

Среда Spark

Spark SQL / DataFrame API

Лекция 3

Кирилл Сысоев

Обо мне

5+ лет в Big Data

HSE University

Senior Data Engineer

OneFactor/UZUM Data

Hadoop, Spark, ClickHouse, Kafka, Docker

Python/Scala, SQL



t.me/KRSysoev
ksysoev@hse.ru

Содержание курса

1. Введение в Big Data: как работают и где находятся большие данные;
2. Среда Spark. Spark RDD / Spark SQL;
3. **Advanced SQL;**
4. Spark ML / Spark TimeSeries;
5. Advanced ML и проверка результатов качества моделей;
6. Spark GraphX /Spark Streaming;
7. Экосистема Spark (MLFlow, AirFlow,H2O AutoML);
8. Spark в архитектуре проекта / Spark CI/CD.

Взаимодействие

Общение:

Мой telegram – личные вопросы/консультации/рекомендации

Лекции + ДЗ:

Telegram-чат «DS-15 Промышленное машинное обучение Spark» – после лекций буду туда публиковать материалы лекций и описание ДЗ с дедлайном

Сдача ДЗ:

Почта – в установленный дедлайн буду ждать письмо с вложением

Кратко про прошлую лекцию

Apache Spark: Введение

1. **Что такое Apache Spark?**
2. **SparkSession, SparkConf, SparkContext**
3. **Driver & Executors**
4. **Transformations & Actions**
5. **RDD**

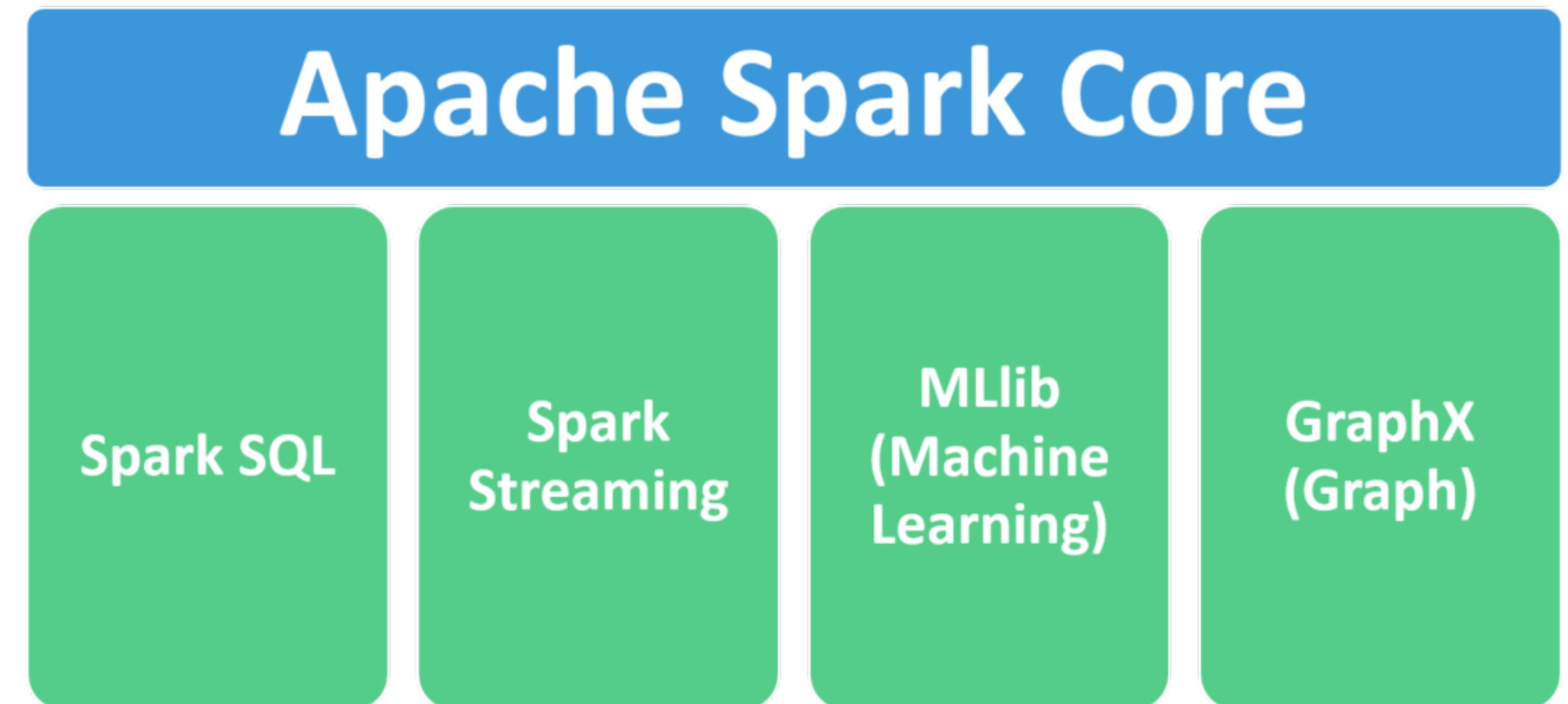
Кратко про прошлую лекцию

Apache Spark

Apache Spark — это распределённый движок обработки данных, созданный для быстрой работы с большими объёмами данных.

