXT_SESSION_ROW[0], NULL) AS NEXT_SESSION_ID,
        IFF(PREV_SESSION_ID IS NULL, 1000000, DATEDIFF(SECOND, PREV_EVENT_DTM, e.EVENT_DTM)) AS PREV_SESSION_DTM_DIFF,
        IFF(NEXT_SESSION_ID IS NULL, 1000000, DATEDIFF(SECOND, e.EVENT_DTM, NEXT_EVENT_DTM)) AS NEXT_SESSION_DTM_DIFF,
        CASE LEAST(PREV_SESSION_DTM_DIFF, NEXT_SESSION_DTM_DIFF)
            WHEN NEXT_SESSION_DTM_DIFF
                THEN NEXT_SESSION_ID
            ELSE
                PREV_SESSION_ID
        END AS CORR_SESSION_ID
    FROM
        PROD_ODS_BATCH.YANDEXMETRIKA_159737.EVENT e
)
```

...

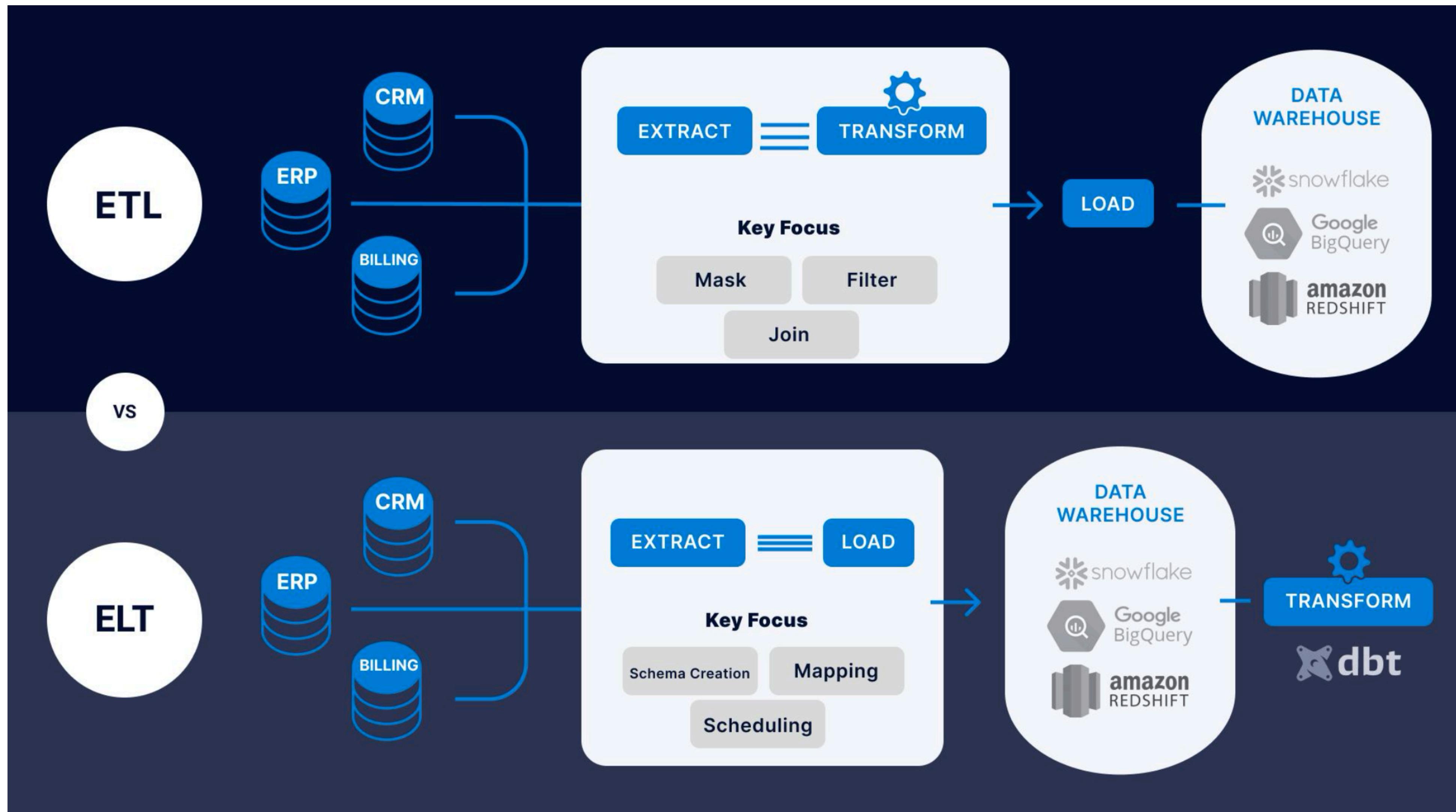
## OLTP

```
SELECT "id", "created_at", ...
