



Отметьте посещение

НИУ ВШЭ

Современные архитектуры БД (BigData)

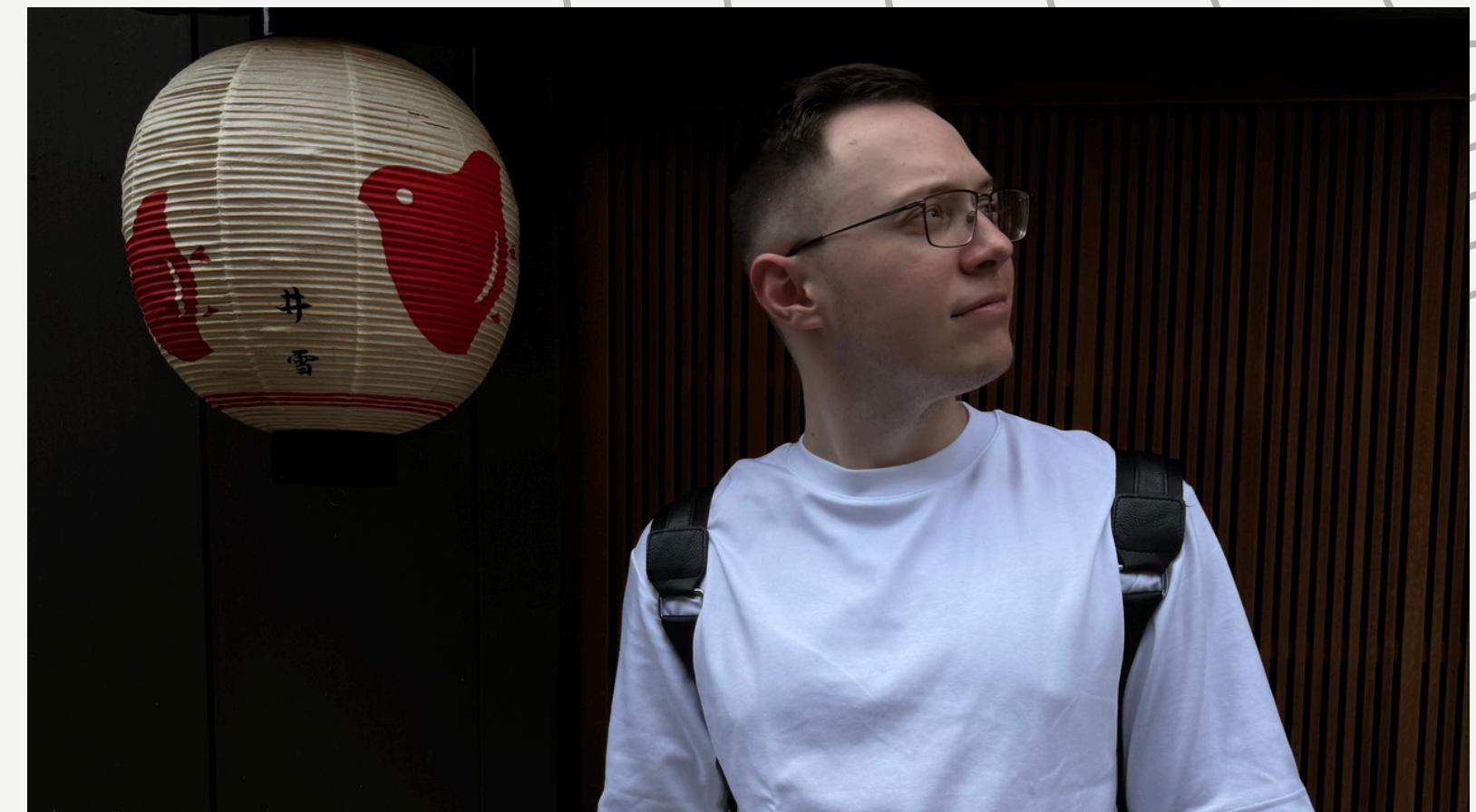
Сысоев Кирилл Романович

Обо мне

4.5+ лет в BigData
HSE University

Senior Data Engineer
1) GlowByte Consulting
2) PochtaTech
3) OneFactor

Hadoop, Spark, Kafka, k8s, YandexCloud
Python/Scala, SQL



t.me/KRSysoev
ksysoev@hse.ru

О ЧЕМ ПОГОВОРИМ

- Предпосылки создания Хранилища Данных
 - Виды Хранилища Данных
 - BigData
 - Распределенные вычисления

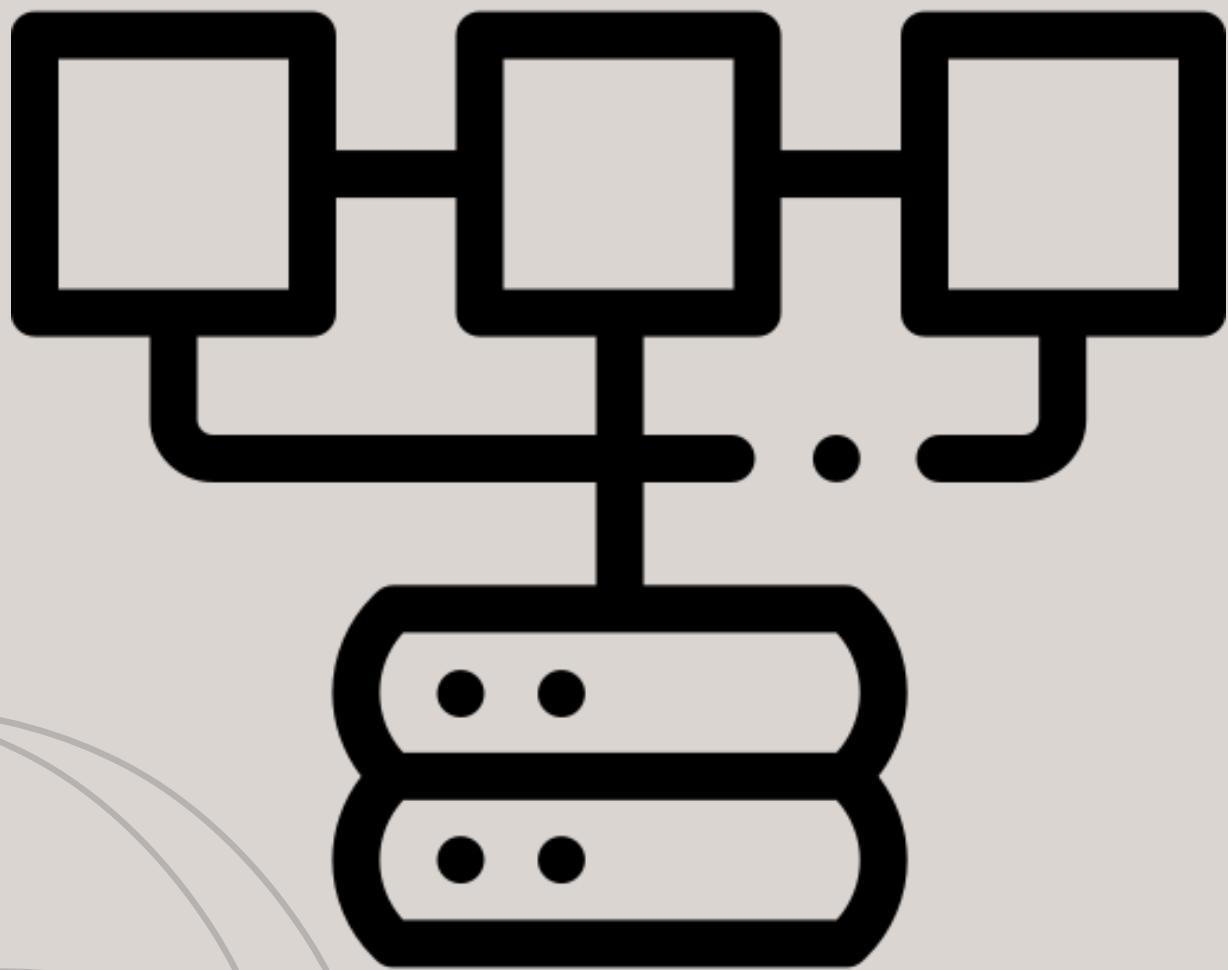
Эволюция архитектур



https://www.youtube.com/watch?v=EvefrwYmOn0&ab_channel=TechTrain



БД vs СУБД



База Данных

База данных — это набор упорядоченных и структурированных данных, которые обычно хранятся в электронном виде в компьютерной системе.

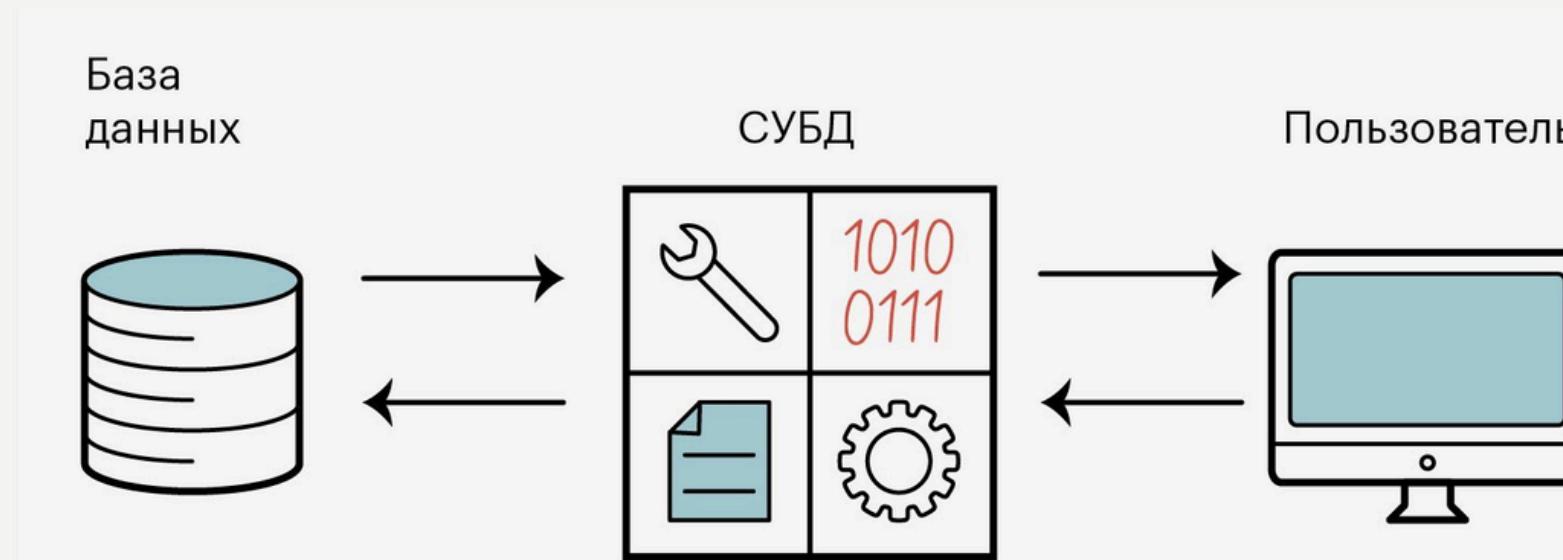
База данных обычно управляетя системой управления базами данных (СУБД).

Примеры

- 1) Таблицы с клиентами/заказами/товарами/...
- 2) Документы в спец форматах (csv, json, xml, ...)
- 3) Документ, содержащий графовые связи между пользователями
- 4) Файл ключ-значение: логи сессий пользователей

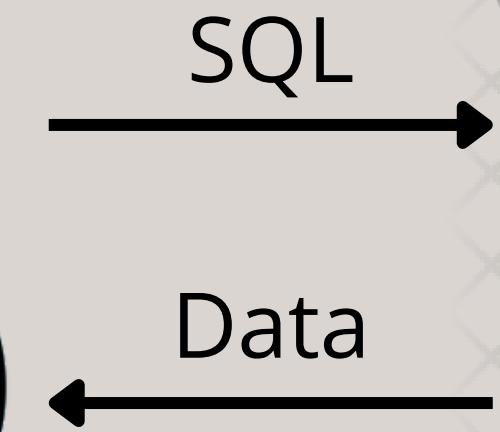
СУБД

Система управления базами данных (СУБД)
— это набор инструментов, которые
позволяют удобно управлять базами данных:
удалять, добавлять, фильтровать и находить
элементы, менять их структуру и создавать
резервные копии.



Типы СУБД

1. **Реляционные** (MySQL, PostgreSQL, Oracle Database, Microsoft SQL Server)
2. **Ключ-значение** (Redis, Amazon DynamoDB, Memcached)
3. **Документо-ориентированные** (MongoDB, CouchDB, RavenDB)
4. **Колонко-ориентированные СУБД (Column-oriented DBMS)** (Apache Cassandra, HBase, Amazon Redshift)
5. **Графовые** (Neo4j, ArangoDB, Amazon Neptune)
6. **Поисковые базы данных (Search Engines)**
7. **Базы данных временных рядов**
8. **Объектно-ориентированные базы данных** (db4o, ObjectDB)
9. **Мультимодальные СУБД**
10. **Native XML СУБД**
11. **GEO/GIS (пространственные) и специализированные СУБД**
12. **Event СУБД (баз данных переходов состояний)**
13. **Контентные СУБД**
14. **Векторные базы данных**
- 15....



Свойства системы

Какой тип нагрузки?

Какой вид данных приходит?

Свойства системы

Какой тип нагрузки?

ОЛТР

Какой вид.данных приходит?

Структурированные
данные

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.

Вид данных

**Структурированные
данные**

Данные, организованные по **заранее определённой модели или схеме**, обычно представленные в виде таблиц с фиксированными полями и типами данных.

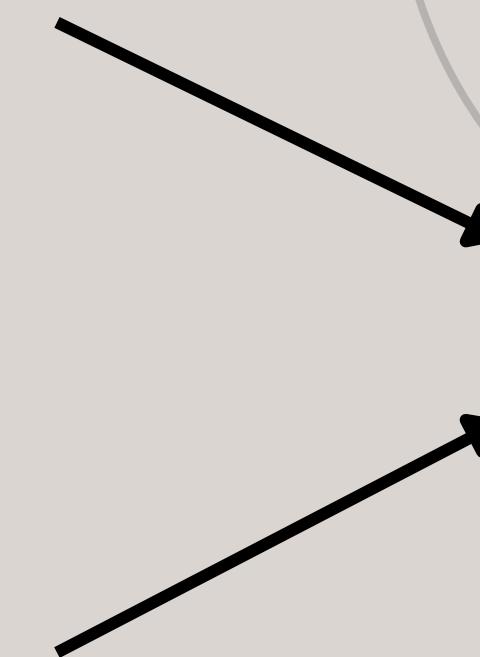
Такие данные легко вводить, хранить, искать и анализировать с помощью реляционных баз данных и SQL-запросов

**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.

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.

**Неструктурированные
данные**



СУБД





~~ETL~~



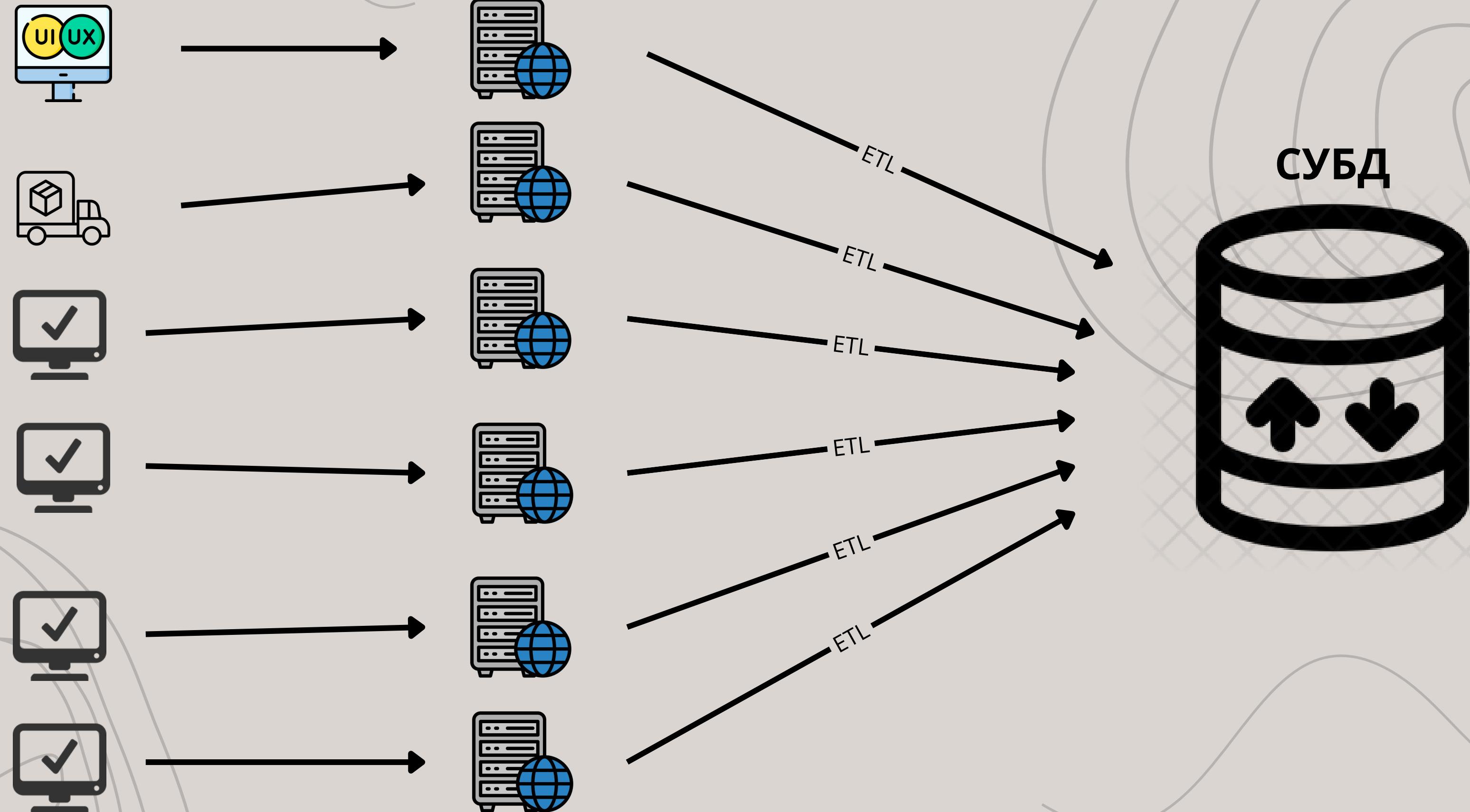


ETL

ETL

СУБД

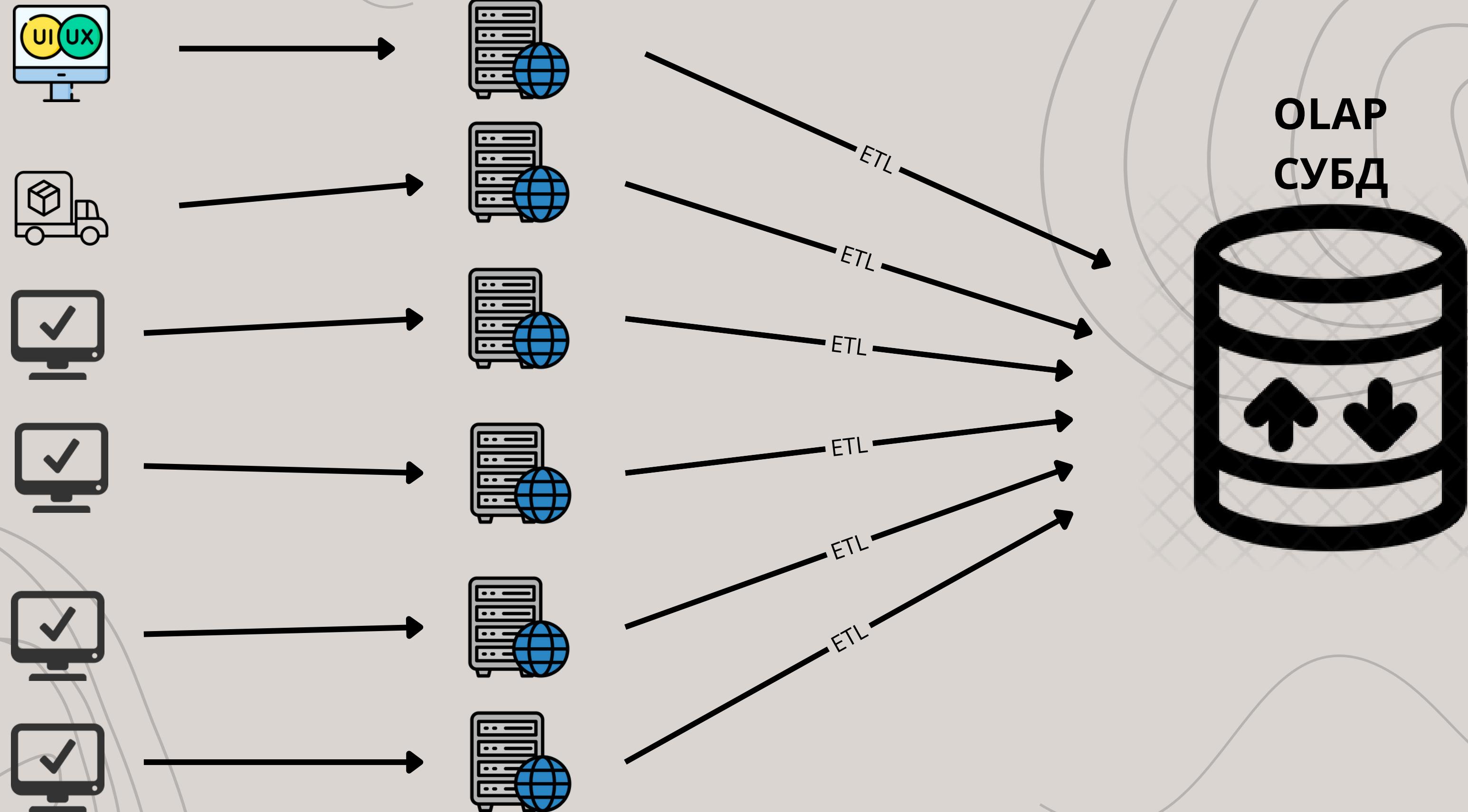




Задачи бизнеса

- 1) Анализ текучести кадров по параметрам**
- 2) Анализ затрат на обучение и развитие персонала**
- 3) Анализ производительности сотрудников**
- 4) Оценка вовлеченности сотрудников**
- 5) Анализ эффективности рекрутинговых компаний**