Преимущества:

- Скорость:** Работает в памяти, в 10–100 раз быстрее MapReduce.
- Универсальность:** Поддерживает batch, real-time (Streaming), SQL-запросы, ML и графовую обработку.
- Простота и гибкость:** Удобные API на Scala, Python, Java и SQL.
- Масштабируемость:** Легко масштабируется на кластерах.
- Экосистема:** Широкий набор библиотек (Spark SQL, Spark Streaming, MLlib, GraphX).
- Поддержка различных источников данных:** Интеграция с Hadoop, базами данных, облачными хранилищами.



Кратко про прошлую лекцию

Что такое **SparkSession**, **SparkConf**, **SparkContext**?

SparkSession — интерфейс взаимодействия с Spark
(рекомендуемый способ работы).

```
val spark = SparkSession.builder()  
    .appName("App")  
    .getOrCreate()
```

SparkConf — настройки приложения.

```
val conf = new SparkConf()  
    .setAppName("App")  
    .setMaster("local[*]")
```

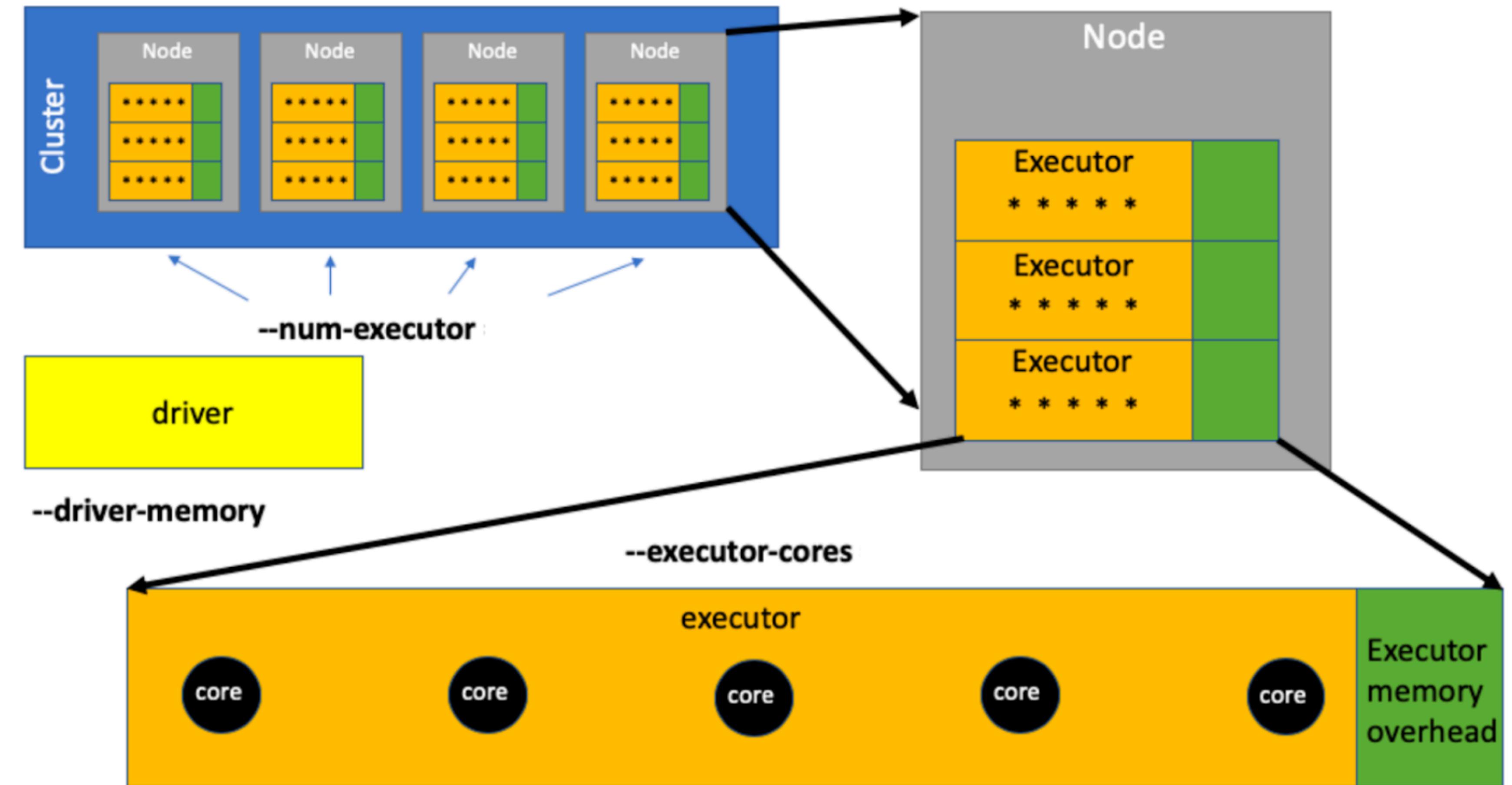
SparkContext — обеспечивает связь с кластером и управление задачами.

```
val sc = spark.sparkContext
```