FROM public.orders
WHERE "id" = 219781
```

# ETL vs ELT

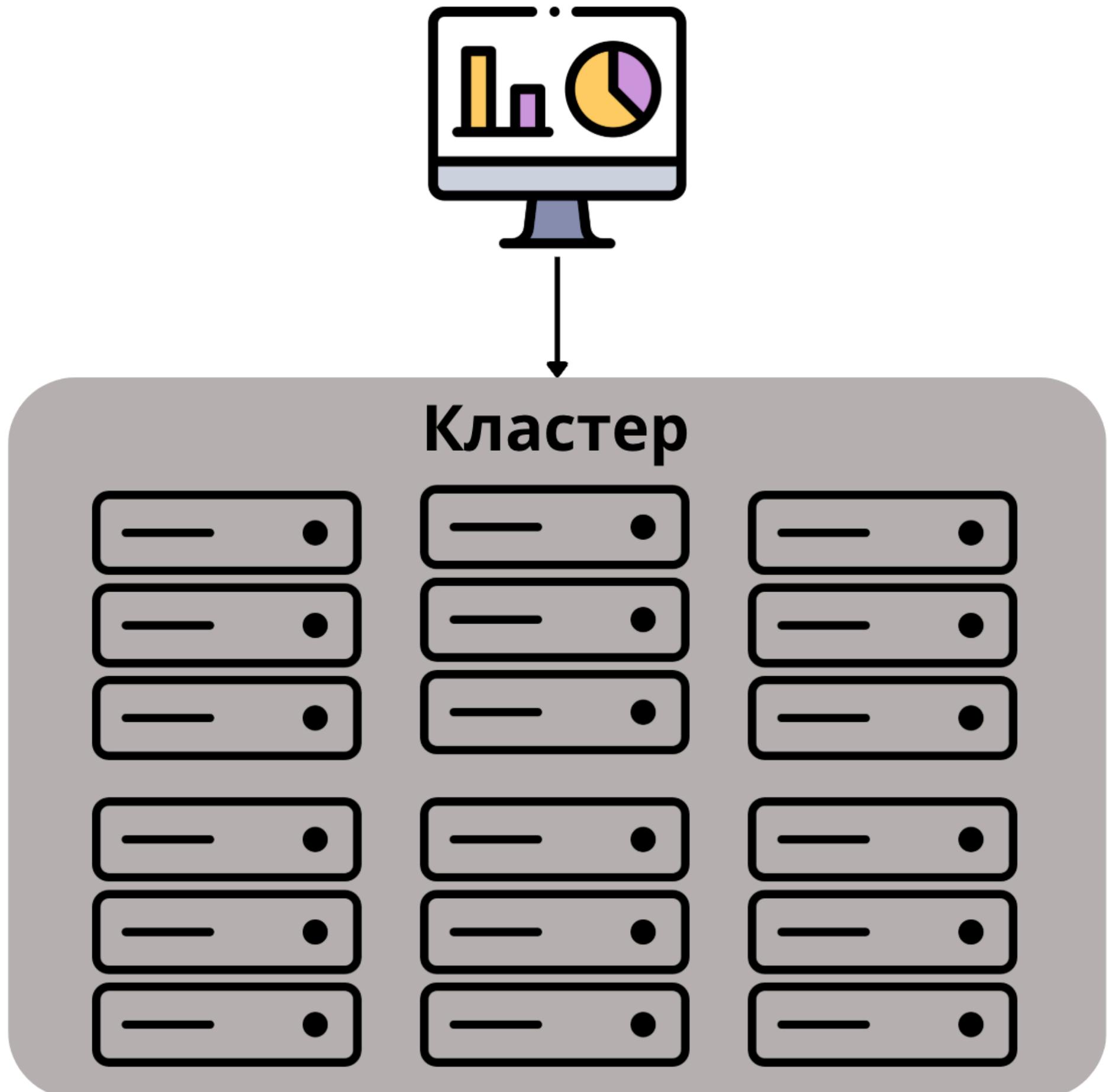


# ETL vs ELT



# Кластер

**Кластер** — это группа серверов, объединенных с помощью коммуникационных каналов для совместного решения информационно-вычислительных задач. Для пользователя кластер выступает единой системой, независимо от своей структуры и состава. Логически кластер существует как единый сервер, составленный из объединенных в группу серверных машин.



**Теперь начнем отвечать на наши вопросы**

# **2 ключевых вопроса в Big Data**

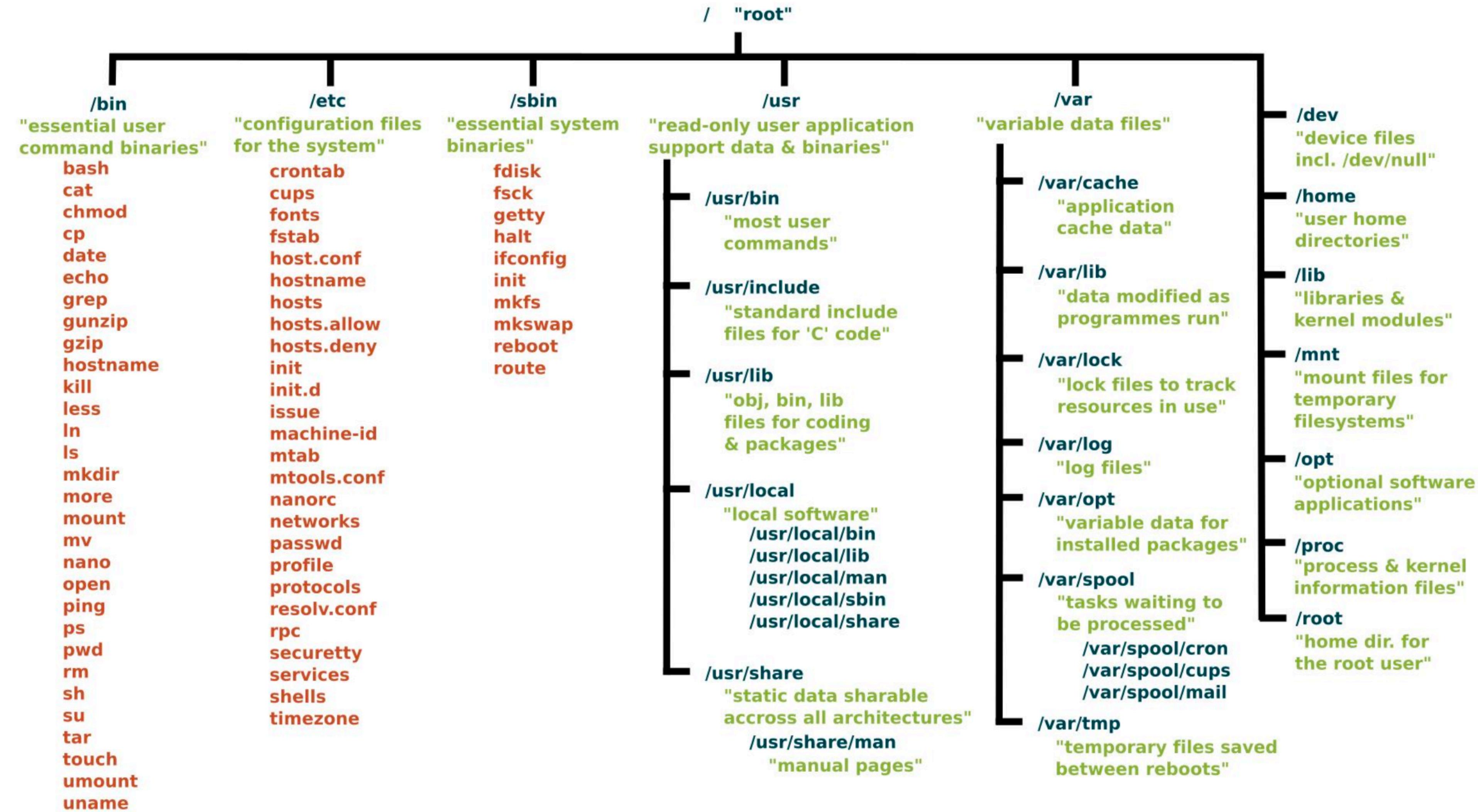