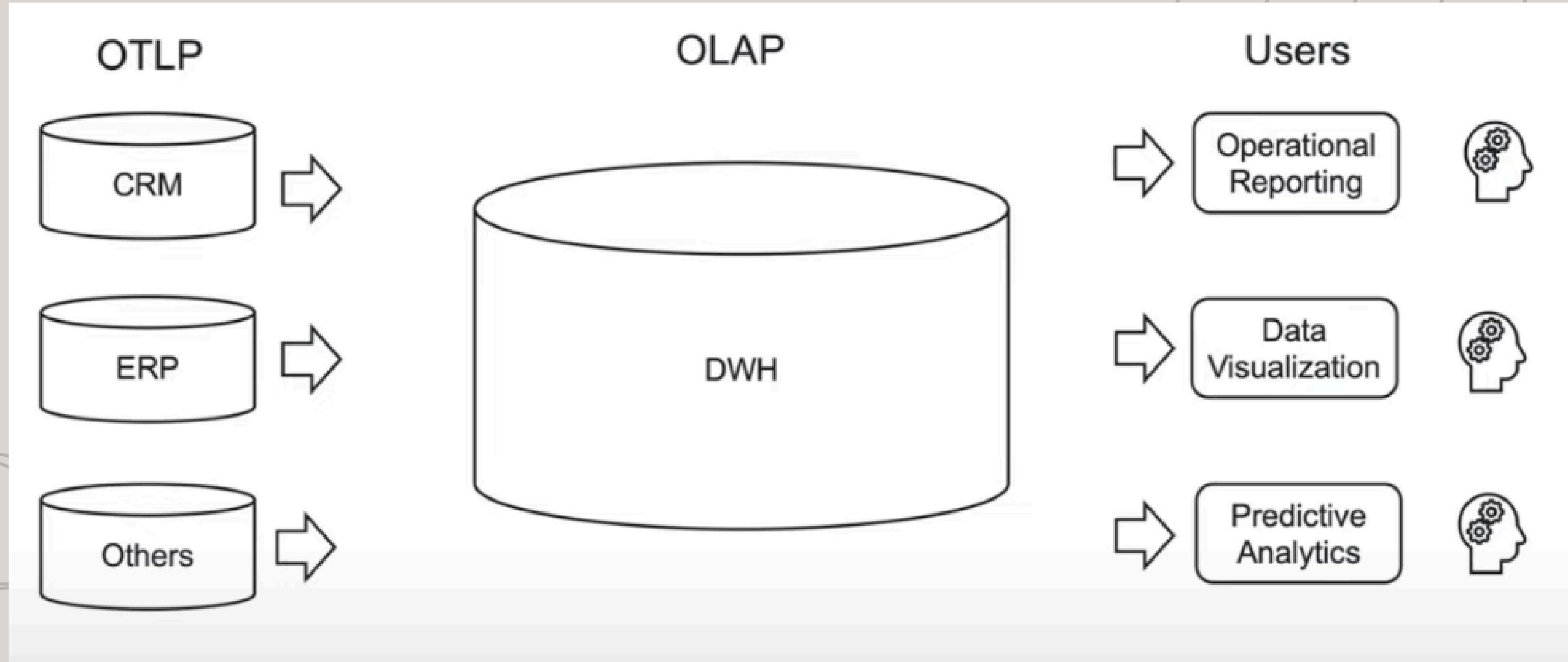
...



OLTP vs OLAP

Online Transactional Processing Online Analytical Processing

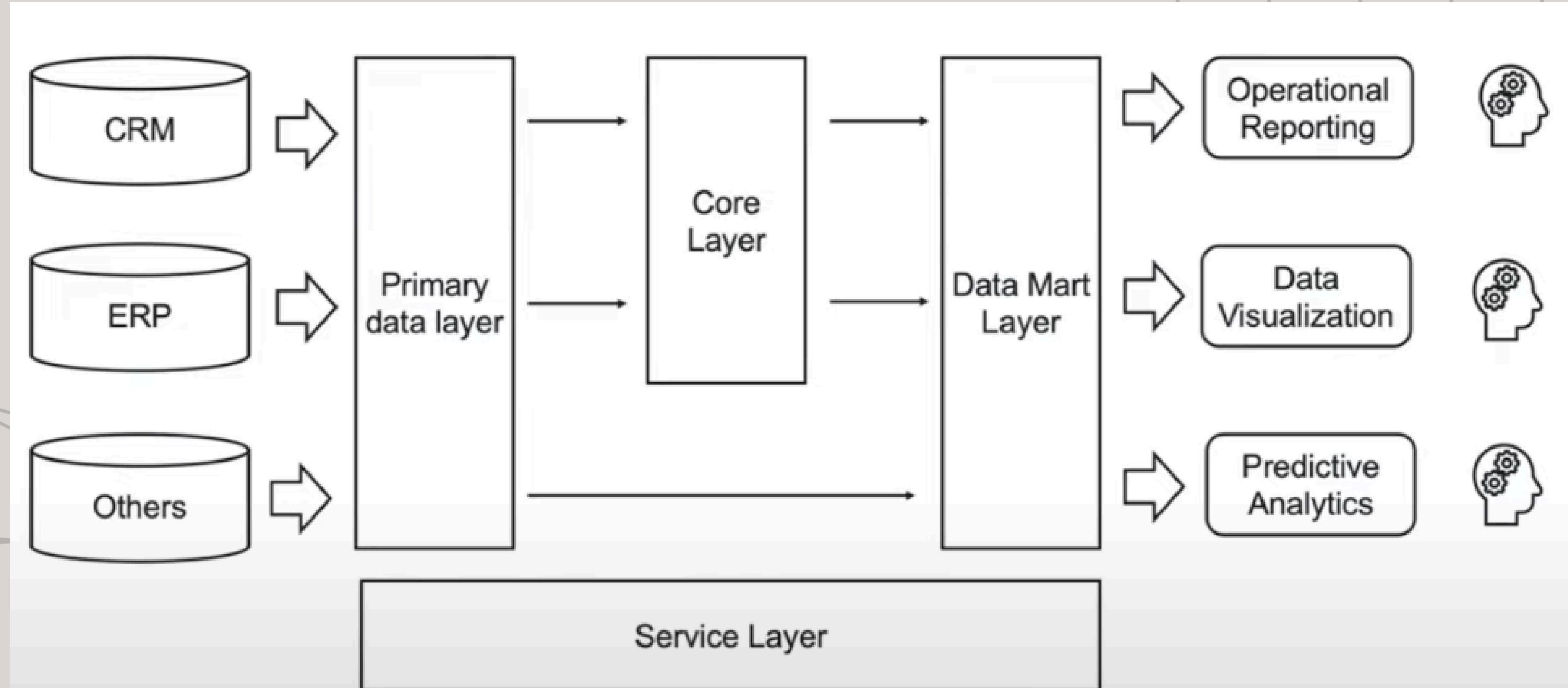
Критерии	OLAP	OLTP
Цель	OLAP помогает анализировать большие объемы данных для обеспечения поддержки принятия решений.	OLTP помогает управлять транзакциями в реальном времени и обрабатывать их.
Источник данных	OLAP использует исторические и объединенные данные из нескольких источников.	OLTP использует 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.



Что такое DWH

DWH (Data Warehouse) – это хранилище данных, которое используется для хранения большого объема структурированной и иногда полуструктурной информации, предназначенный для анализа и принятия решений на основе данных. Основная цель DWH – интеграция данных из различных источников для проведения глубокого анализа и отчетности.

Data Warehouse



Подходы проектирования DWH

Подход Иммана

Описание: В этом подходе хранилище данных проектируется как централизованное корпоративное хранилище, которое содержит нормализованные данные в 3-й нормальной форме (3NF). В рамках EDW все данные из разных систем-источников приводятся к единой корпоративной модели данных.

Основная идея: Создание хранилища данных как единого источника правды (single source of truth), где все данные детализированы, структурированы и нормализованы. Отчеты и аналитические витрины данных (Data Marts) создаются на основе этого корпоративного хранилища.

Преимущества:

- Целостная модель данных на уровне всей организации.
- Упрощенная интеграция данных из различных источников.
- Более точная и комплексная аналитика на уровне компании.

Недостатки:

- Длительный процесс разработки и развертывания.
- Сложность управления большим количеством нормализованных таблиц.
- Высокие требования к техническим ресурсам и поддержке.

Когда использовать: Подходит для крупных организаций, которым необходимо централизованное хранилище для работы с большими объемами данных и сложными аналитическими запросами.

Подход Кимбалла

Описание: В этом подходе хранилище данных проектируется как набор витрин данных (Data Marts), каждая из которых оптимизирована для определенной области бизнеса (например, финансы, продажи, маркетинг). Данные в витринах денормализованы и организованы по звездообразной или снежинкообразной схеме.

Основная идея: Быстрое создание витрин данных для конкретных бизнес-процессов, с фокусом на конечных пользователях и их потребности в аналитике. DWH представляет собой набор денормализованных витрин, которые интегрируются в виде «общей звезды».

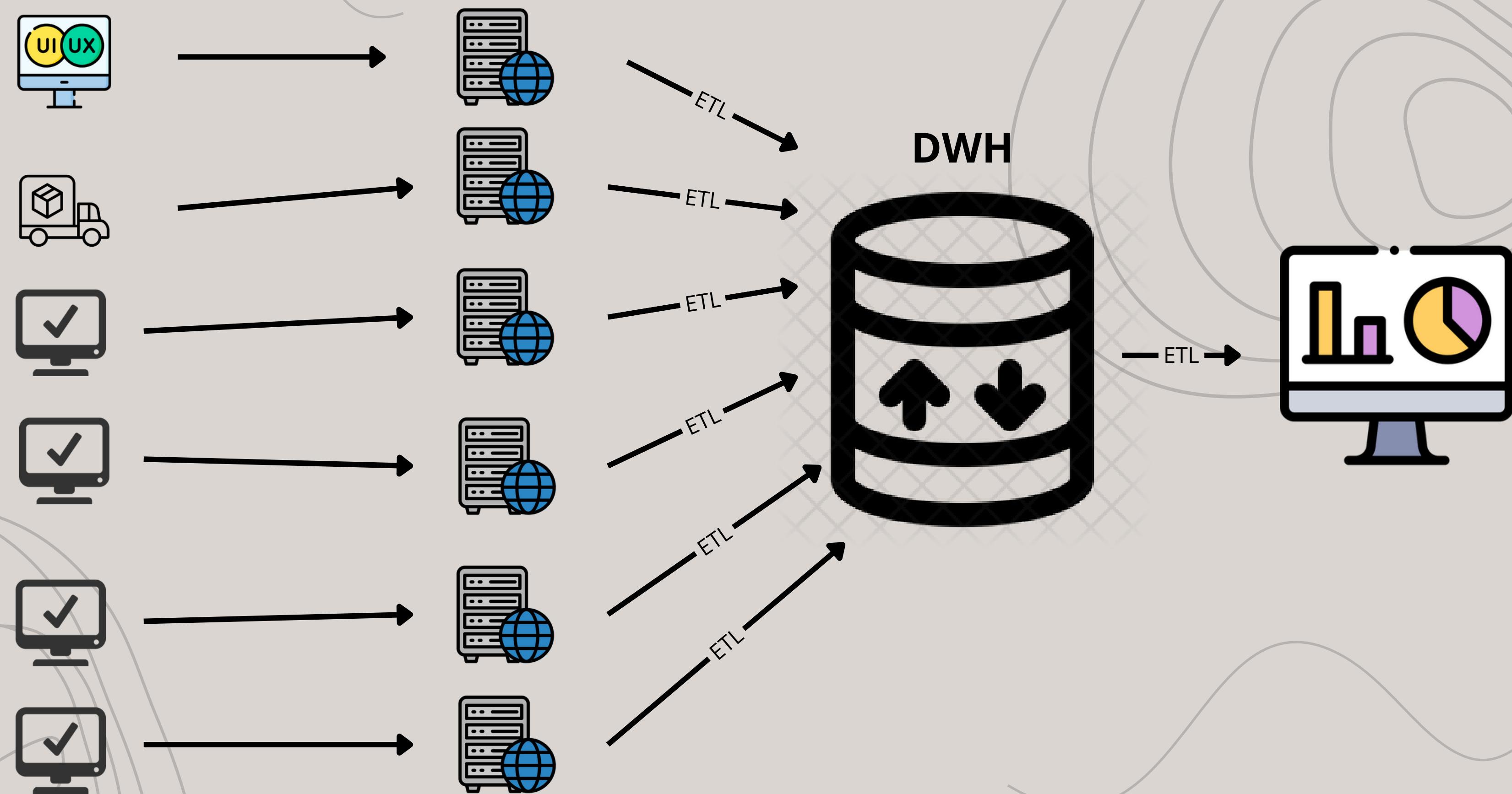
Преимущества:

- Быстрое развертывание аналитических решений для бизнес-пользователей.
- Прямая оптимизация под конкретные задачи и области бизнеса.
- Упрощенная структура данных для конечных пользователей.

Недостатки:

- Возможно дублирование данных между витринами.
- Трудности с поддержанием целостности данных на уровне всей организации.
- Ограничения при расширении масштаба или интеграции с новыми источниками данных.

Когда использовать: Подходит для небольших или средних компаний, которым требуется быстрое внедрение аналитики для различных подразделений. Также удобен, если есть необходимость в гибкости и быстром запуске решений.



Что такое BigData

Что такое BigData

01

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

02

Характеристики (5V):

- Volume (Объем)
- Velocity (Скорость)
- Variety (Разнообразие)
- Veracity (Достоверность)
- Value (Ценность)

03

Основные источники данных в Big Data:

1. **Социальные сети:** Посты, лайки, комментарии и активности пользователей на платформах, таких как Facebook, Twitter, Instagram.
2. **Интернет вещей (IoT):** Датчики, устройства и машины, генерирующие данные в режиме реального времени.
3. **Транзакционные системы:** Данные покупок, банковские транзакции, операции на биржах.
4. **Логи веб-сайтов и приложений:** Информация о действиях пользователей на сайтах, в приложениях и серверных логах.
5. **Мобильные устройства:** Геолокационные данные, данные приложений, активности пользователей.
6. **Медицинские системы:** Данные о пациентах, записи о состоянии здоровья, изображения, геномные данные.

Что такое Big Data

04

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 неструктурированными 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.

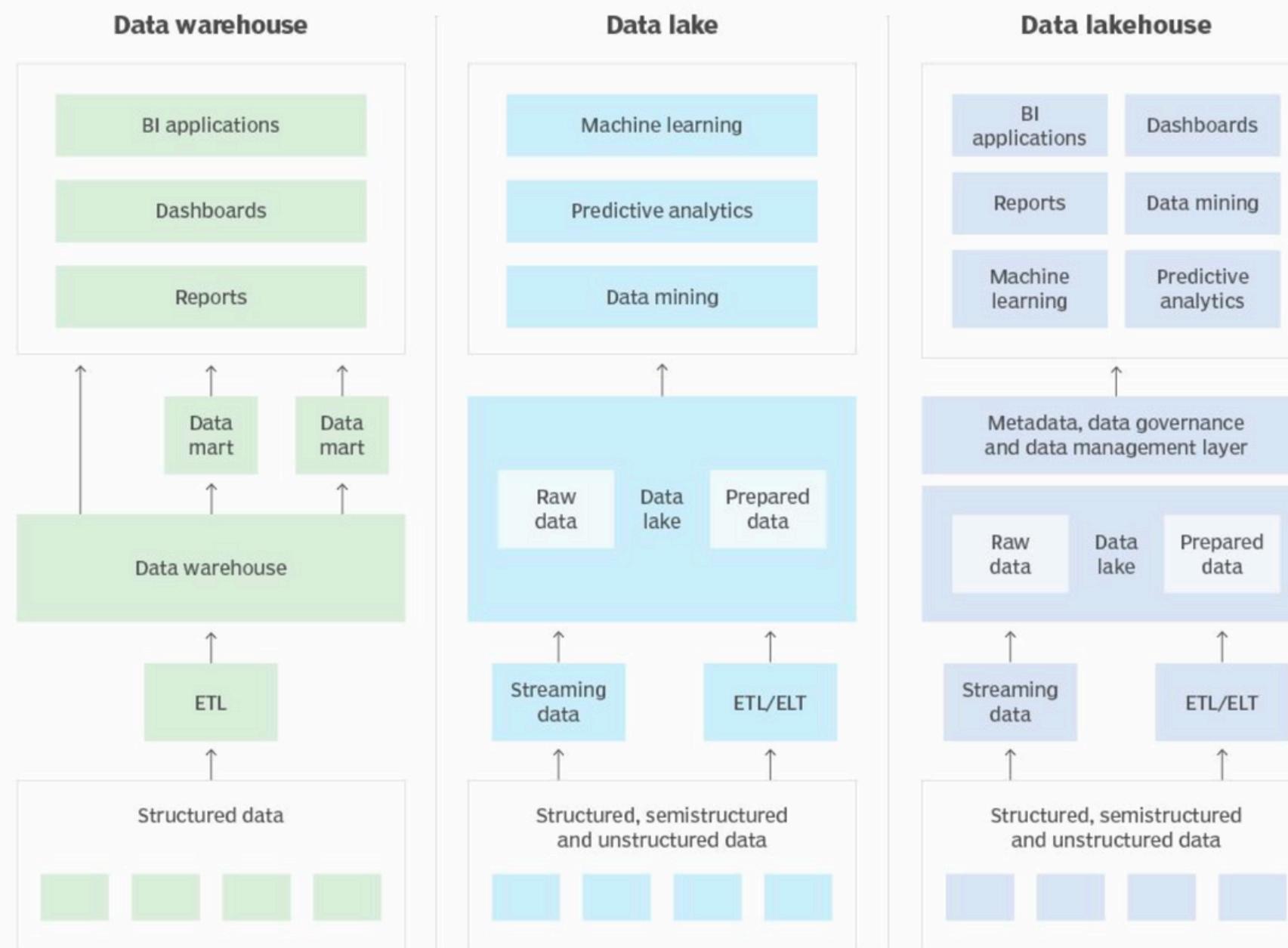
05

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.