**1. Как хранить большие данные? (Storage Layer)**

# Файловая система

**Основными функциями файловой системы являются:**

- размещение и упорядочивание на носителе данных в виде файлов;
- определение максимально поддерживаемого объема данных на носителе информации;
- создание, чтение и удаление файлов;
- назначение и изменение атрибутов файлов (размер, время создания и изменения, владелец и создатель файла, доступен только для чтения, скрытый файл, временный файл, архивный, исполняемый, максимальная длина имени файла и т. п.);
- определение структуры файла;
- поиск файлов;
- организация каталогов для логической организации файлов;
- защита файлов при системном сбое;
- защита файлов от несанкционированного доступа и изменения их содержимого

# Linux File System

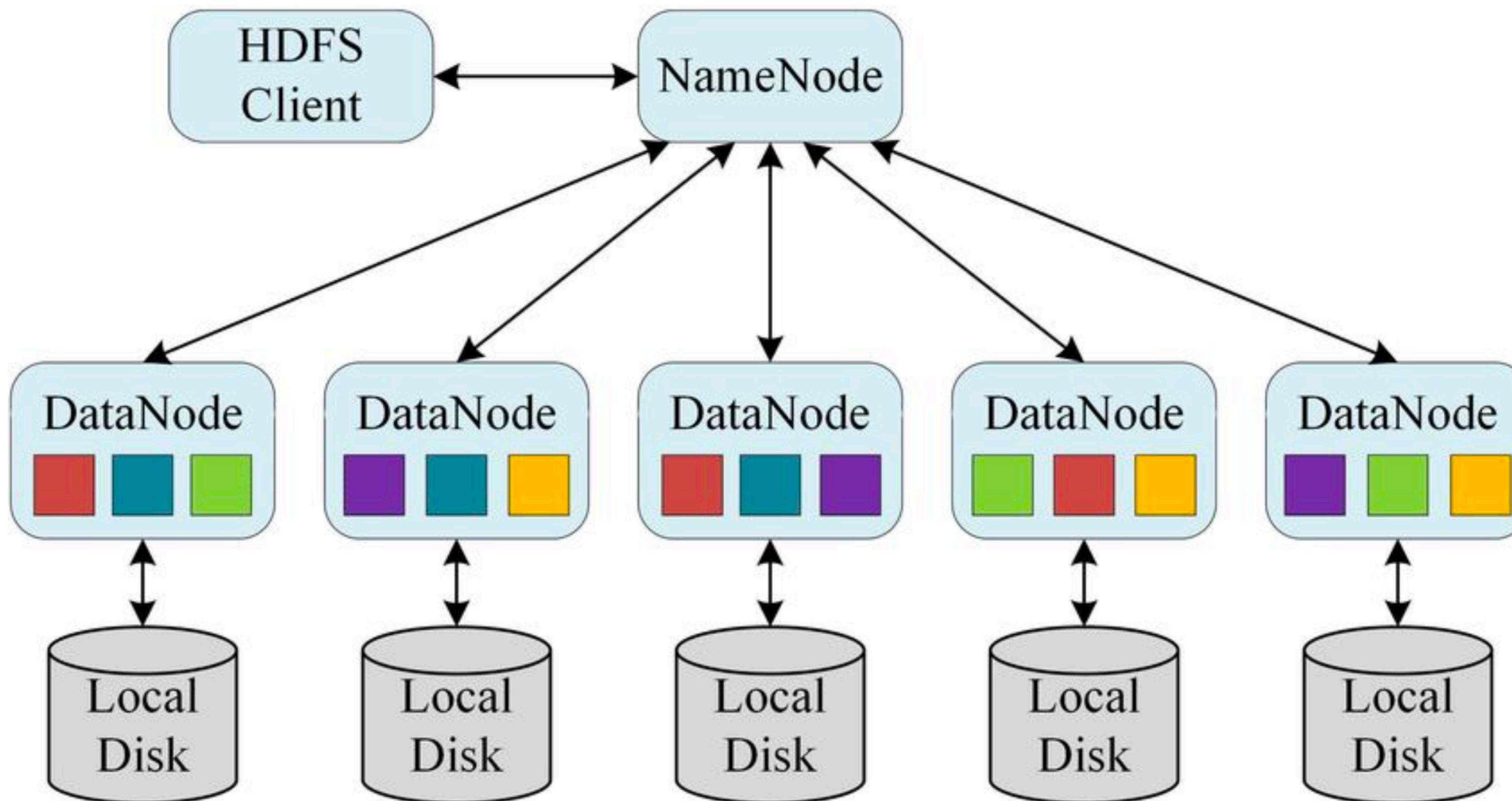


# HDFS - Hadoop Distributed File System

## Основные свойства:

- Доступность данных, за счёт механизмов отказоустойчивость;
- Приложения, работающие в HDFS, имеют большие наборы данных. Типичный файл в HDFS имеет размер от гигабайтов до терабайт;
- Эффективно работает в парадигме Write once Read many;
- Не требует дорогостоящего оборудования и может быть запущена на маломощных машинах;

# Как устроена HDFS

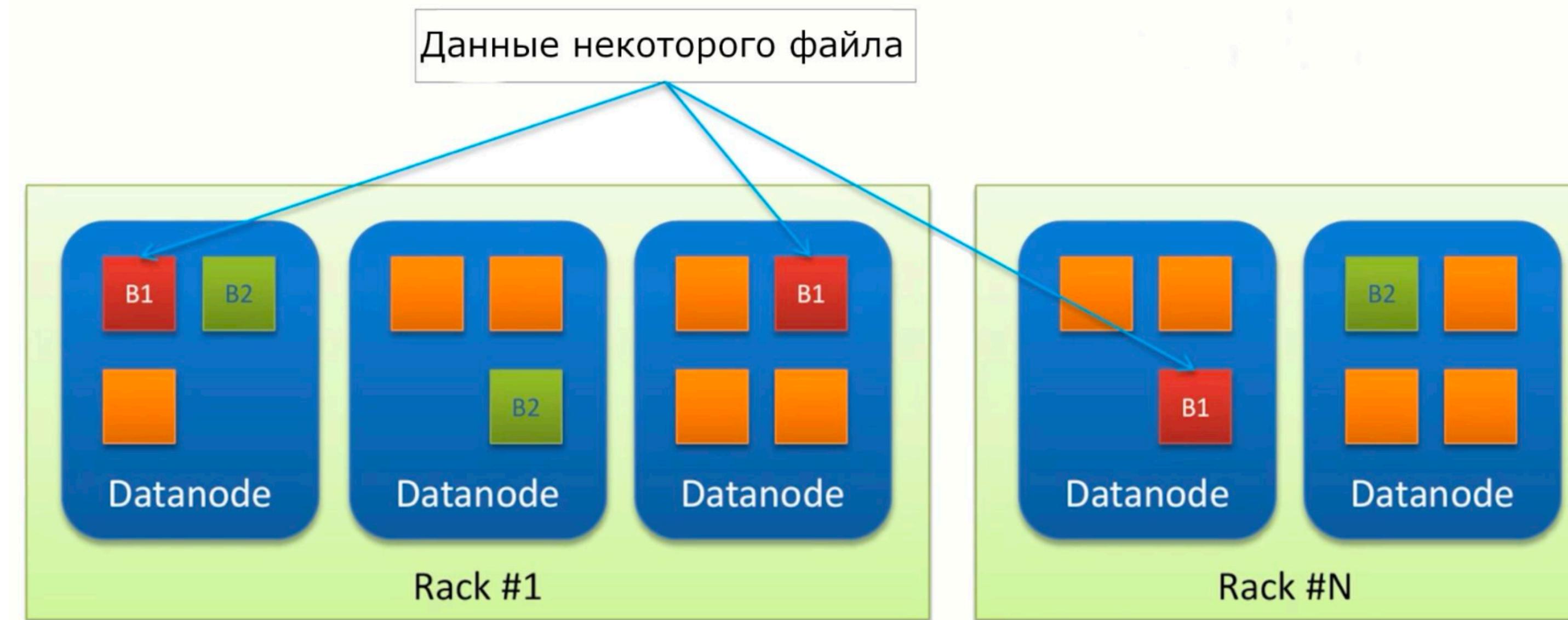


# Из чего состоит HDFS

- Кластер - набор вычислительных узлов (серверов), соединённых сетью;
- HDFS client - аппаратный или программный компонент системы, откуда поступают команды на произведение некоторых действий в распределённом хранилище данных;
- NameNode - специально выделенных узел, на котором содержится мета-информация о том, на каком узле хранятся данные для того или иного файла
- DataNode - один из множество серверов, на котором непосредственно располагаются данные

# Отказоустойчивость в HDFS

Основной принцип, благодаря которому достигается надёжное хранение данных в HDFS заключается в том, что копии одних и тех же данных располагаются на нескольких отдельных друг от друга серверах. (Фактор репликации)



# **2 ключевых вопроса в Big Data**

**2. Как обрабатывать большие данные? (Processing layer)**

# Модель вычисления Map-Reduce

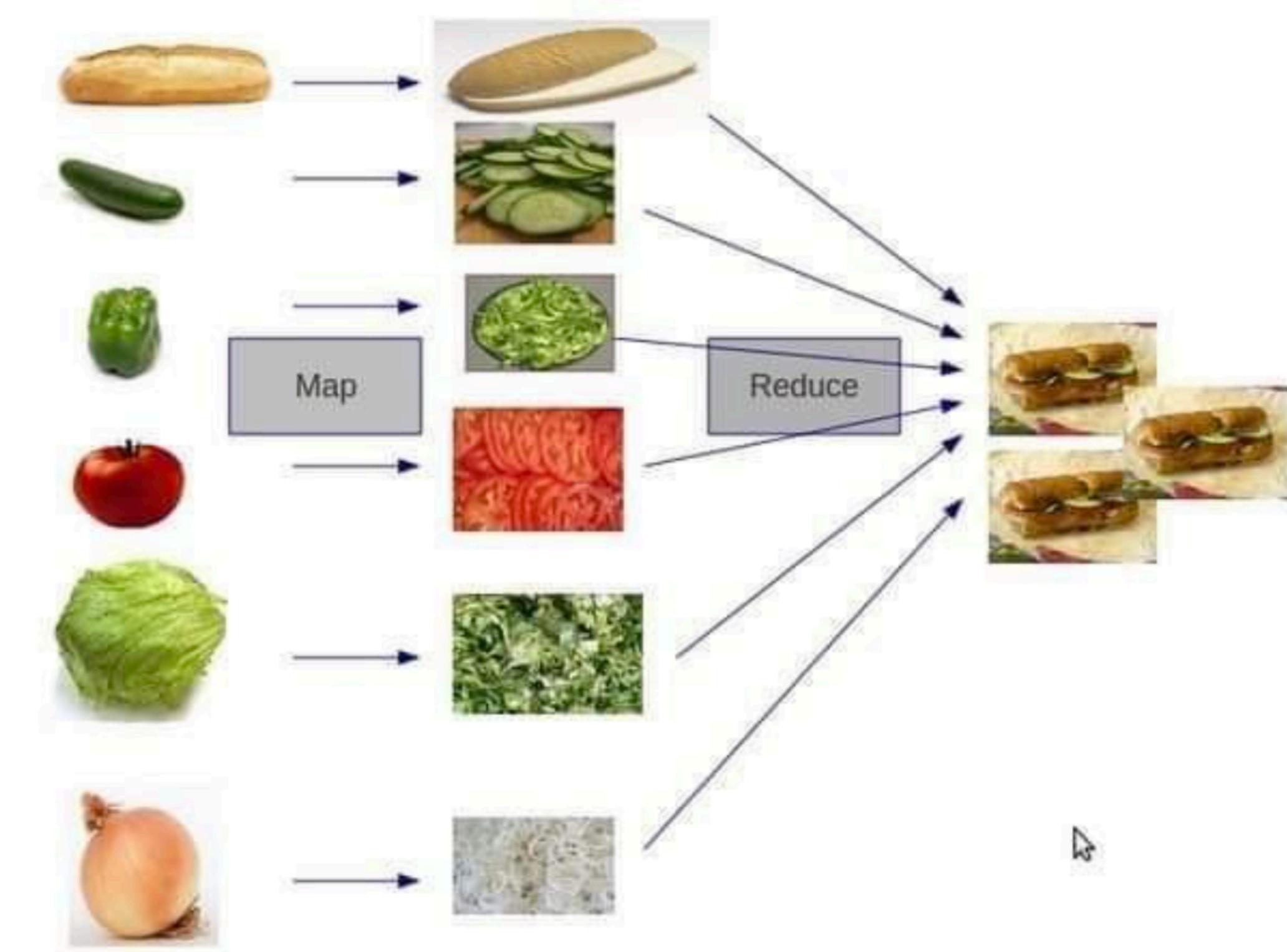
Для эффективной обработки больших объёмов данных применяется двухшаговая модель вычислений Map-Reduce.

- Шаг Map: на данном шаге к каждой порции данных применяется некоторая функция  $F$ .

$$map(F, [x_1, \dots, x_N]) \rightarrow [F(x_1), \dots, F(x_N)]$$

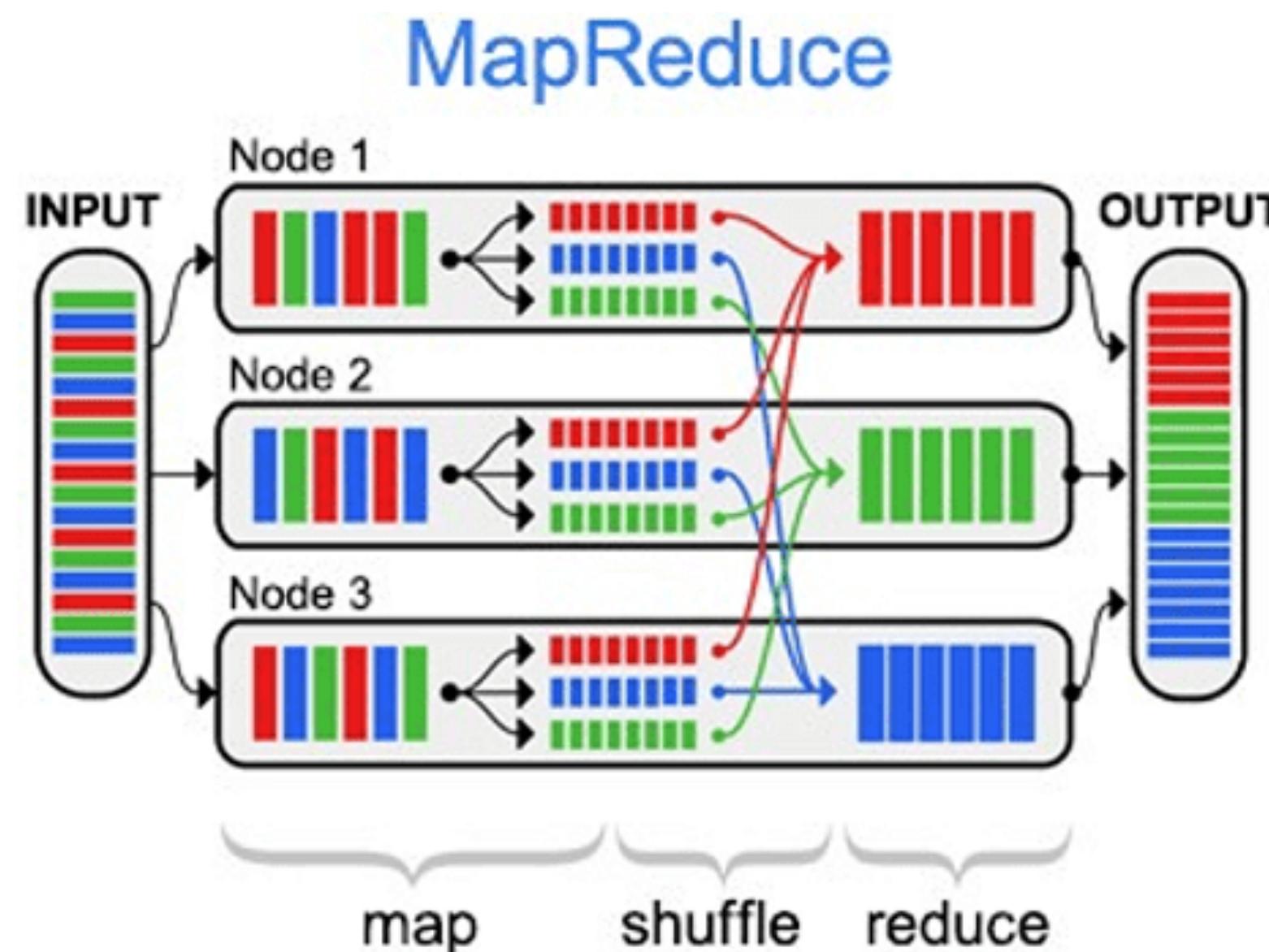
- Шаг Reduce: на данном шаге происходит объединение результатов из предыдущего шага Map и получение итогового результата

$$Reduce([F(x_1), \dots, F(x_N)]) \rightarrow Y$$



# MapReduce

**Задача: Посчитать количество слов в файле.**

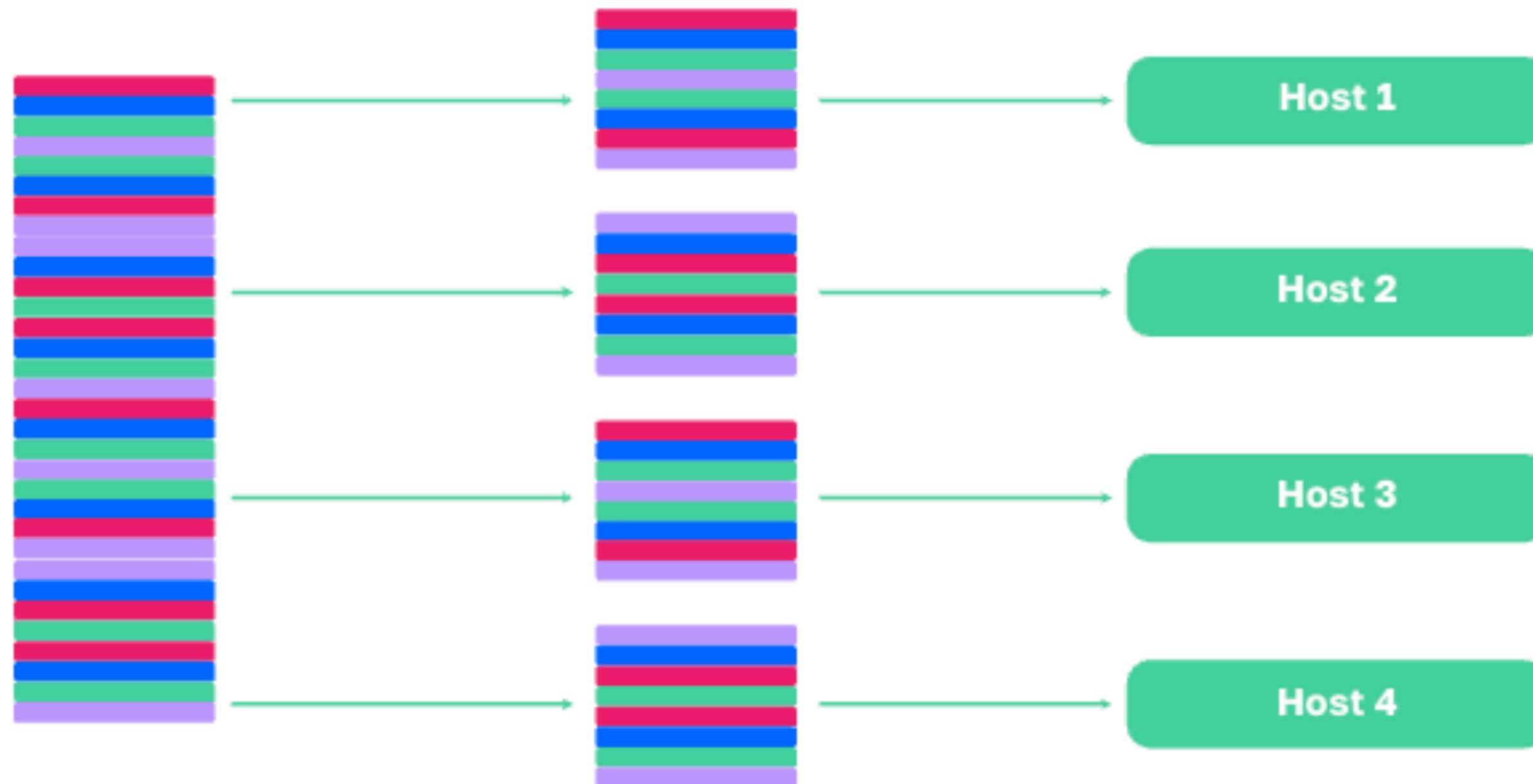