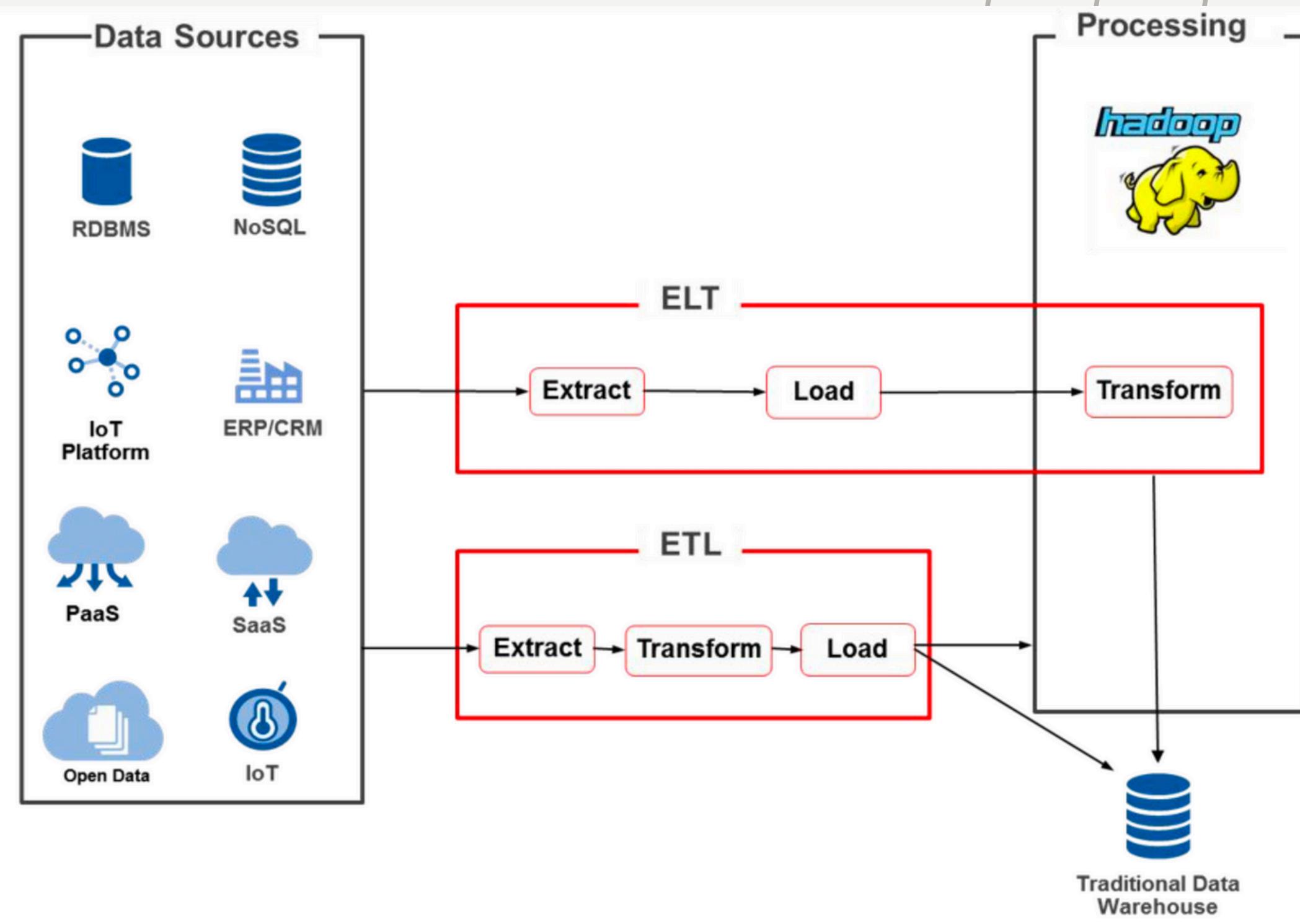
Data Lake

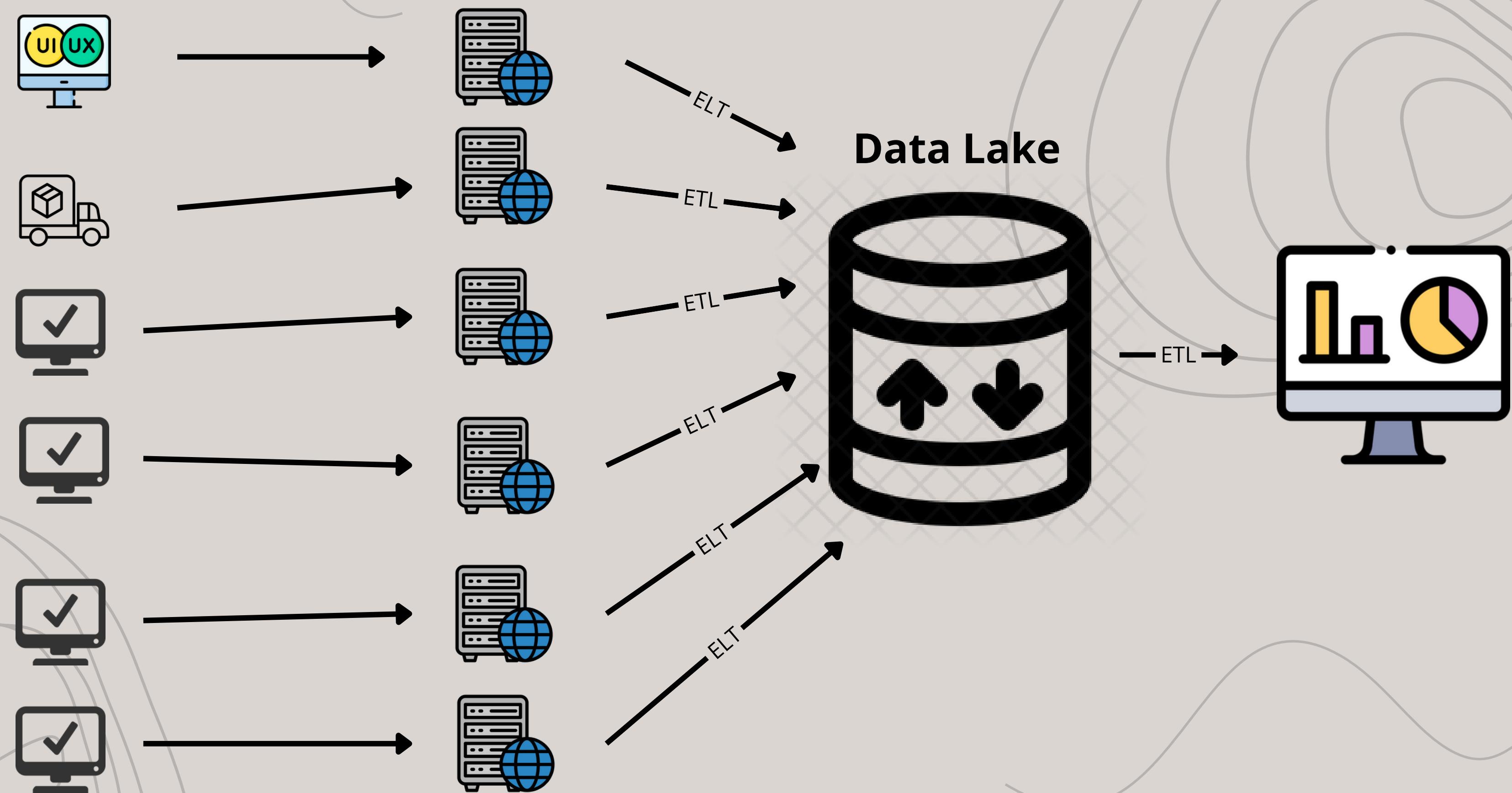
Data warehouse vs. data lake vs. data lakehouse



ETL vs ELT

ETL
VS.
ELT





Кластер

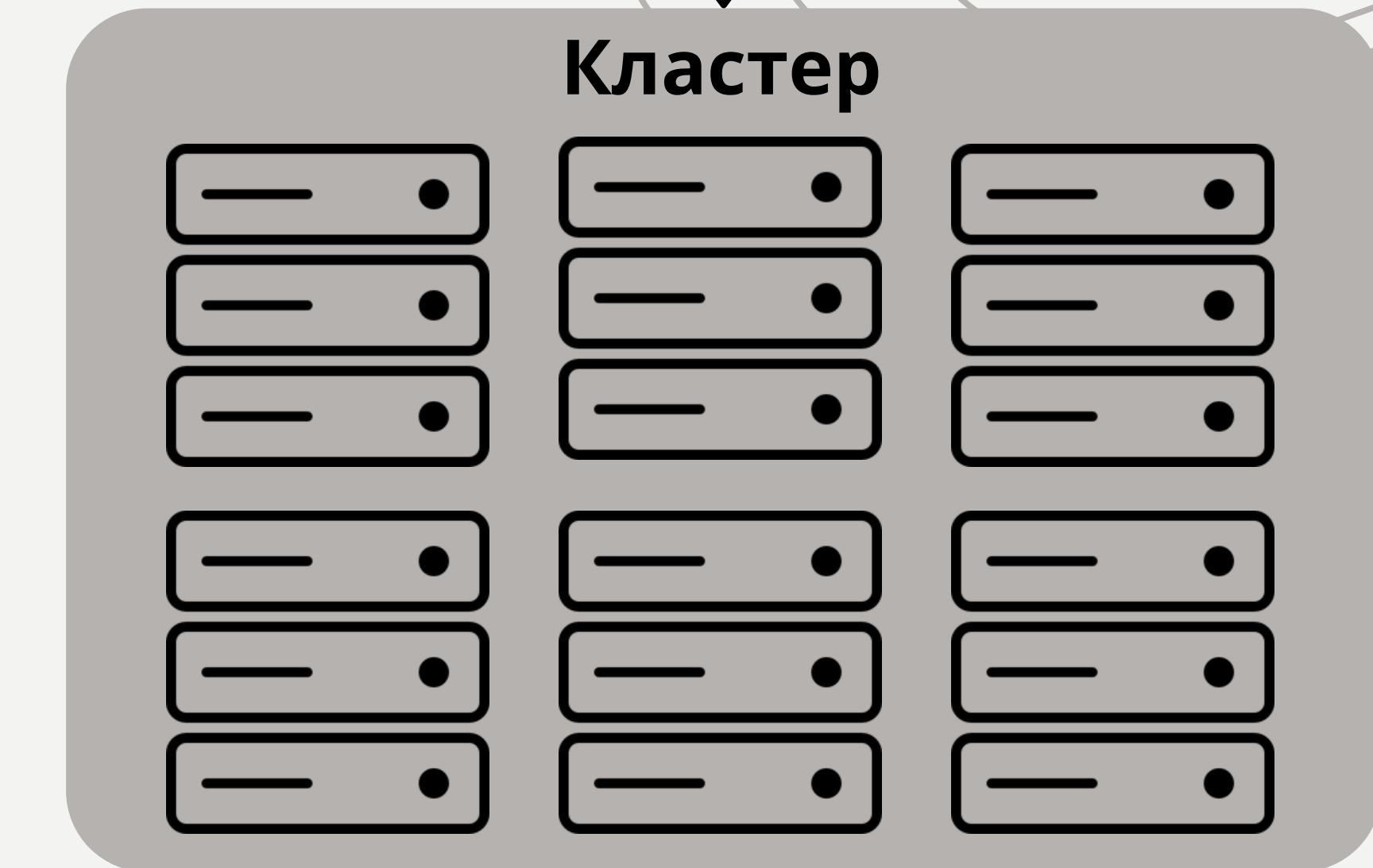
Кластер — это группа серверов, объединенных с помощью коммуникационных каналов для совместного решения информационно-вычислительных задач.

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.



Кластер



Задача HR-аналитики

Анализ текучести кадров с учетом различных факторов



Таблица сотрудников
(employee)



Таблица увольнений
(dismissals)

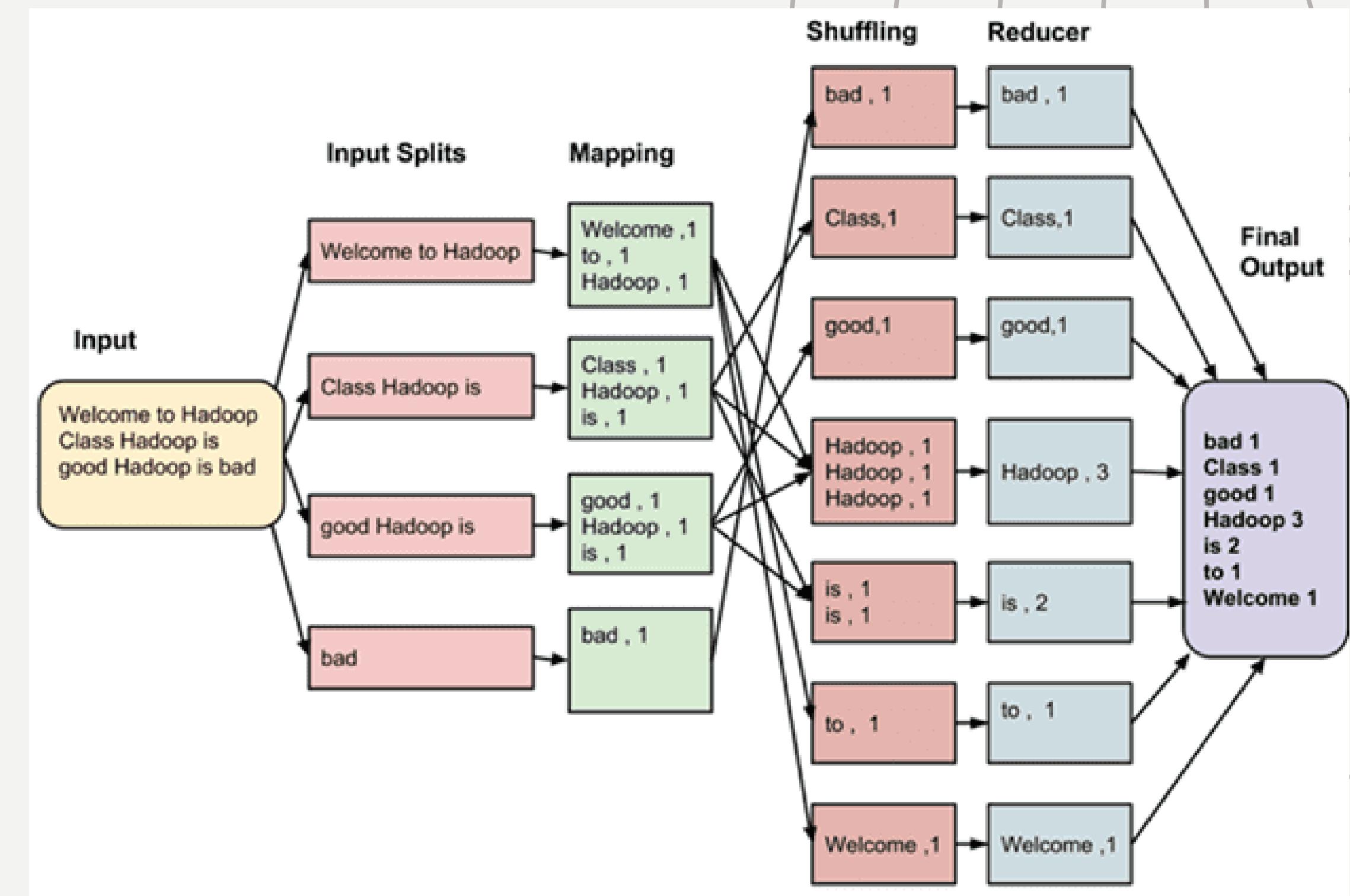
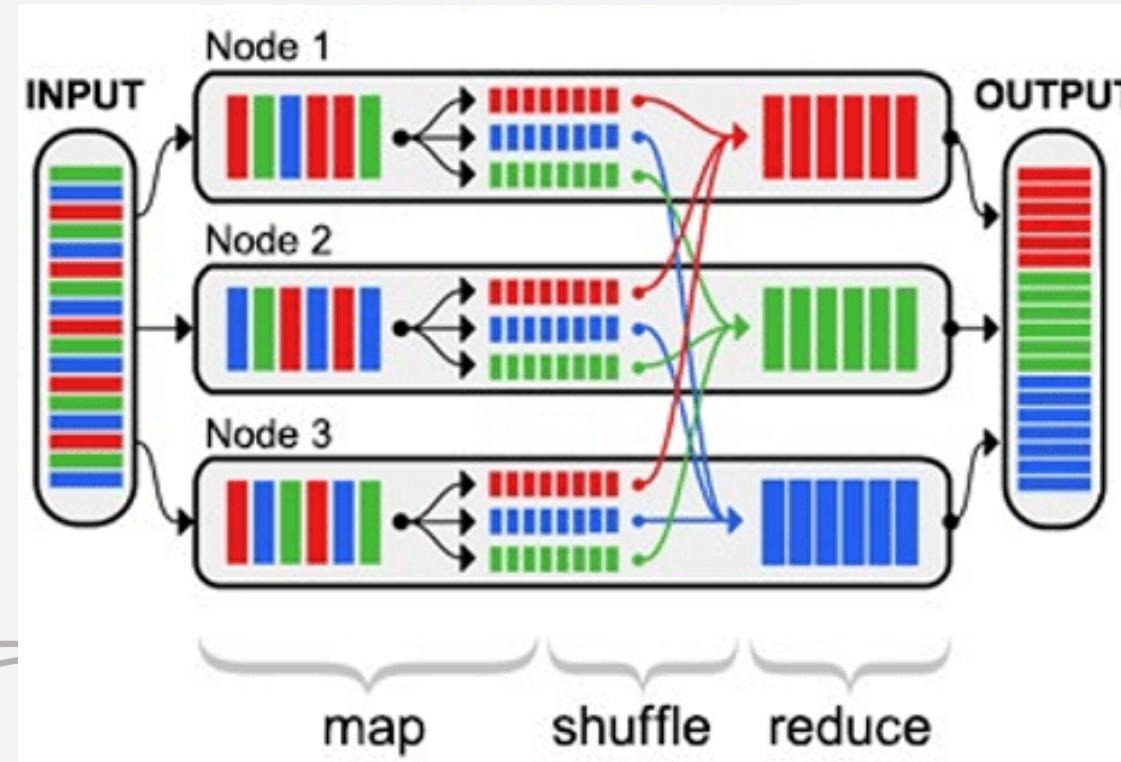
Основные параметры:

- 1) Возраст
- 2) Должность
- 3) Стаж работы

SQL:

```
SELECT
    e.age,
    e.job,
    e.exp,
    count(e.id) as cnt
FROM employee as e
INNER JOIN dismissals as d
    ON e.id = d.emp_id
    AND d.dis_flg = 1
GROUP BY e.age, e.job, e.exp;
```