# MapReduce

На этапе Мар к каждому input применяется функция, которая преобразует входные данные (на каждой ноде/хосте)

## Шаг 1. Мар

На примере четырёх машин

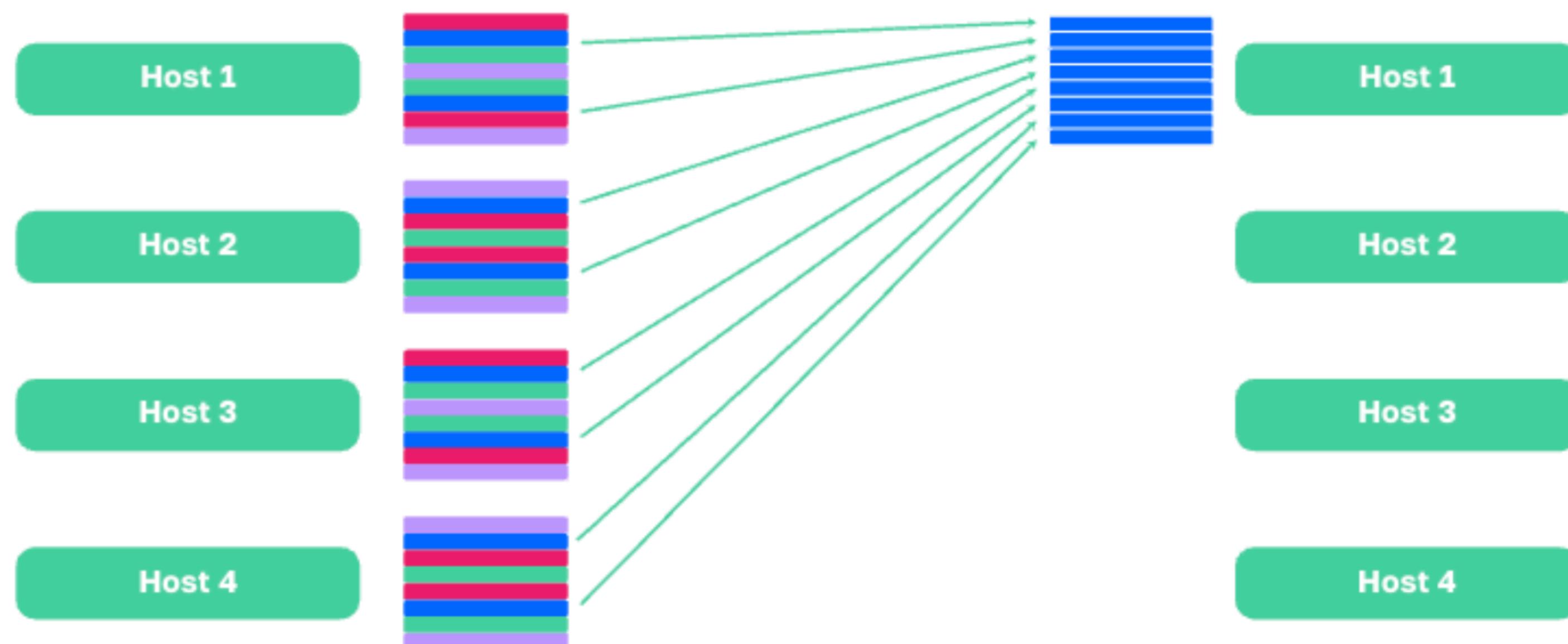


# MapReduce

Цель данного этапа  
направить правильным  
образом данные на  
Reducer

## Шаг 2. Shuffle

Все синие строчки окажутся на одной машине

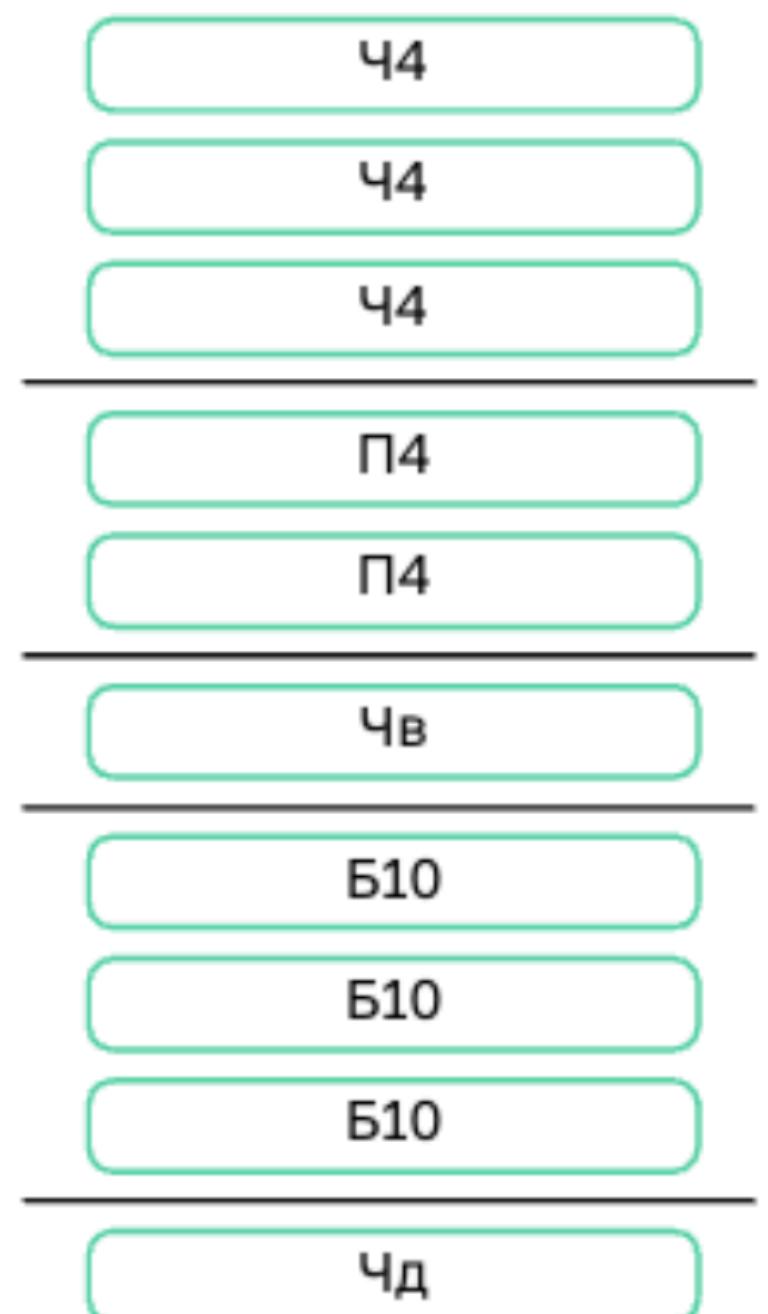


# MapReduce

На этапе Reduce происходит агрегация набора данных – в нашем случае суммирование количества одинаковых слов (на каждой ноде/хосте)

## Шаг 3. Reducer

- Читаем отсортированный файл построчно
- Сравниваем текущую строку с прошлой
- Если равны, то увеличиваем счетчик на 1
- Если нет, то пишем результат и обнуляем счетчик
- Расход оперативной памяти минимальный



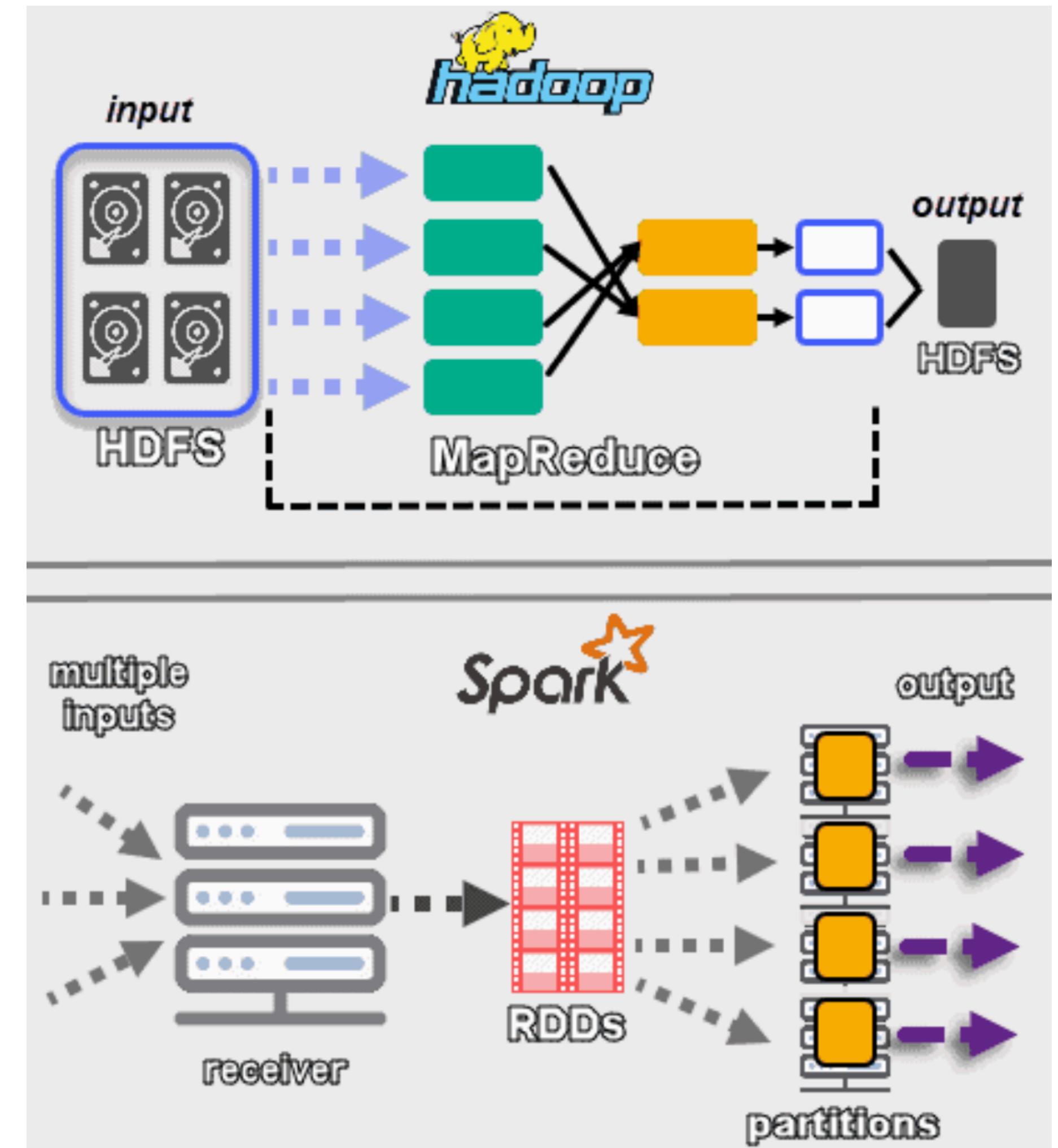
# Как с этим работают на практике

Запускать вычисления на распределённом вычислительном кластере крайне непростая задача.

Необходимо решить множество аспектов, связанных с как с настройкой самого кластера так и с написанием корректных и эффективных программ на нём.

Для упрощения работы с большими данными были разработаны технологии, позволяющие разработчикам:

1. Абстрагироваться от технических деталей
2. Писать более эффективные программы за счёт оптимизации вычислений



**Спасибо за внимание!**