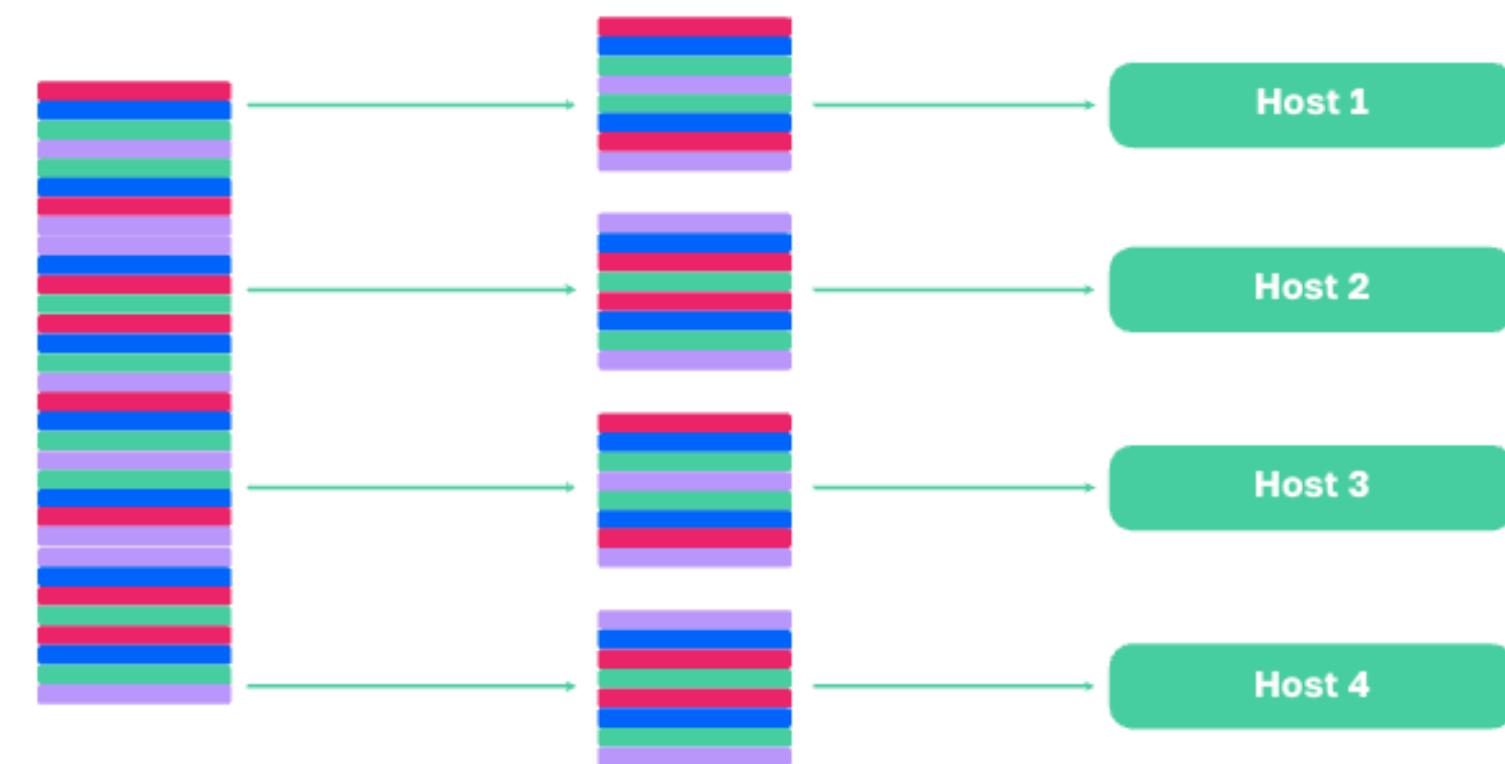
MapReduce



MapReduce

Шаг 1. Мар

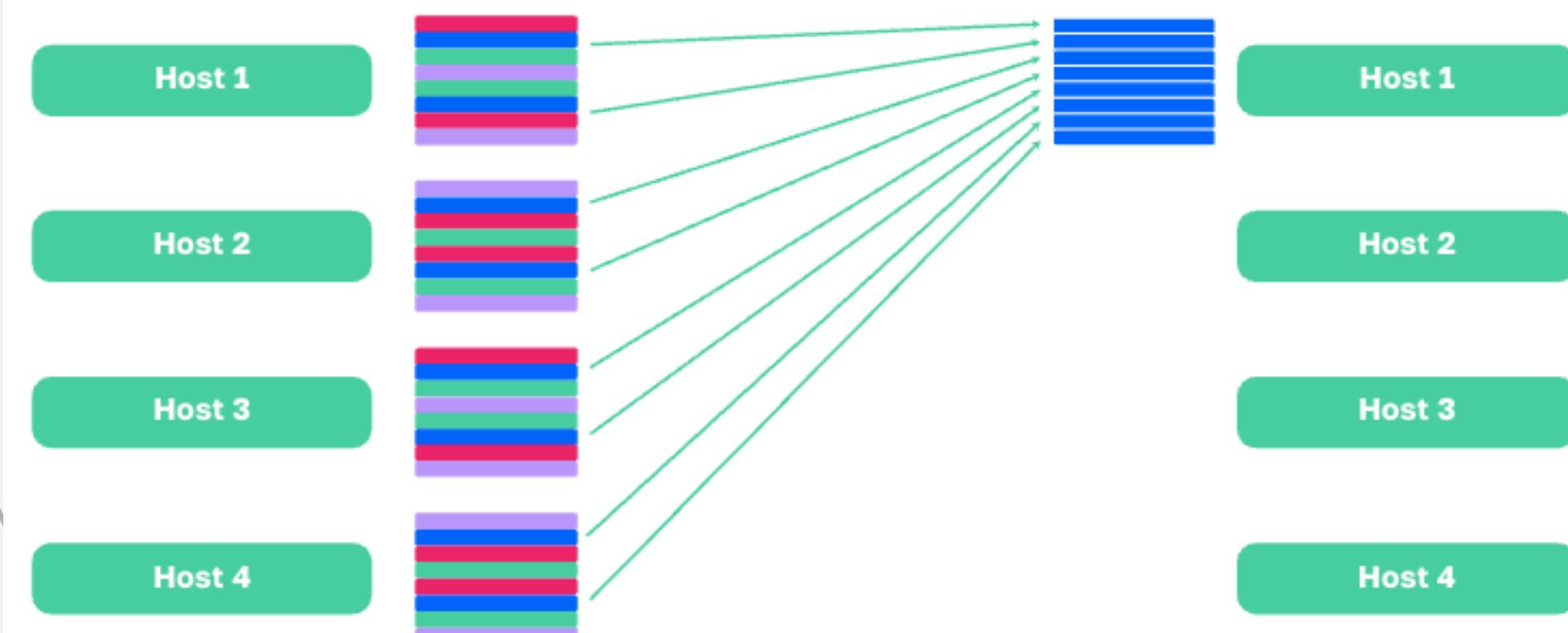
На примере четырёх машин



MapReduce

Шаг 2. Shuffle

Все синие строчки окажутся на одной машине



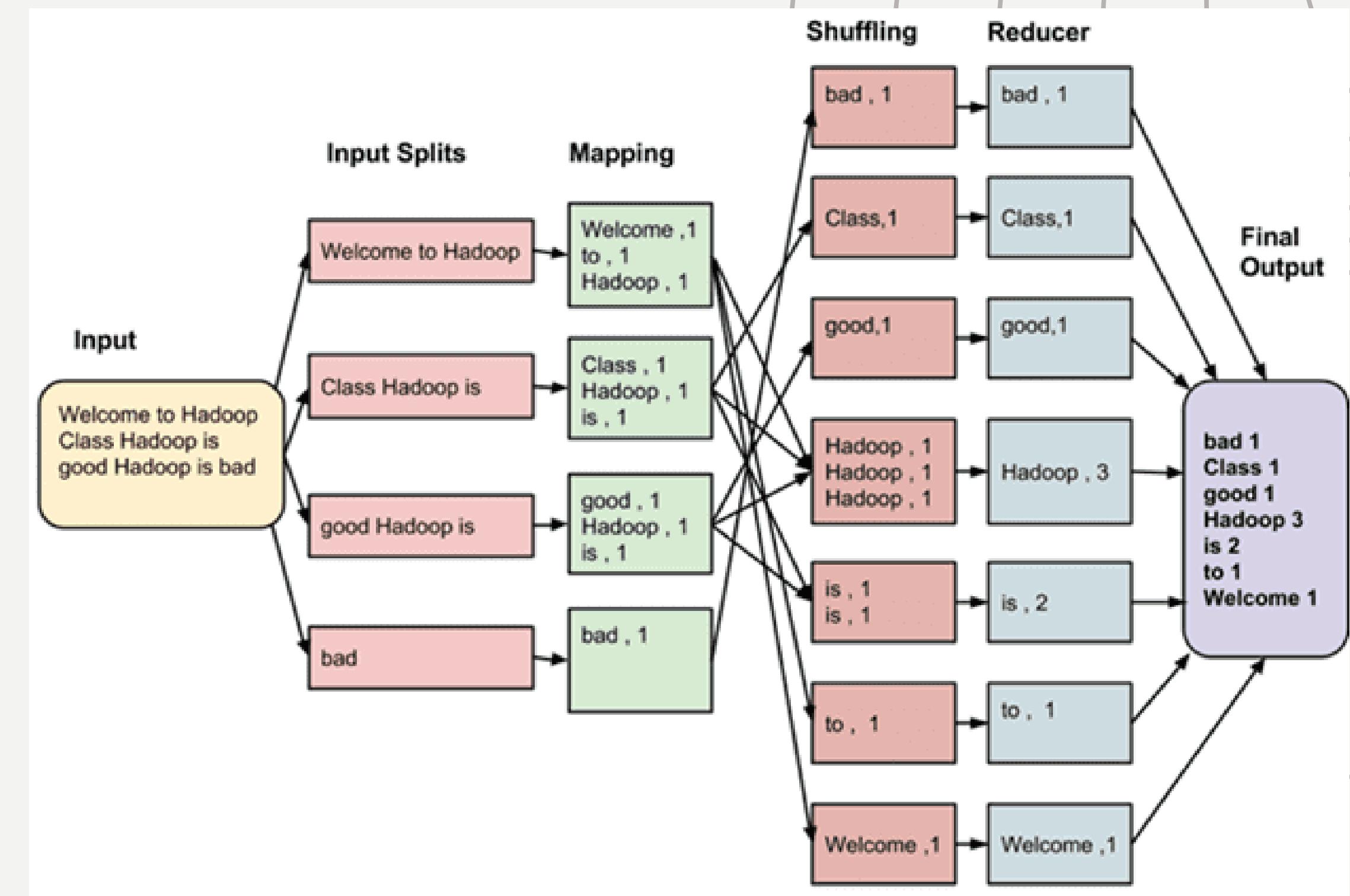
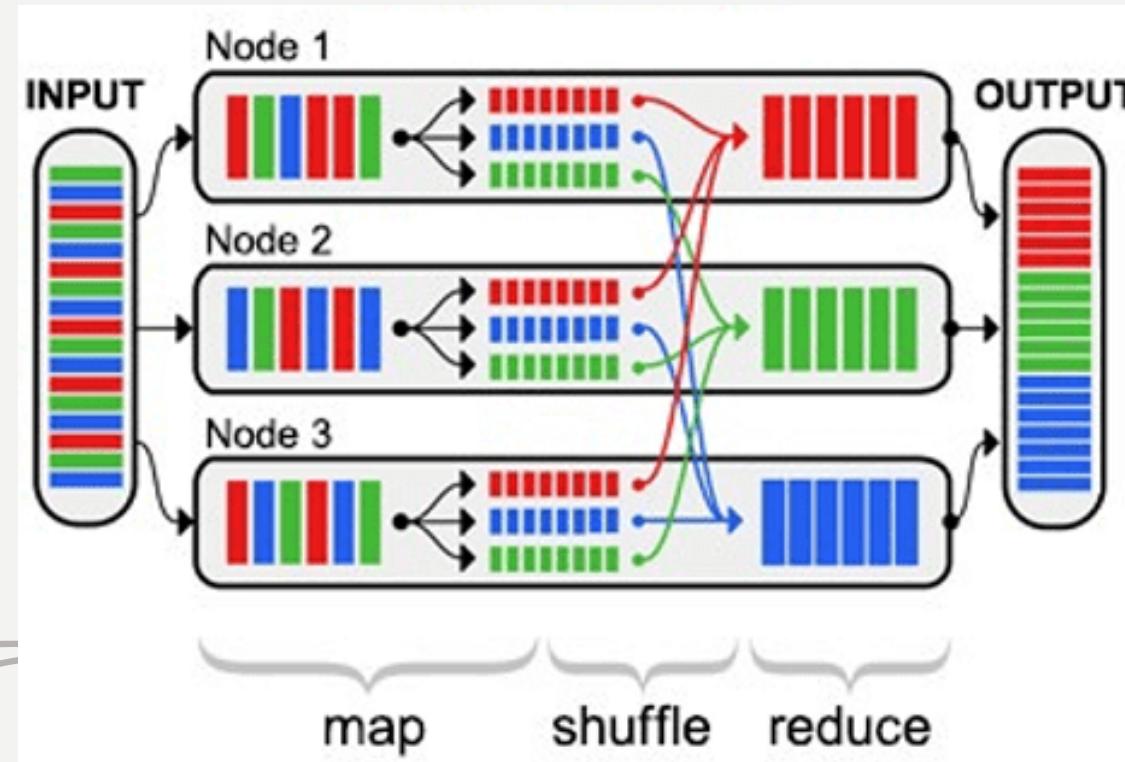
MapReduce

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

Ч4
Ч4
Ч4
П4
П4
Чв
Б10
Б10
Б10
Чд

MapReduce



Задача HR-аналитики

Анализ текучести кадров с учетом различных факторов



Таблица сотрудников
(employee)

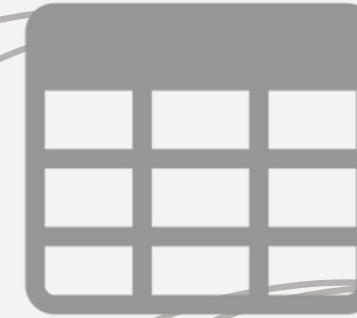


Таблица увольнений
(dismissals)

Основные параметры:

- 1) Возраст
- 2) Должность
- 3) Стаж работы

<https://bigdataschool.ru/wiki/mapreduce>

map:

```
def map_function(employee_filter: dict) -> tuple:  
    key = (employee_filter['age'], employee_filter['job'], employee_filter['exp'])  
    yield key, 1
```

reduce:

```
def reduce_function(key: tuple, values: list) -> tuple:  
    total_resignations = sum(values)  
    yield key, total_resignations
```

output:

```
(возраст: 25-30, должность: менеджер, стаж: 1-2 года) -> 35 увольнений  
(возраст: 30-35, должность: разработчик, стаж: 3-5 лет) -> 12 увольнений  
...
```

Литература

O'REILLY®

ВЫСОКО- НАГРУЖЕННЫЕ ПРИЛОЖЕНИЯ

Программирование
масштабирование
поддержка



ПИТЕР®

Мартин Клеппман

БИБЛИОТЕКА СКРИПАНКА

Джудит Гуревич, Алих Ньюдомит,
Ферн Халлер, Мартин Клеппман

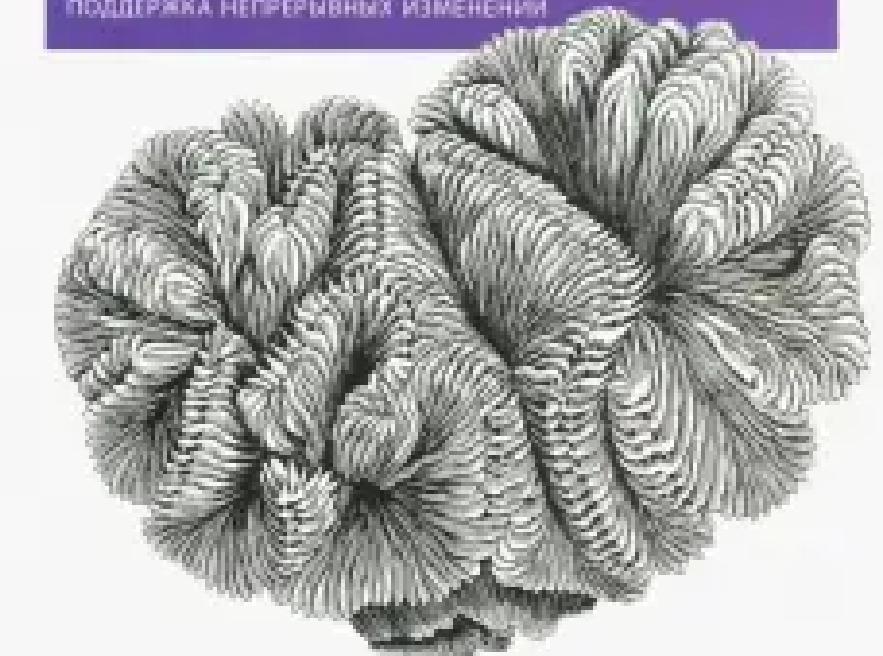
Просто о больших данных

СКРИПАНКА

O'REILLY®

ЭВОЛЮЦИОННАЯ АРХИТЕКТУРА

ПОДДЕРЖКА НЕПРЕРЫВНЫХ ИЗМЕНЕНИЙ



Нил Форд, Ребекка Парсонс, Патрик Куа

ПИТЕР®

Кратко резюмируем

OLTP vs OLAP

БД -> СУБД -> DWH -> Data Lake

ETL vs ELT

Виды данных

Кластер

MapReduce

Что будет на следующем занятии

Экосистема Hadoop

Apache Spark

Практика

НИУ ВШЭ

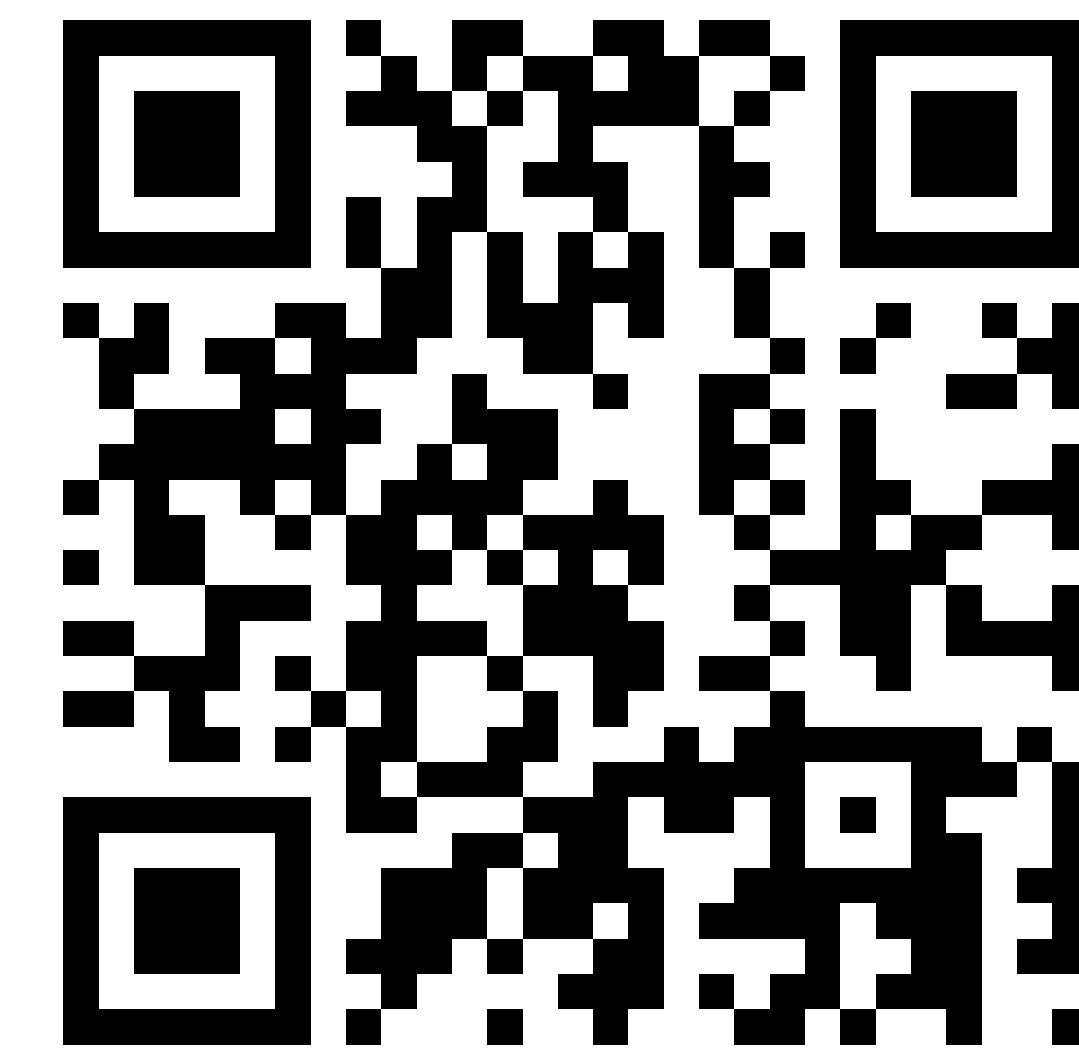
Спасибо за
внимание!

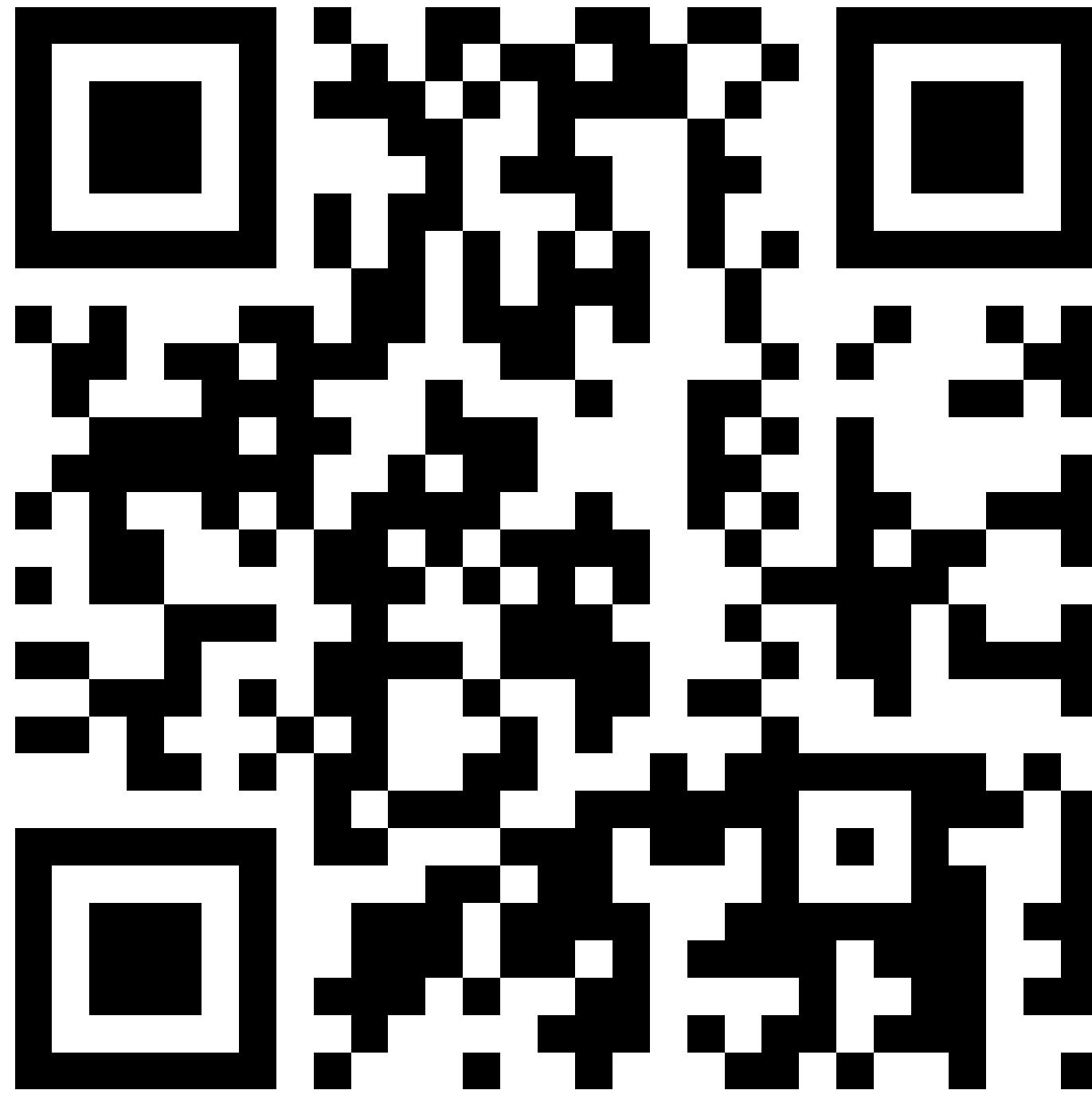
Вопросы

Отметь активность



Отметь посещение





Отметьте посещение

НИУ ВШЭ

Современные архитектуры БД (BigData)

Сысоев Кирилл Романович

Обо мне

**4.5+ лет в BigData
HSE University**

Senior Data Engineer
1) GlowByte Consulting
2) PochtaTech
3) OneFactor

Hadoop, Spark, Kafka, k8s, YandexCloud
Python/Scala, SQL



t.me/KRSysoev
ksysoev@hse.ru

Вспомним

OLTP vs OLAP

БД -> СУБД -> DWH -> Data Lake

ETL vs ELT

Характеристика BigData

Виды данных

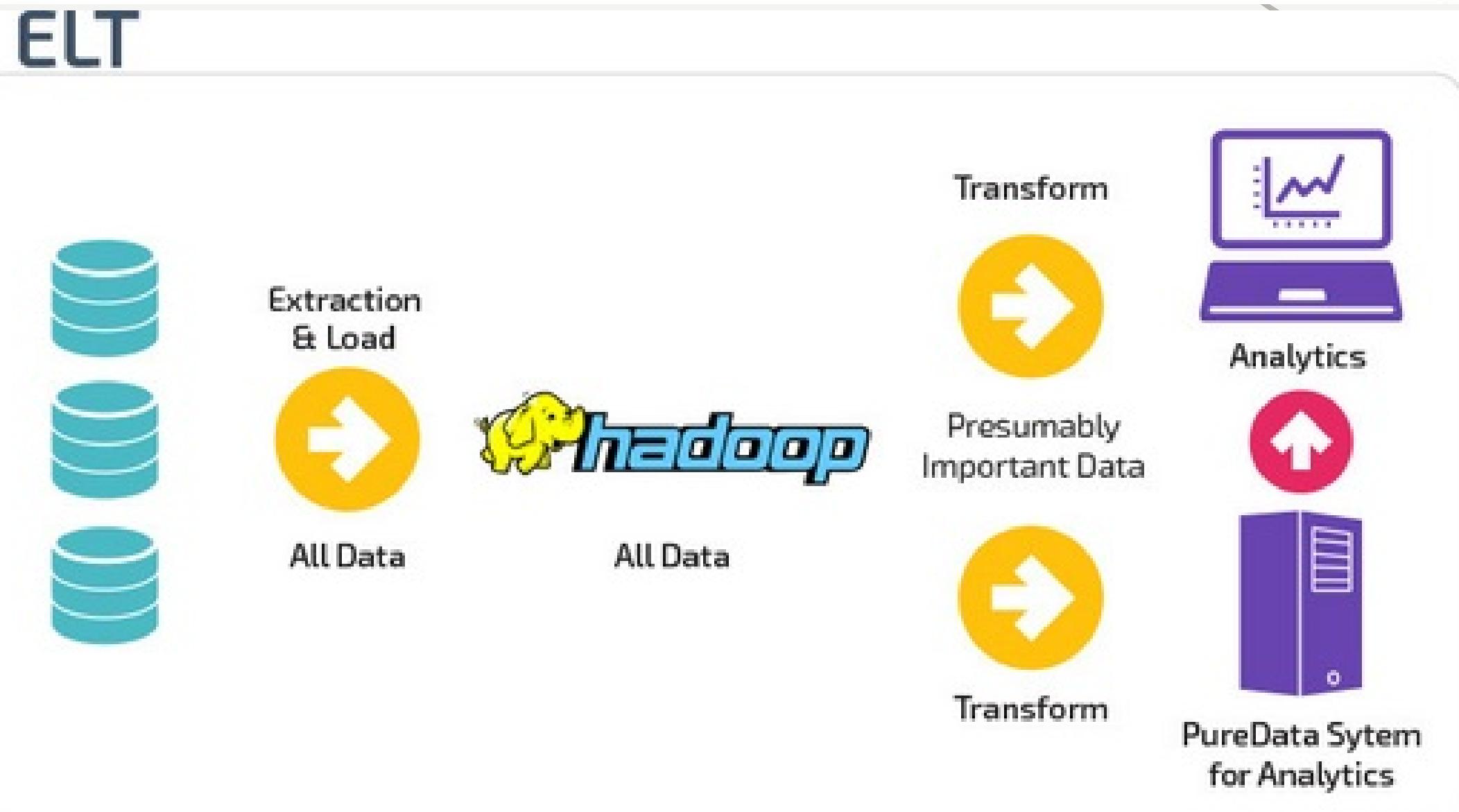
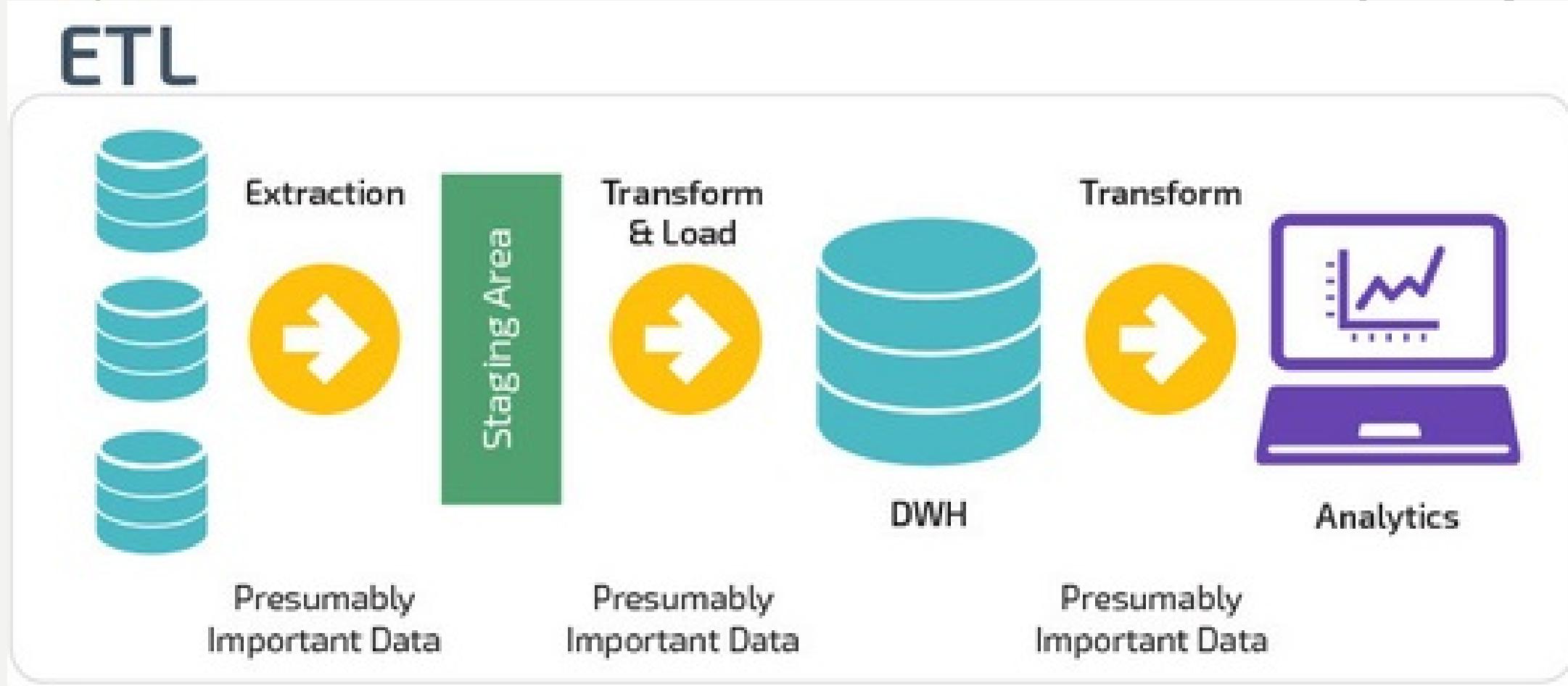
Кластер

MapReduce

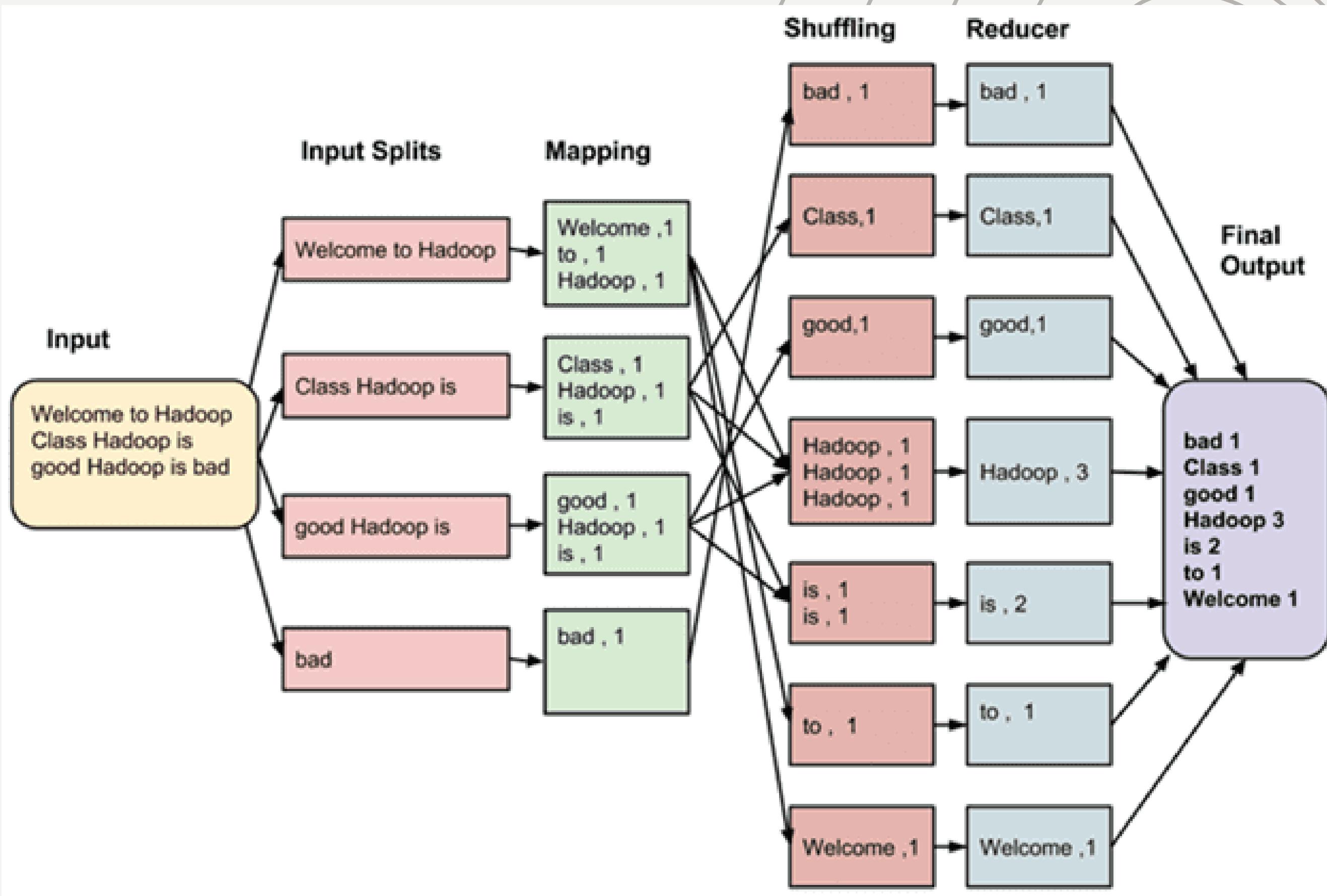
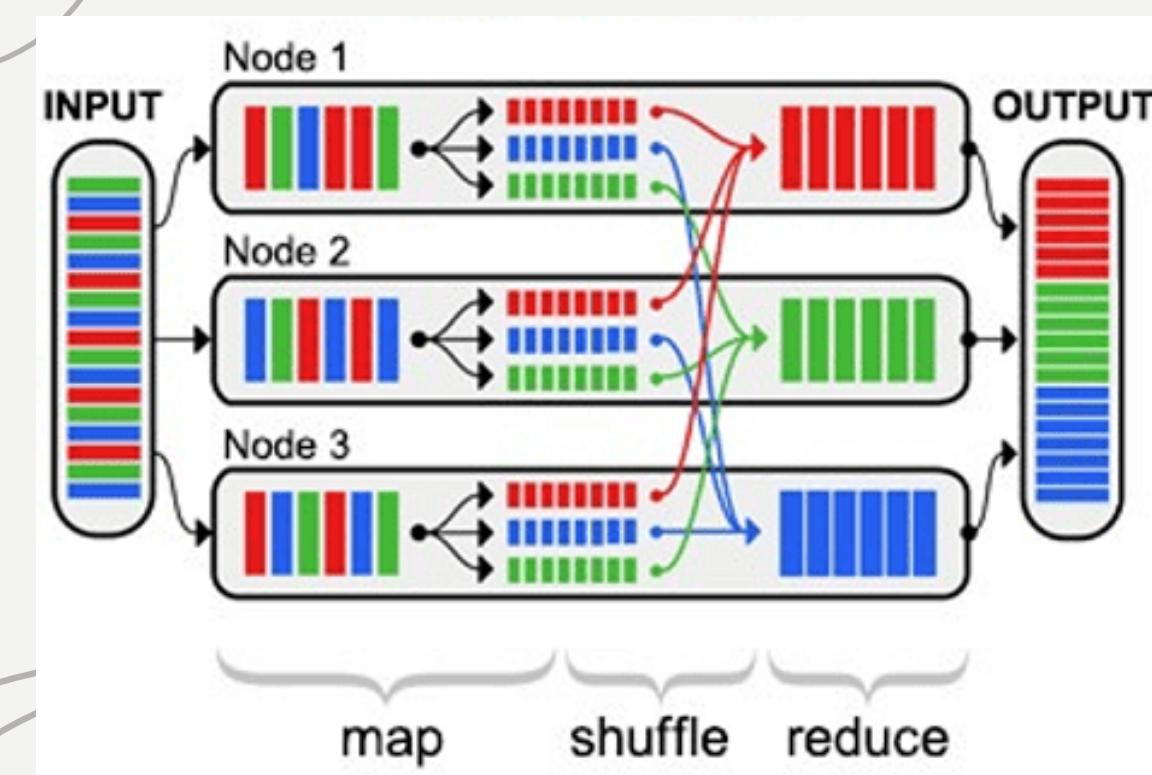
О ЧЕМ ПОГОВОРИМ

- Экосистема Hadoop
- Apache Spark
- Практика

ETL VS ELT



MapReduce

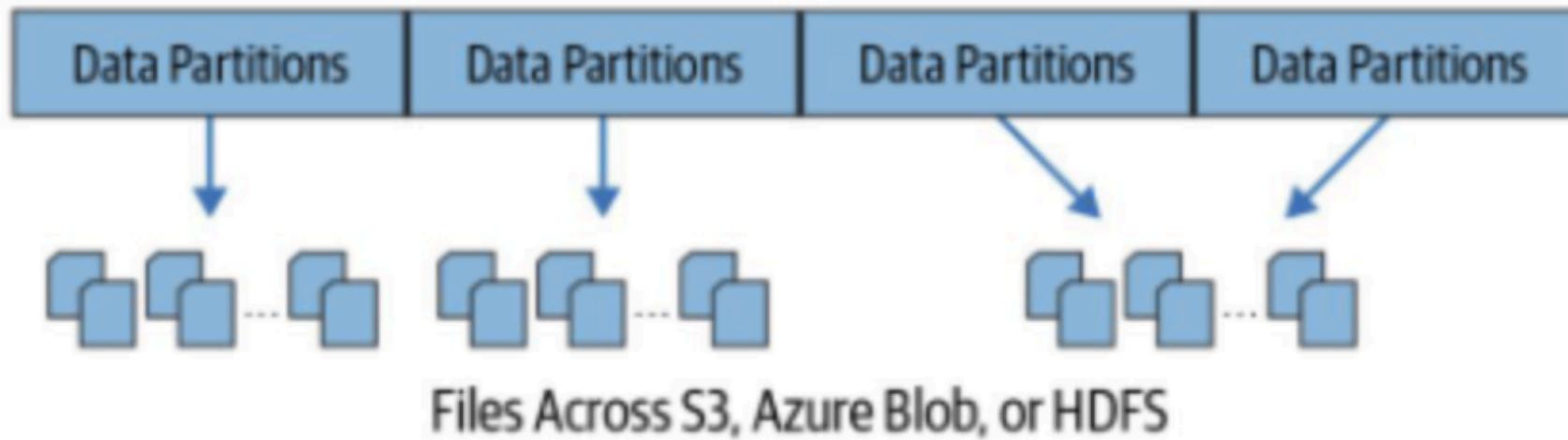


BigData

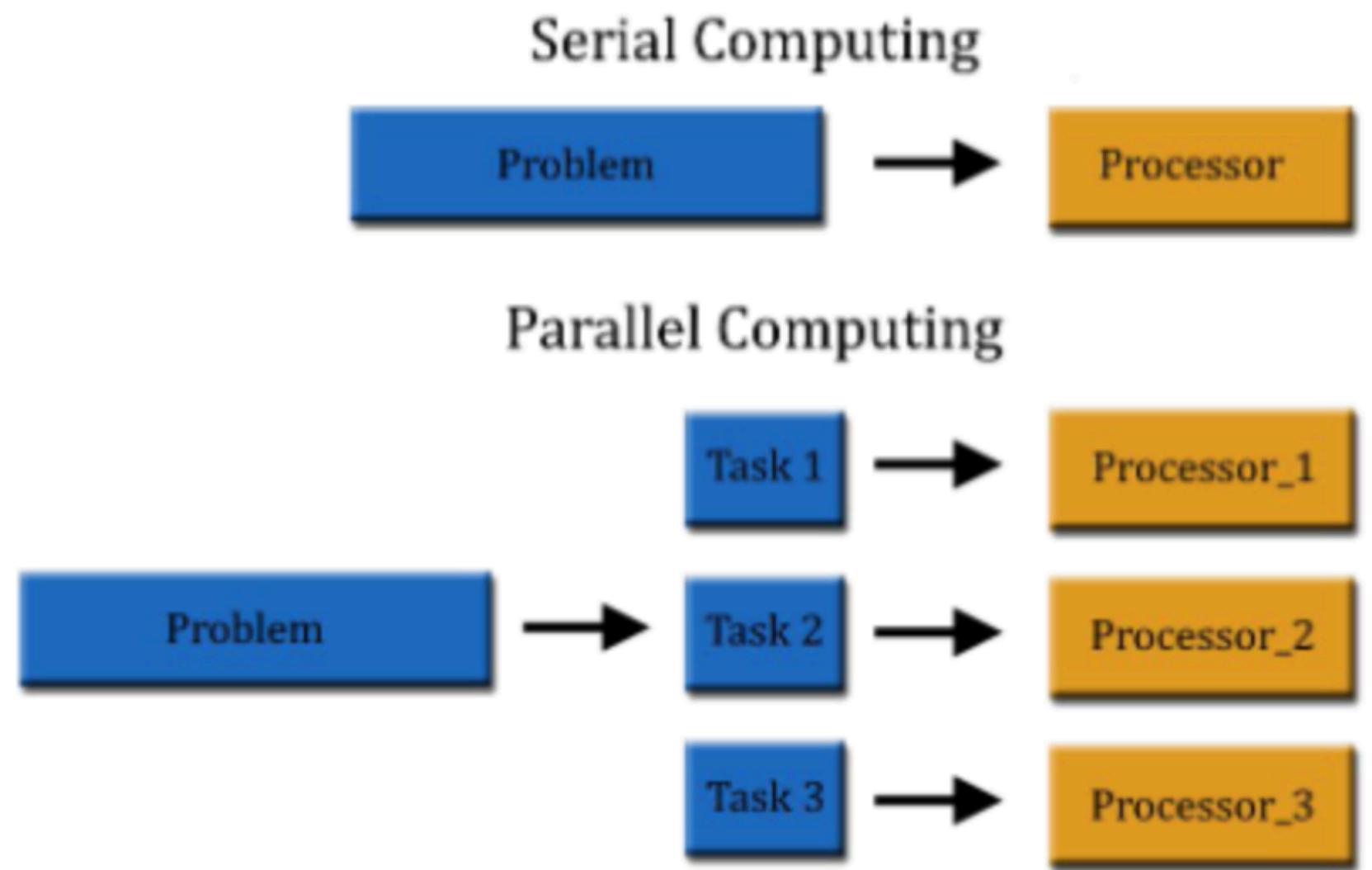
BigData

- не влезают на одну машину

Распределенное хранилище



Параллельные вычисления



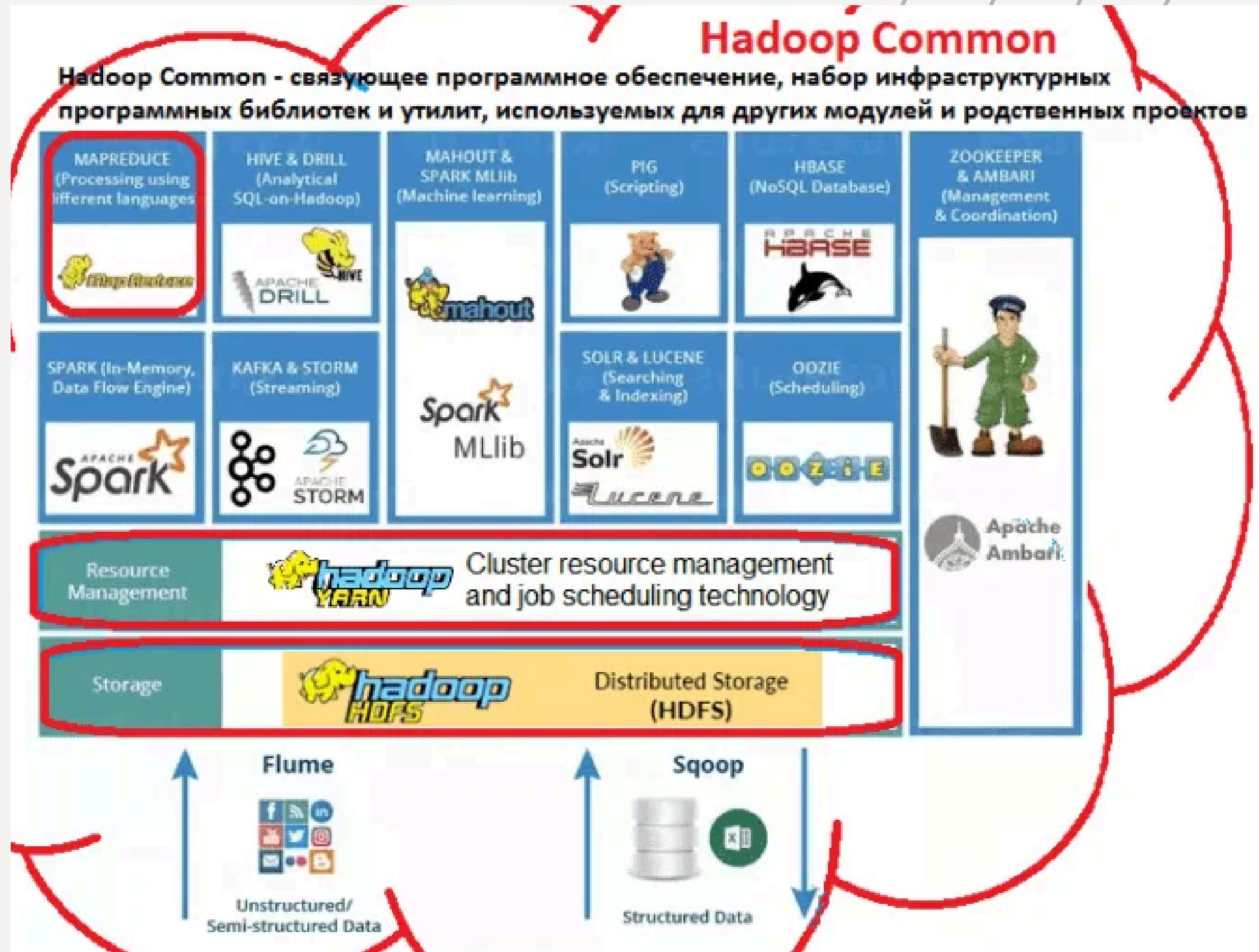
Компоненты системы в BigData

Управление ресурсами кластера (Resource management layer)

Хранение данных (Storage Layer)

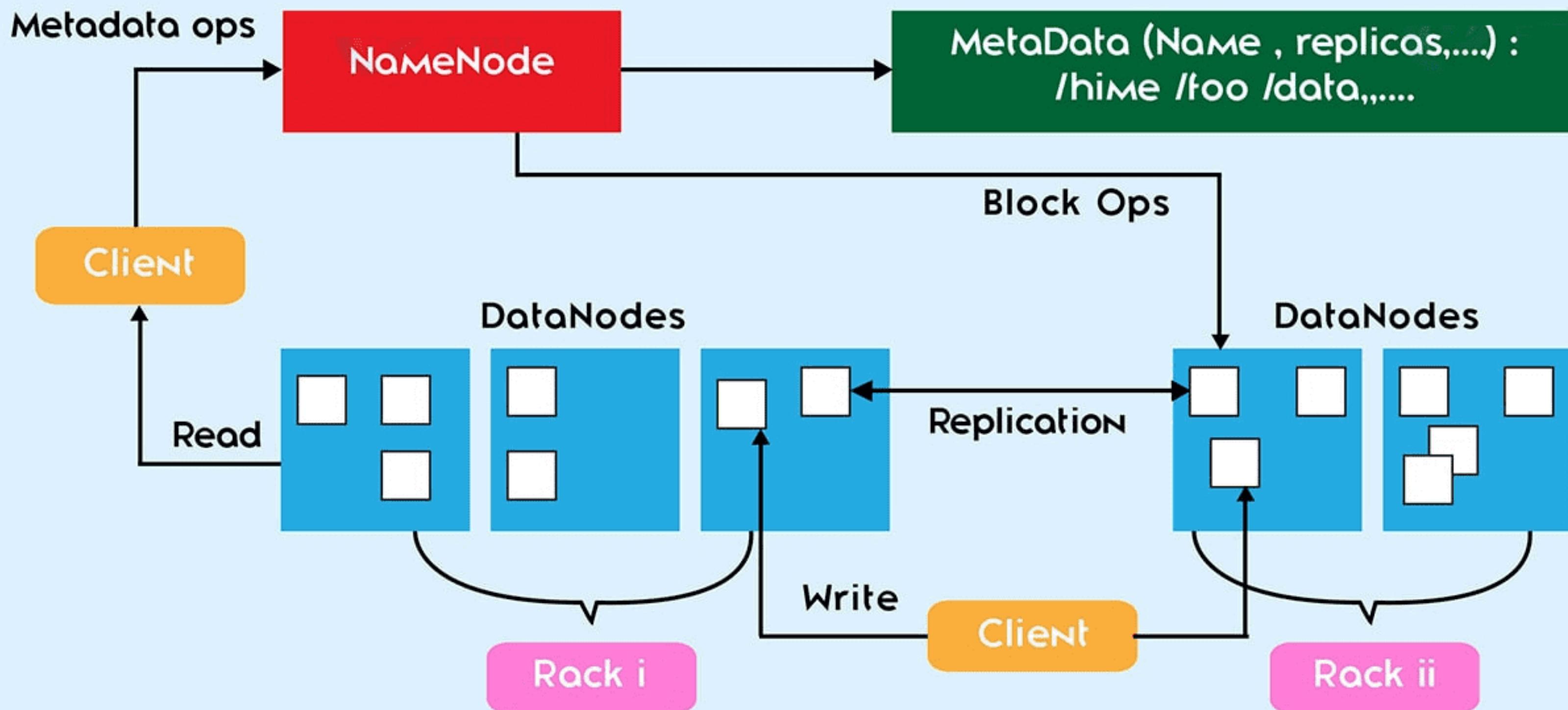
Алгоритм трансформации (Processing layer)

Hadoop



HDFS

HDFS Architecture



Hive

The screenshot shows the Hue web interface. At the top, there's a navigation bar with the Hue logo, a 'Query' button, and a search bar. Below the navigation bar, there's a dropdown menu with options: '</> Editor' (selected), 'Scheduler', and a list of data sources. The 'Hive' option in this list is highlighted with a red box. To the right of the dropdown, there's a list of other data sources: Impala, Hive, Pig, Java, Spark, MapReduce, Shell, Sqoop 1, and Distcp. On the left side of the interface, there's a sidebar titled 'Sources' containing links for Impala and Hive.

Query

Search saved documents...

</> Editor

Scheduler

Impala

Hive

Pig

Java

Spark

MapReduce

Shell

Sqoop 1

Distcp

Impala

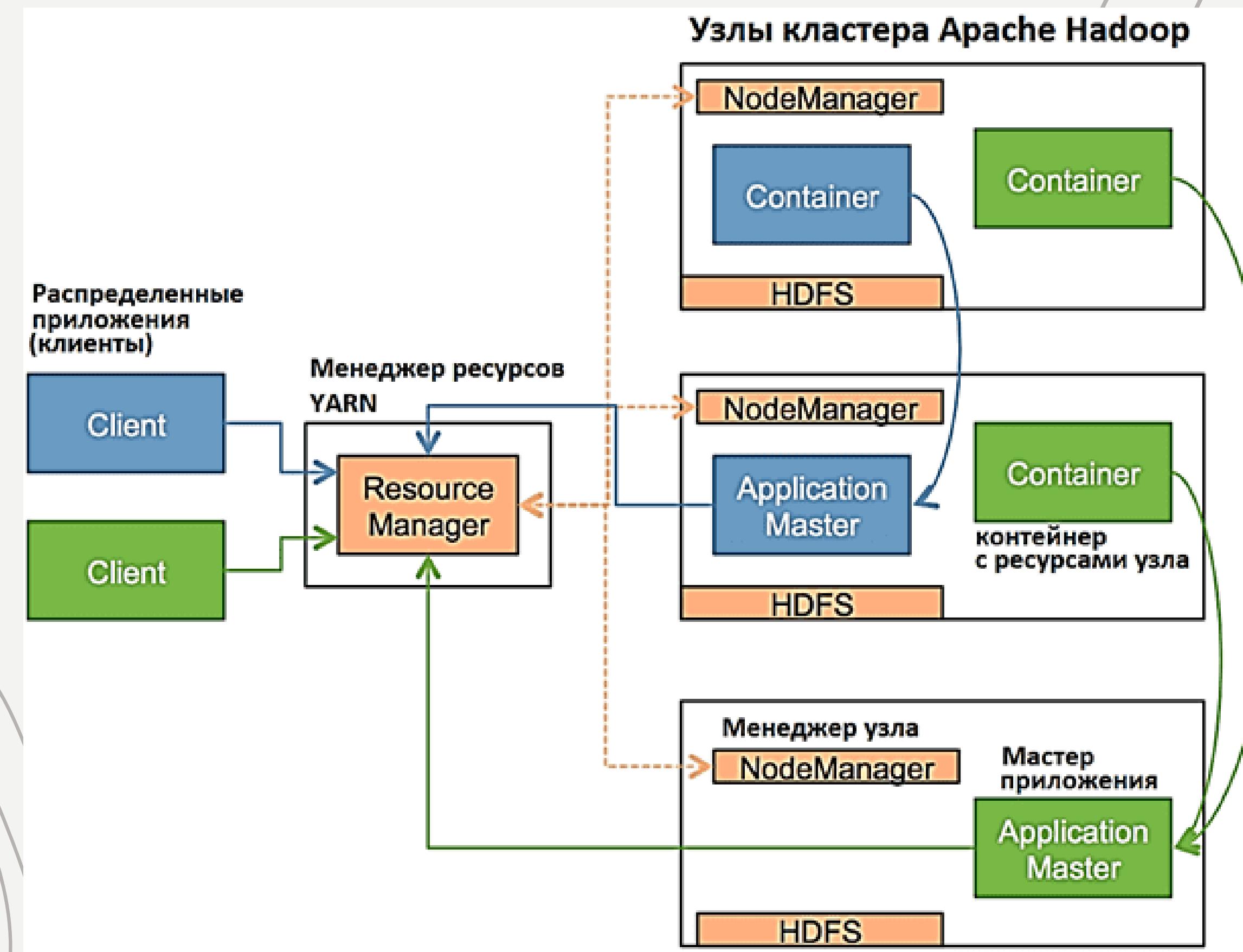
Hive

Sources

Impala

Hive

YARN



YARN (RM UI)

← → C psrlInfa.informatica.com:8088/cluster

 All Applications

Cluster Metrics

Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running	Memory Used	Memory Total	Memory Reserved	Vcores Used	Vcores Total	Vcores Reserved	Active Nodes	Decom N
568	0	0	568	0	0 B	32 GB	0 B	0	32	0	1	0

User Metrics for dr.who

Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running	Containers Pending	Containers Reserved	Memory Used	Memory Pending	Memory Reserved
0	0	0	568	0	0	0	0 B	0 B	0 B

Show 20 ▾ entries

ID	User	Name	Application Type	Queue	StartTime	FinishTime	State	FinalStatus	Running Containers	Allocated CPU Vcores
application_1463379223882_0568	Idmui	InfaSprk0	SPARK	root.Idmui	Mon May 16 13:11:59 -0700 2016	Mon May 16 13:13:03 -0700 2016	FINISHED	SUCCEEDED	N/A	N/A
application_1463379223882_0567	Idmui	InfaSprk0	SPARK	root.Idmui	Mon May 16 13:11:58 -0700 2016	Mon May 16 13:13:01 -0700 2016	FINISHED	SUCCEEDED	N/A	N/A
application_1463379223882_0566	Idmui	InfaSprk0	SPARK	root.Idmui	Mon May 16 13:11:56 -0700 2016	Mon May 16 13:13:00 -0700 2016	FINISHED	SUCCEEDED	N/A	N/A
application_1463379223882_0565	Idmui	InfaSprk0	SPARK	root.Idmui	Mon May 16 13:11:55 -0700 2016	Mon May 16 13:12:59 -0700 2016	FINISHED	SUCCEEDED	N/A	N/A
application_1463379223882_0564	Idmui	InfaSprk0	SPARK	root.Idmui	Mon May 16 13:11:54	Mon May 16 13:12:59	FINISHED	SUCCEEDED	N/A	N/A

YARN (RM UI)

The screenshot shows the Hadoop Resource Manager (RM) User Interface. At the top, there is a logo of a yellow elephant and the word "hadoop". Below it, the title "Application application_1484222082388_0001" is displayed. On the left, a sidebar titled "Cluster" contains links for "About", "Nodes", "Node Labels", "Applications" (with sub-links for "NEW", "NEW_SAVING", "SUBMITTED", "ACCEPTED", "RUNNING", "FINISHED", "FAILED", "KILLED"), and "Scheduler". Below this is a "Tools" section. The main content area is titled "Kill Application" and displays the following application details:

Application	
User:	hdfs
Name:	org.apache.spark.examples.SparkPi
Application Type:	SPARK
Application Tags:	
Application Priority:	0 (Higher Integer value indicates higher priority)
YarnApplicationState:	FINISHED
Queue:	default
FinalStatus Reported by AM:	SUCCEEDED
Started:	Thu Jan 12 17:48:04 +0530 2017
Elapsed:	27sec
Tracking URL:	History
Log Aggregation Status:	SUCCEEDED
Diagnostics:	
Unmanaged Application:	false
Application Node Label expression:	<Not set>
AM container Node Label expression:	<DEFAULT_PARTITION>

Below this, another table provides resource usage statistics:

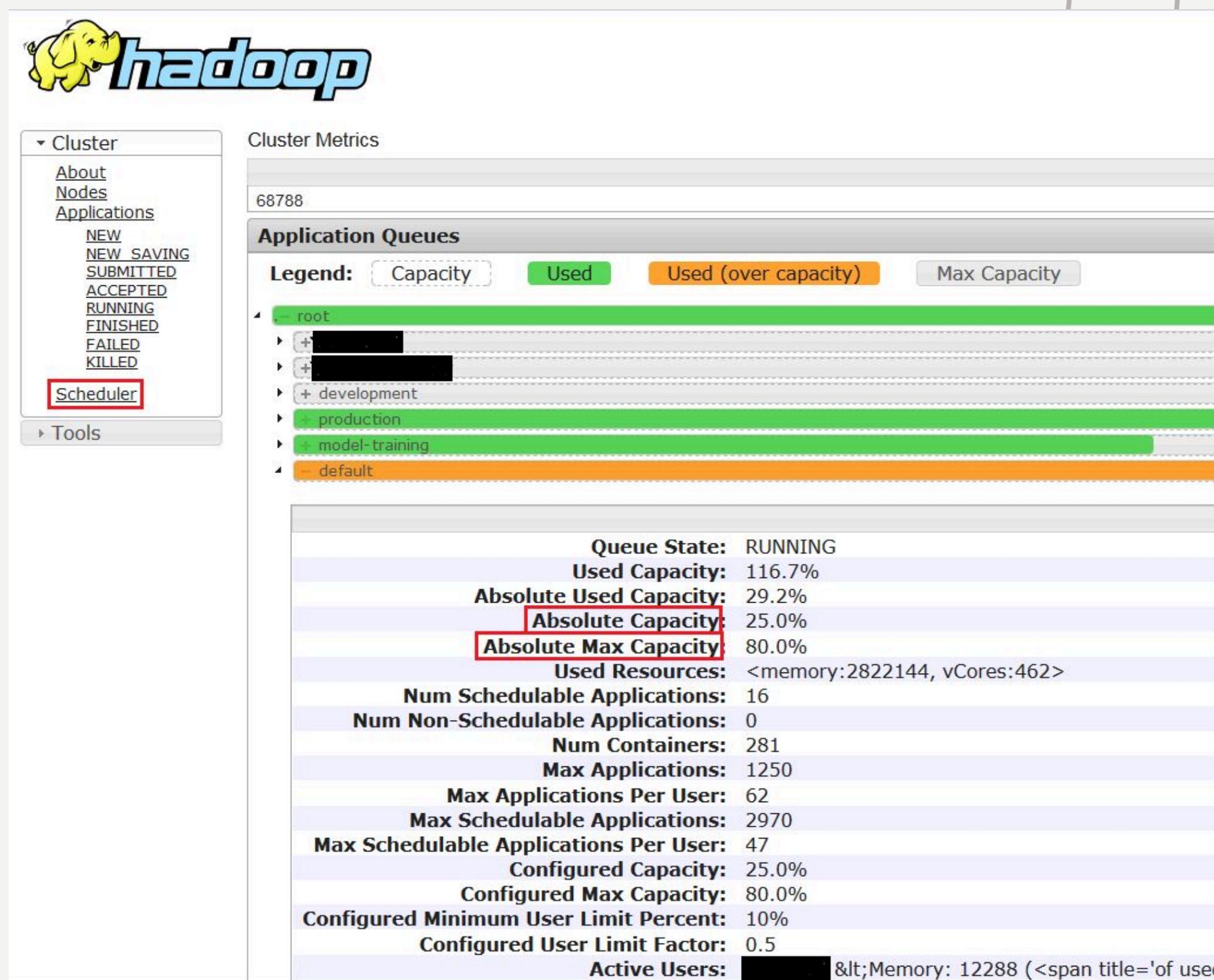
Application	
Total Resource Preempted:	<memory:0, vCores:0>
Total Number of Non-AM Containers Preempted:	0
Total Number of AM Containers Preempted:	0
Resource Preempted from Current Attempt:	<memory:0, vCores:0>
Number of Non-AM Containers Preempted from Current Attempt:	0
Aggregate Resource Allocation:	257551 MB-seconds, 76 vcore-seconds

At the bottom, there is a search bar and a table header for a list of attempts:

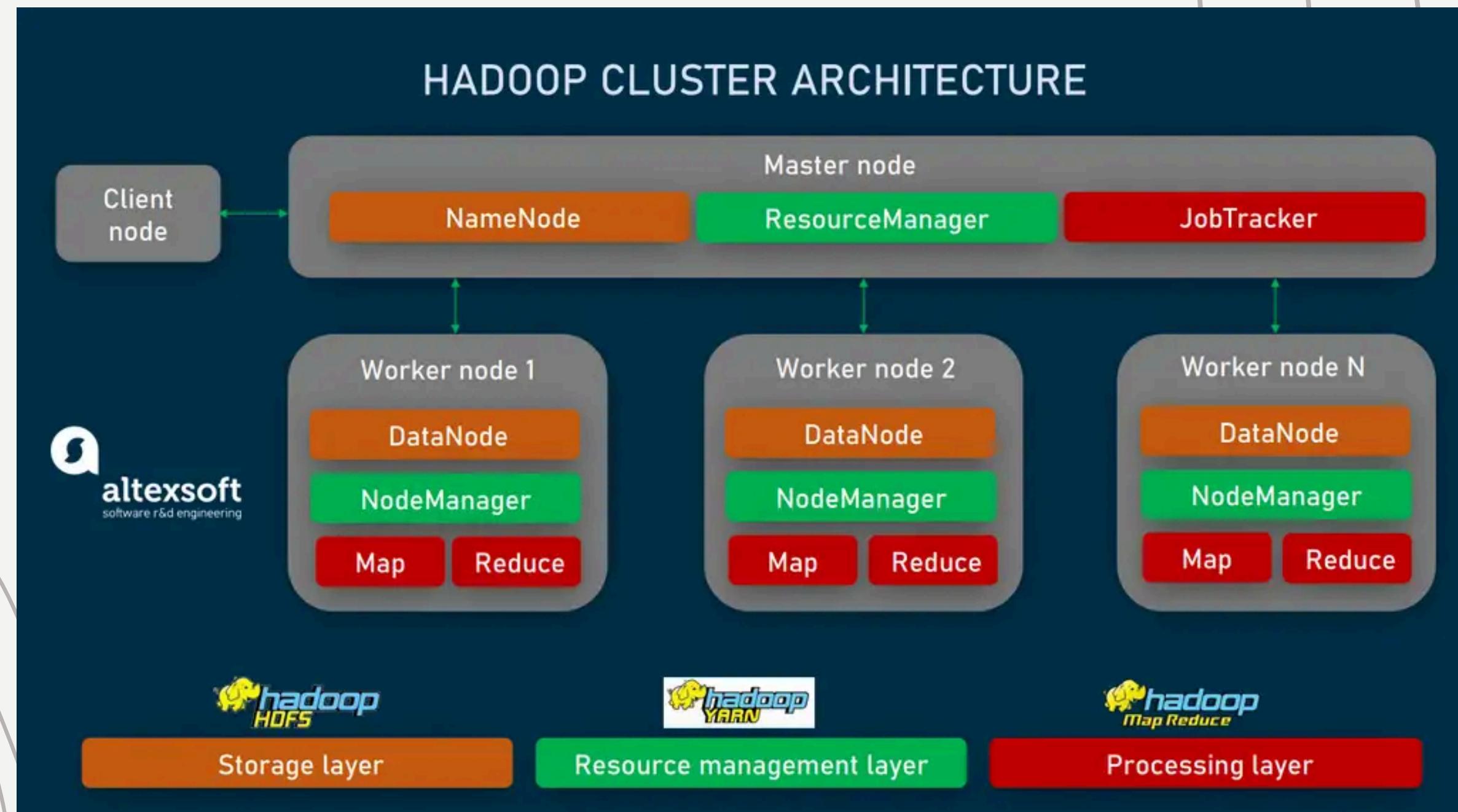
Attempt ID	Started	Node	Logs	Blacklisted Node
------------	---------	------	------	------------------

Navigation controls at the bottom include "Show 20 entries" and a "Search:" field.

YARN (RM UI)



Кластер Hadoop



Запуск задания MapReduce

map:

```
def map_function(employee_filter: dict) -> tuple:  
    key = (employee_filter['age'], employee_filter['job'], employee_filter['exp'])  
    yield key, 1
```

reduce:

```
def reduce_function(key: tuple, values: list) -> tuple:  
    total_resignations = sum(values)  
    yield key, total_resignations
```

output:

```
(возраст: 25-30, должность: менеджер, стаж: 1-2 года) -> 35 увольнений  
(возраст: 30-35, должность: разработчик, стаж: 3-5 лет) -> 12 увольнений  
...
```

**Как запустить алгоритм
расчета в Hadoop:**

- 1) ssh hadoop.r1.master:9899
- 2) hadoop jar /path/to/hadoop-streaming.jar \
-input /input -output /output \
-mapper mapper.py -reducer reducer.py

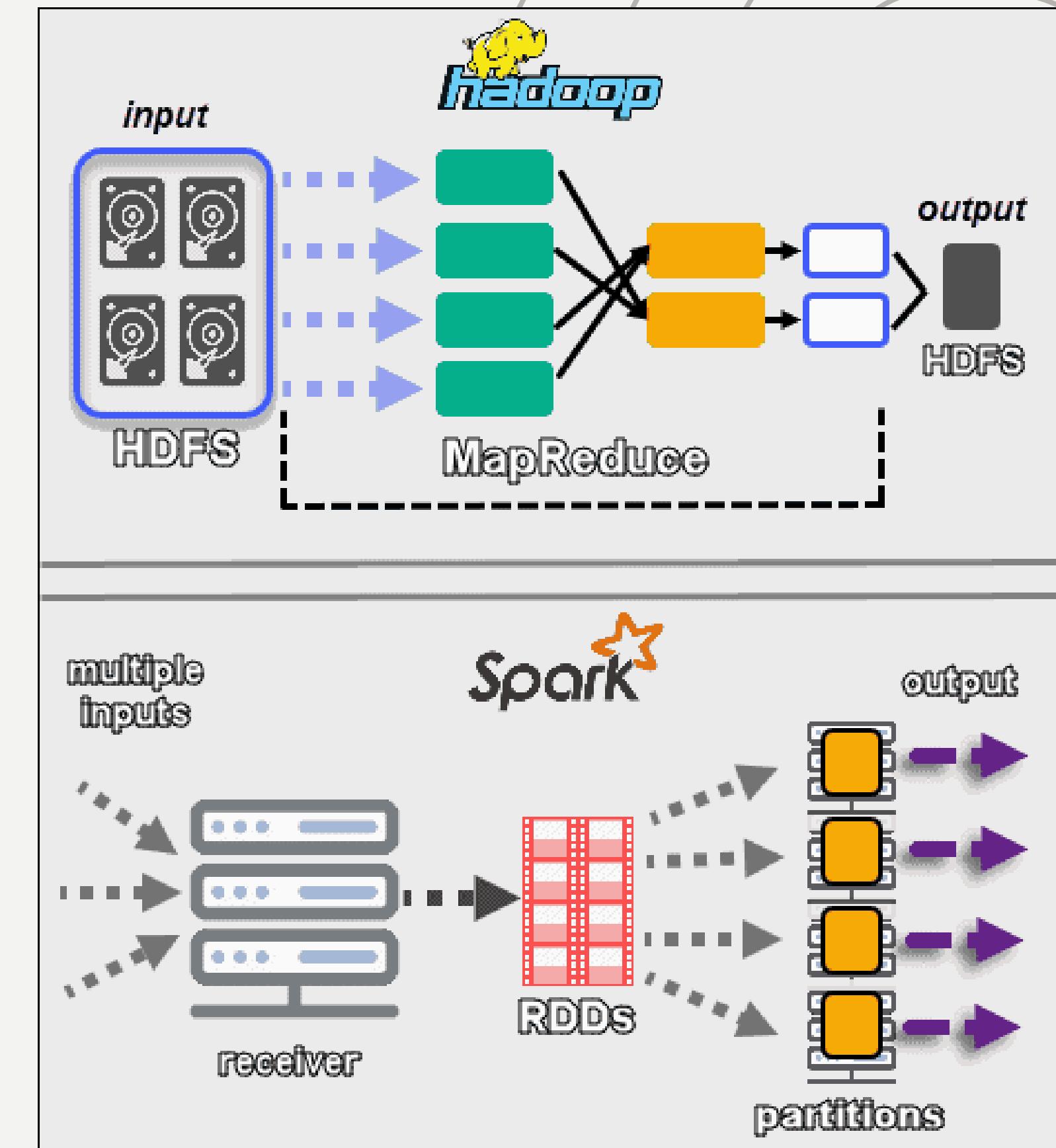
MapReduce vs Spark



MapReduce vs Spark

1) Производительность (x100)

MapReduce записывает данные на диск между шагами
Spark хранит промежуточные результаты вычислений в оперативной памяти (in-memory) + встроенная оптимизация



MapReduce vs Spark

2) Гибкость

MapReduce требует более

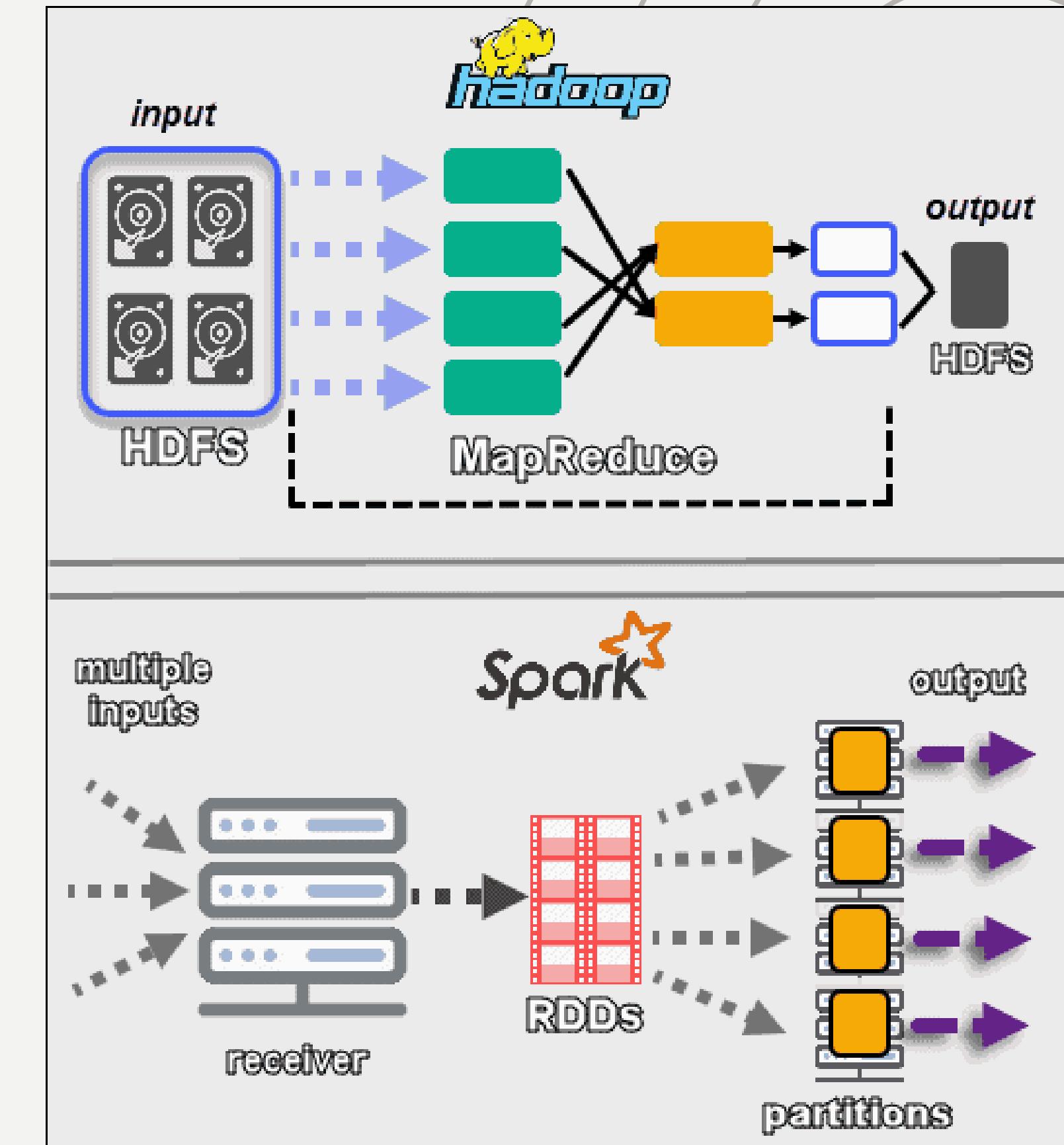
низкоуровневого

программирования

Spark API предоставляет

использование на ЯП: Scala, Python,

Java, R

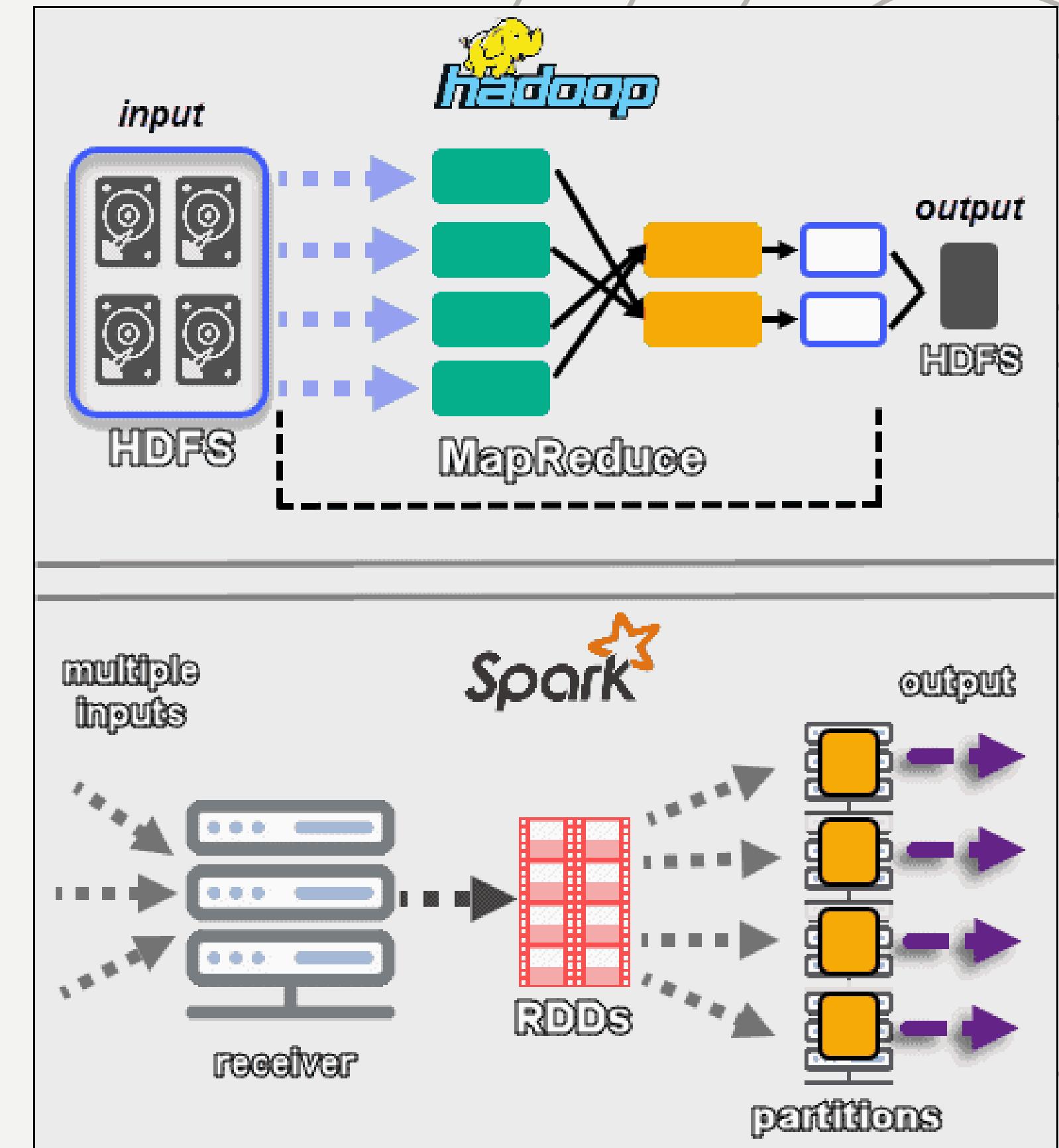


MapReduce vs Spark

3) Удобство разработки

Spark предлагает абстракцию RDD и DataFrame для более высокогоуровневой трансформации данных

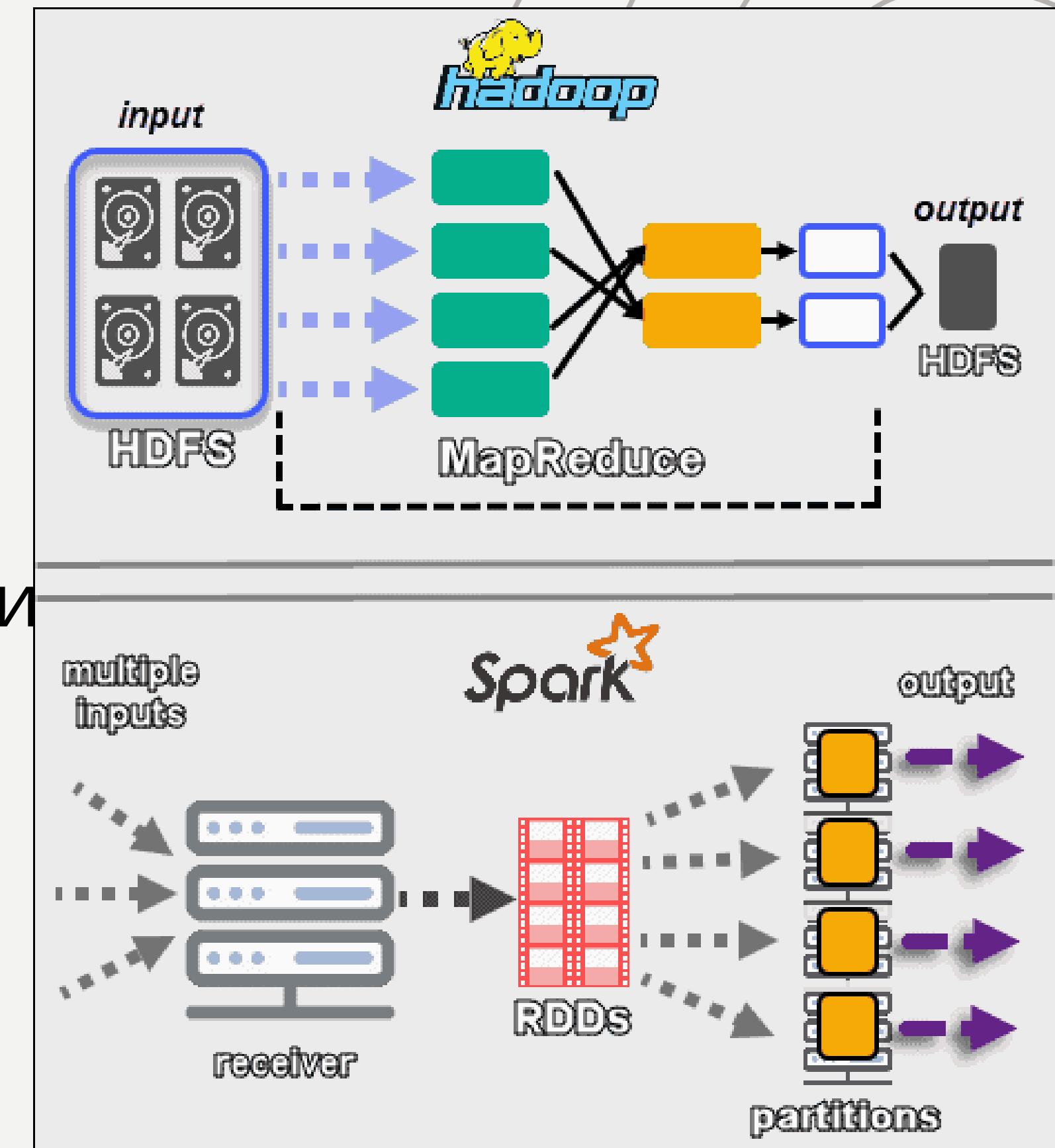
В **Spark** можно запускать задачи через интерфейсы: Spark Shell или Jupyter Notebooks



MapReduce vs Spark

4) Экосистема

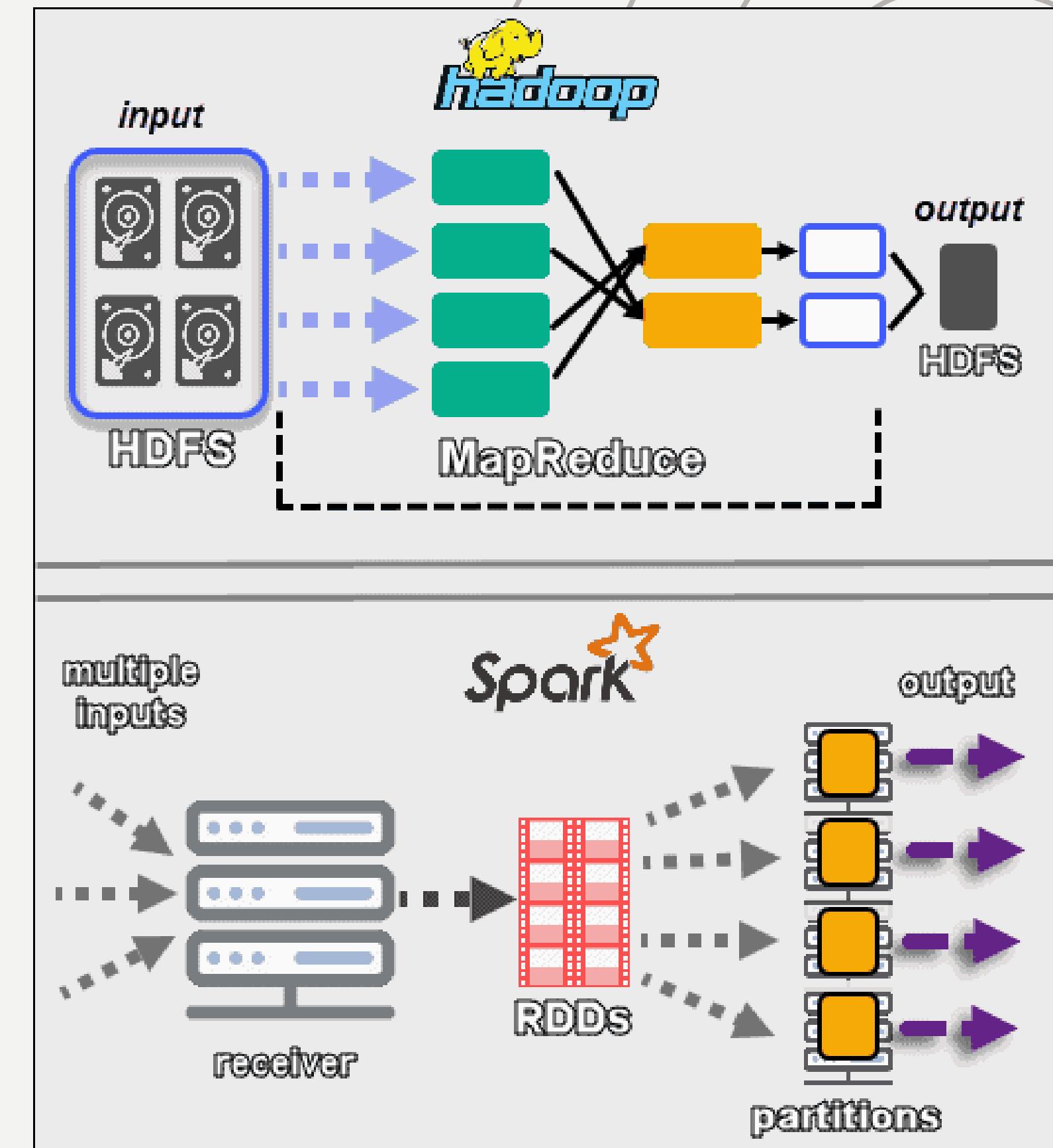
- **Spark SQL:** для работы с данными в стиле реляционных БД.
- **MLlib:** для машинного обучения.
- **GraphX:** для работы с графами.
- **Spark Structured Streaming:** для более продвинутой потоковой обработки.



MapReduce vs Spark

5) Обработка данных в реальном времени

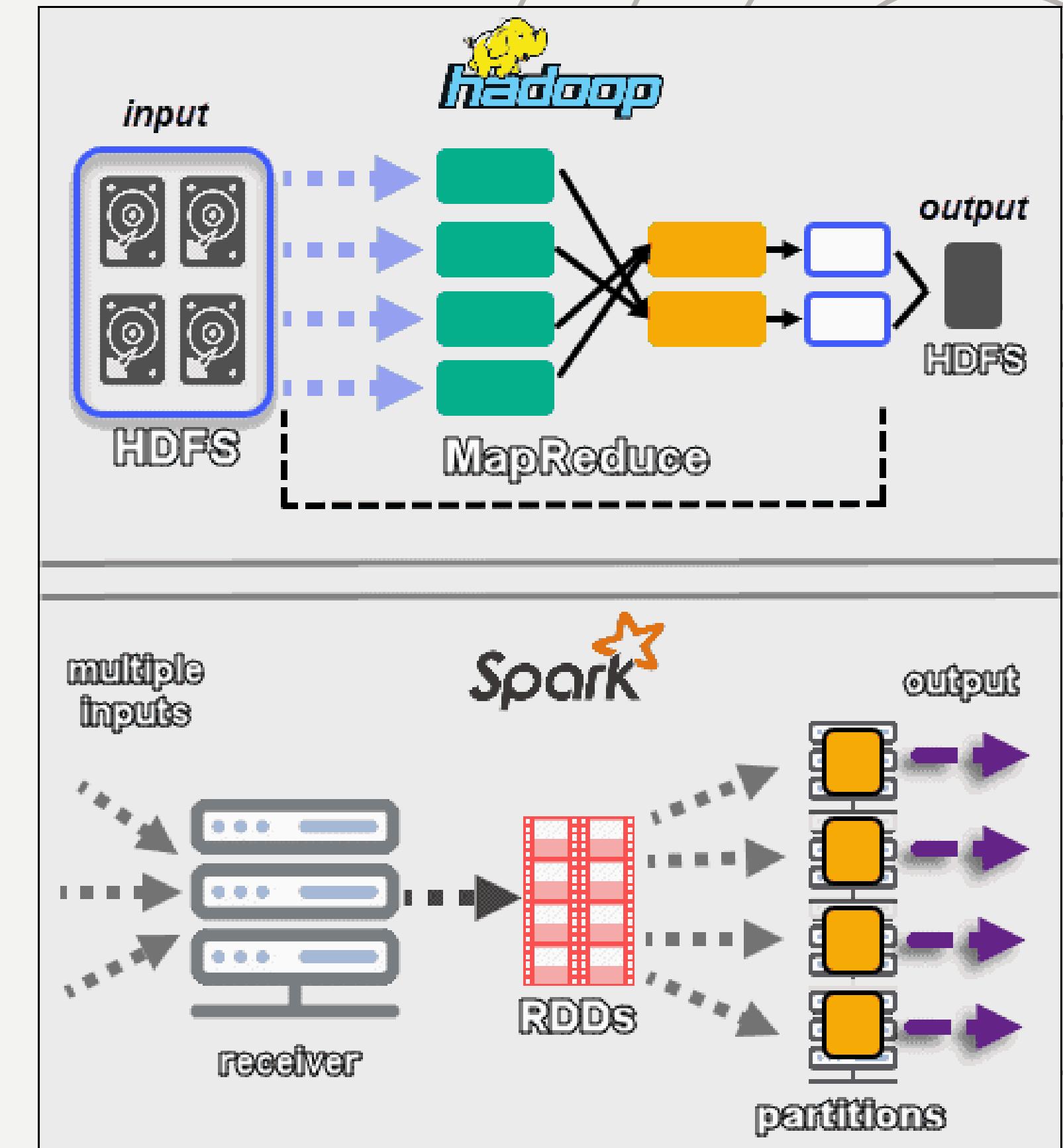
Spark дает возможность работать со стримингом данных



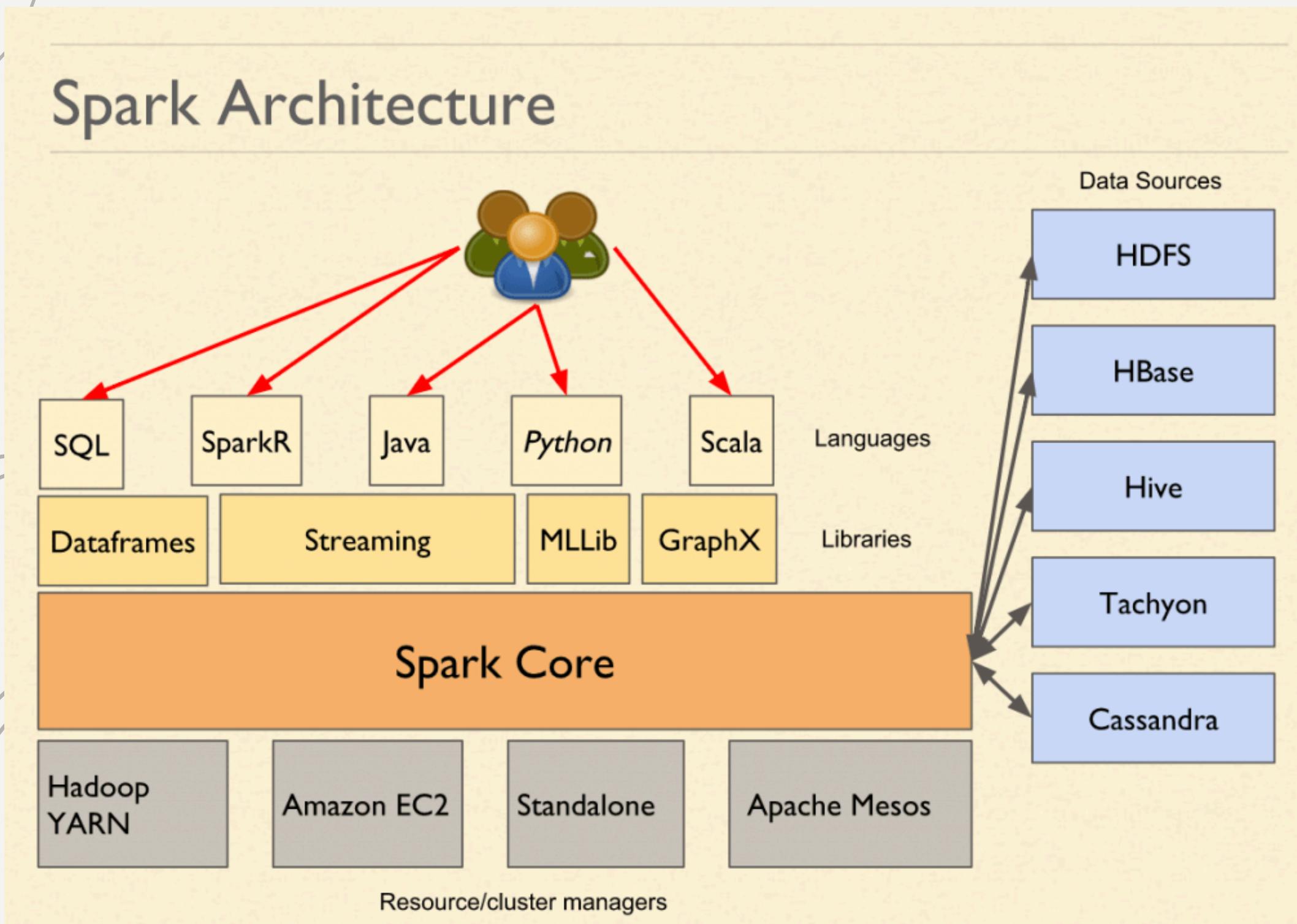
MapReduce vs Spark

6) Web UI

Spark имеет встроенный веб-интерфейс (Web UI), который предоставляет информацию о текущих и завершенных заданиях, стадии выполнения, временных задержках, использовании ресурсов и распределении данных



Apache Spark



RDD (Resilient Distributed Datasets)

Устойчивые распределенные наборы данных

- 1) Устойчив к падениям
- 2) Обрабатывается параллельно
- 3) Использует концепцию ленивых вычислений

Apache Spark

RDD (Resilient Distributed Datasets)

Устойчивые распределенные наборы данных

Концепция ленивых вычислений

Трансформации и действия

Трансформации:

- **Lazy** evaluation - выполняется отложенно
 - оптимизацию
- **Immutable** - создает новый Dataset из существующего
 - fault tolerance (отказоустойчивость)

Действия:

- Возвращает результат/записывает на диск
- Триггерит выполнение трансформаций

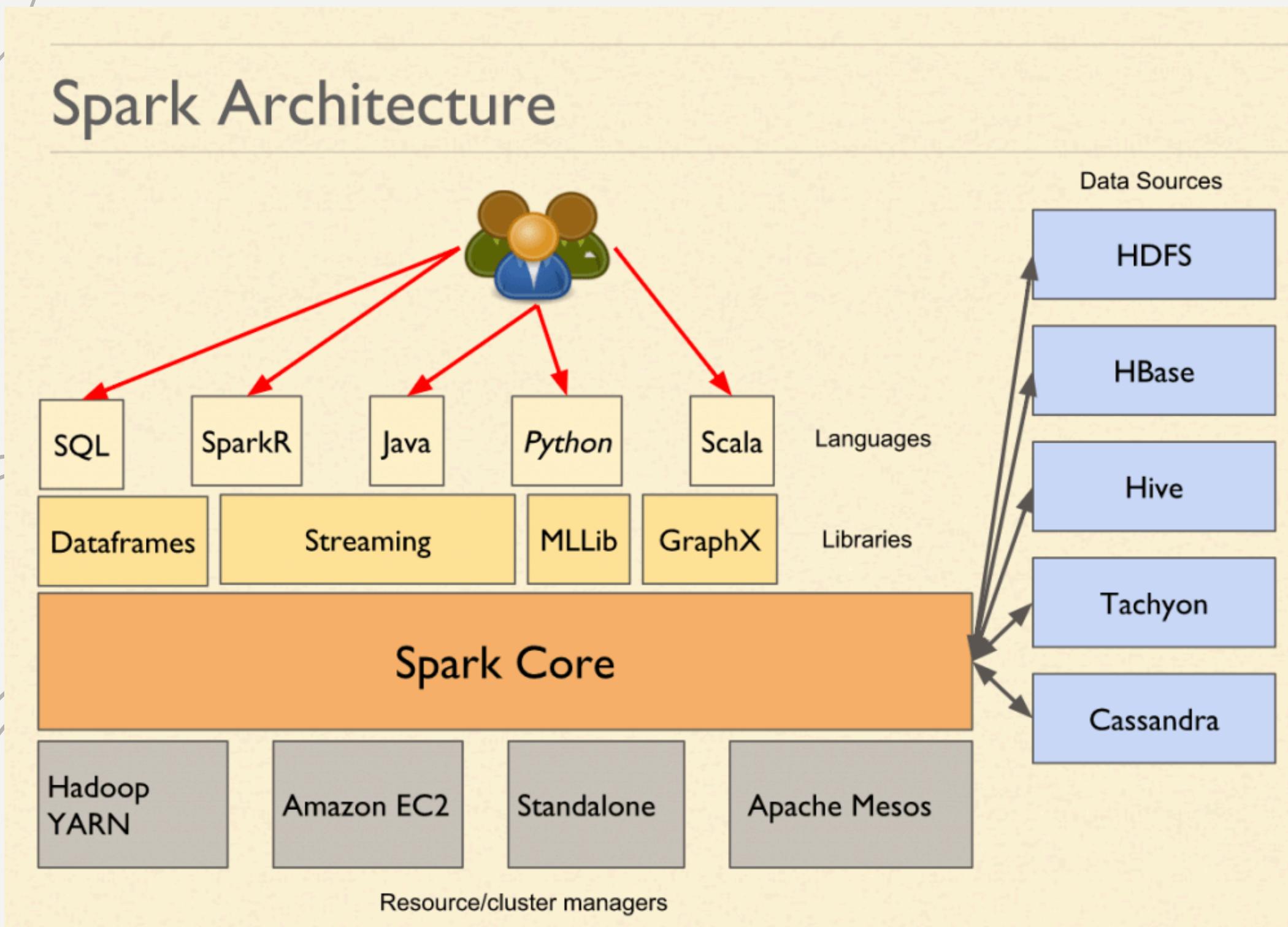
Примеры:

- `orderBy()`
- `groupBy()`
- `filter()`
- `select()`
- `join()`

Примеры:

- `show()`
- `take()`
- `collect()`
- `count()`
- `save()`

Apache Spark



Apache Spark

Основные определения

Application - приложение, поверх Spark APIs, состоит из **Driver** и **Executors** на кластере

SparkSession - объект - точка входа, позволяющая взаимодействовать со Spark с помощью API

Job - параллельное вычисление, состоящее из нескольких задач, порождаемых в ответ на **action**

Stage - каждая job делится на более мелкие наборы задач, которые зависят друг от друга

Task - **unit of work**, которая будет отправлена на **executor**

Apache Spark

Cluster mode: Driver

Driver - процесс запускает main() и отвечает:

- взаимодействует с Cluster Manager
- запрашивает ресурсы (ЦП, память и т. д.) у Cluster Manager для executors
- преобразует все операции Spark DAG, планирует их и распределяет их выполнение на executors
- запускается в собственном **JVM** процессе

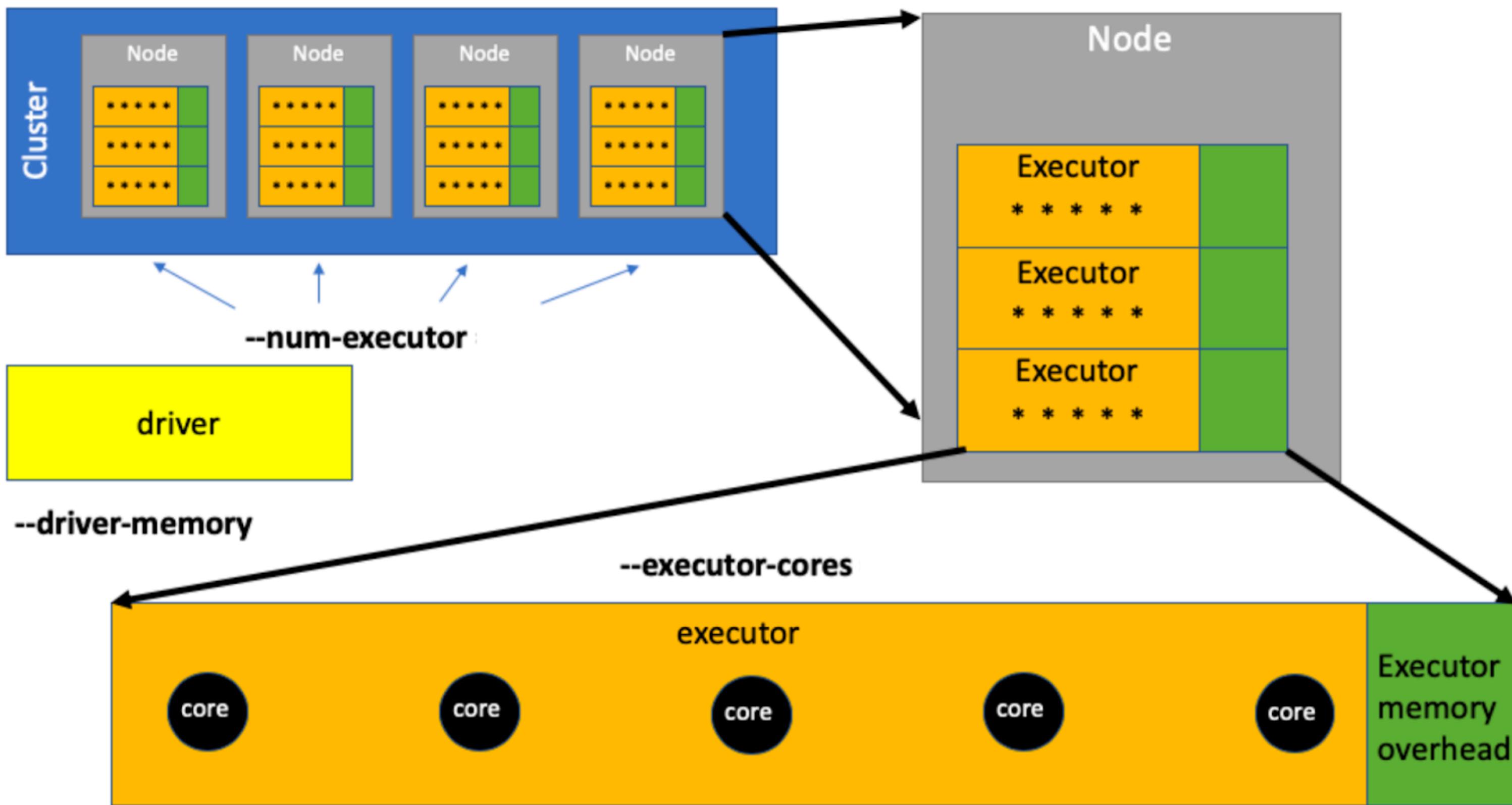
Apache Spark

Cluster mode: Executor

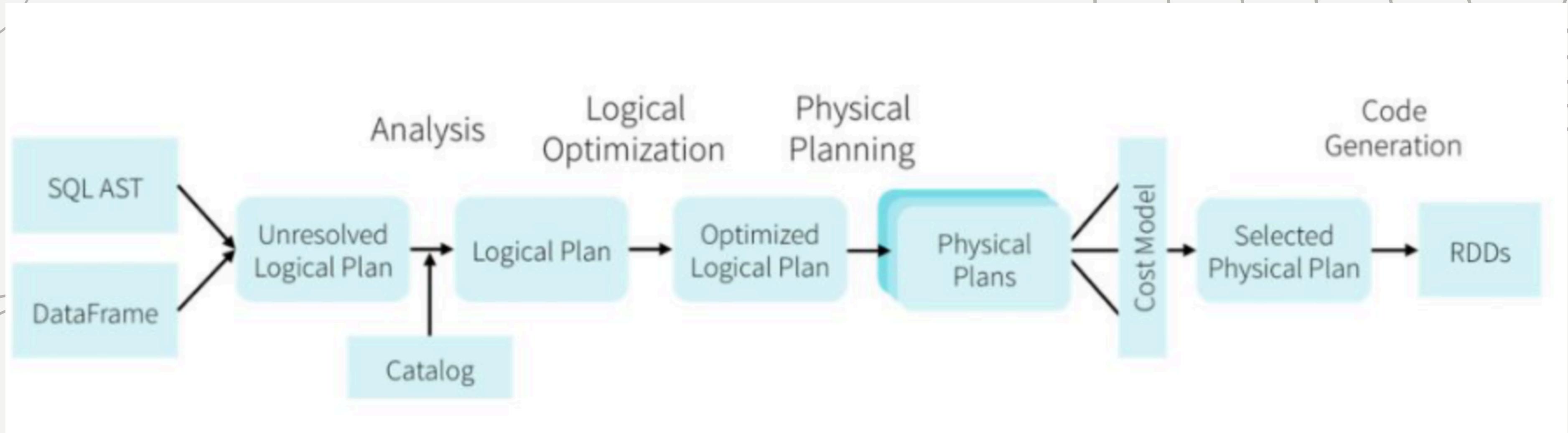
Executor - несут ответственность за фактическое выполнение работы:

- **выполнение кода**, назначенного ему Driver
- и сообщение о состоянии вычислений на этом исполнителе обратно Driver
- запускается в собственном **JVM** процессе

Spark



Apache Spark



Практика

<https://github.com/ReDwile/Spark-Practice-HSE>

Google Collab

Read

- 1) `spark.read.table()`
- 2) `spark.read.parquet()`
- 3) `spark.read.csv()`

Show

- 1) df.show()
- 2) df.describe().show()
- 3) df.printSchema()

Sql

`spark.sql() -> DataFrame`

Select ...

- 1) df.select("id", "name")
- 2) df.filter(" name = 'Kirill' ")
- 3) df.groupBy("name").agg(...)

withColumn

- 1) df.withColumn()
- 2) df.withColumnRename()

Пример реального проекта

Литература



НИУ ВШЭ

Спасибо за
внимание!

Вопросы

Отметь активность



Отметь посещение