Кратко про прошлую лекцию

Сущности Apache Spark

Driver & Executors



Кратко про прошлую лекцию

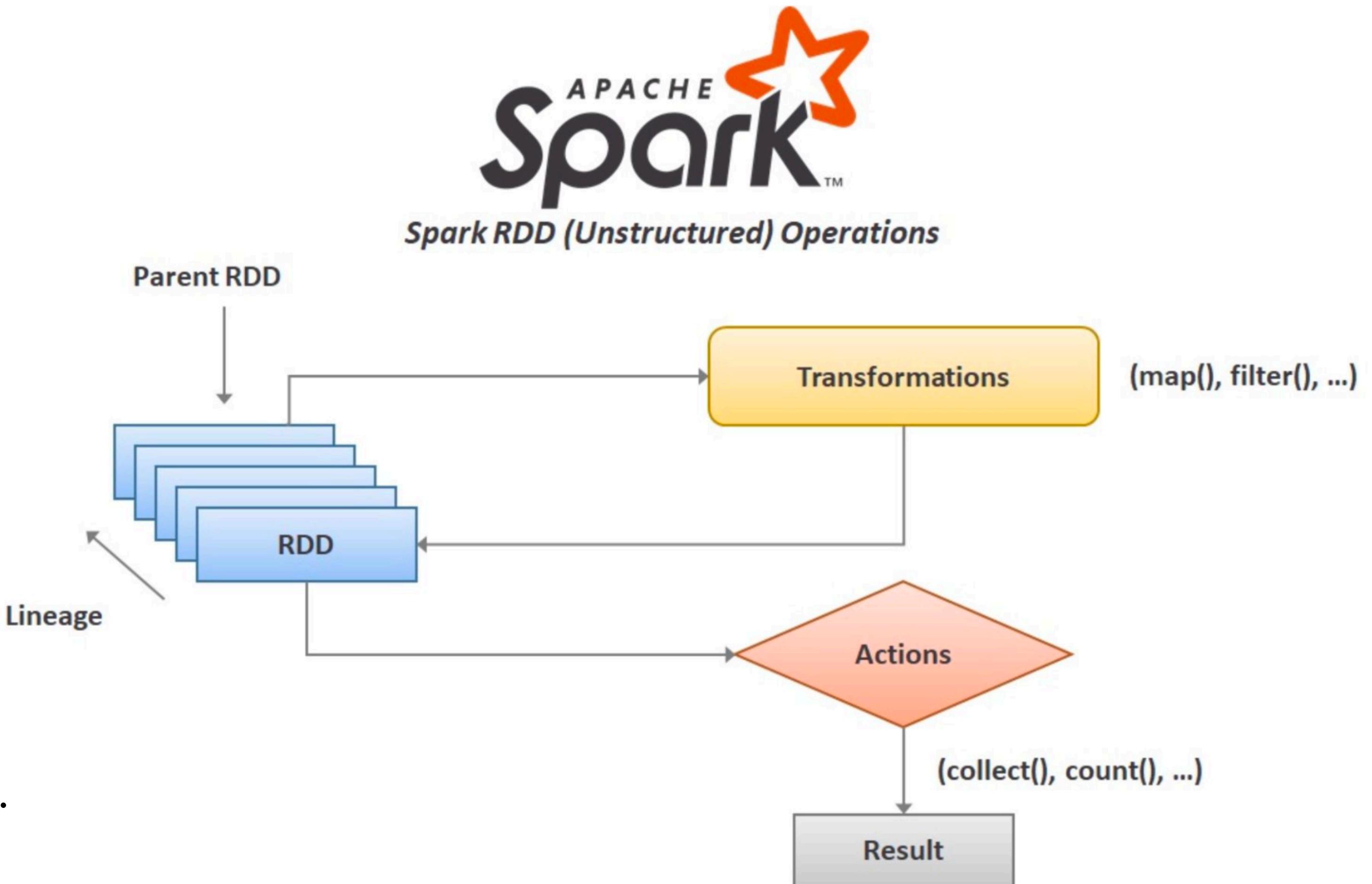
Виды операторов в Spark

Трансформации (transformations):

- Формируют новый RDD/DataFrame.
- Выполняются лениво (lazy evaluation), не запускаются сразу.
- Примеры: *map()*, *filter()*, *flatMap()*, *groupBy()*, *join()*.

Действия (actions):

- Запускают вычисления и возвращают результат драйверу.
- Немедленно запускают цепочку трансформаций.
- Примеры: *collect()*, *count()*, *take()*, *reduce()*, *foreach()*.



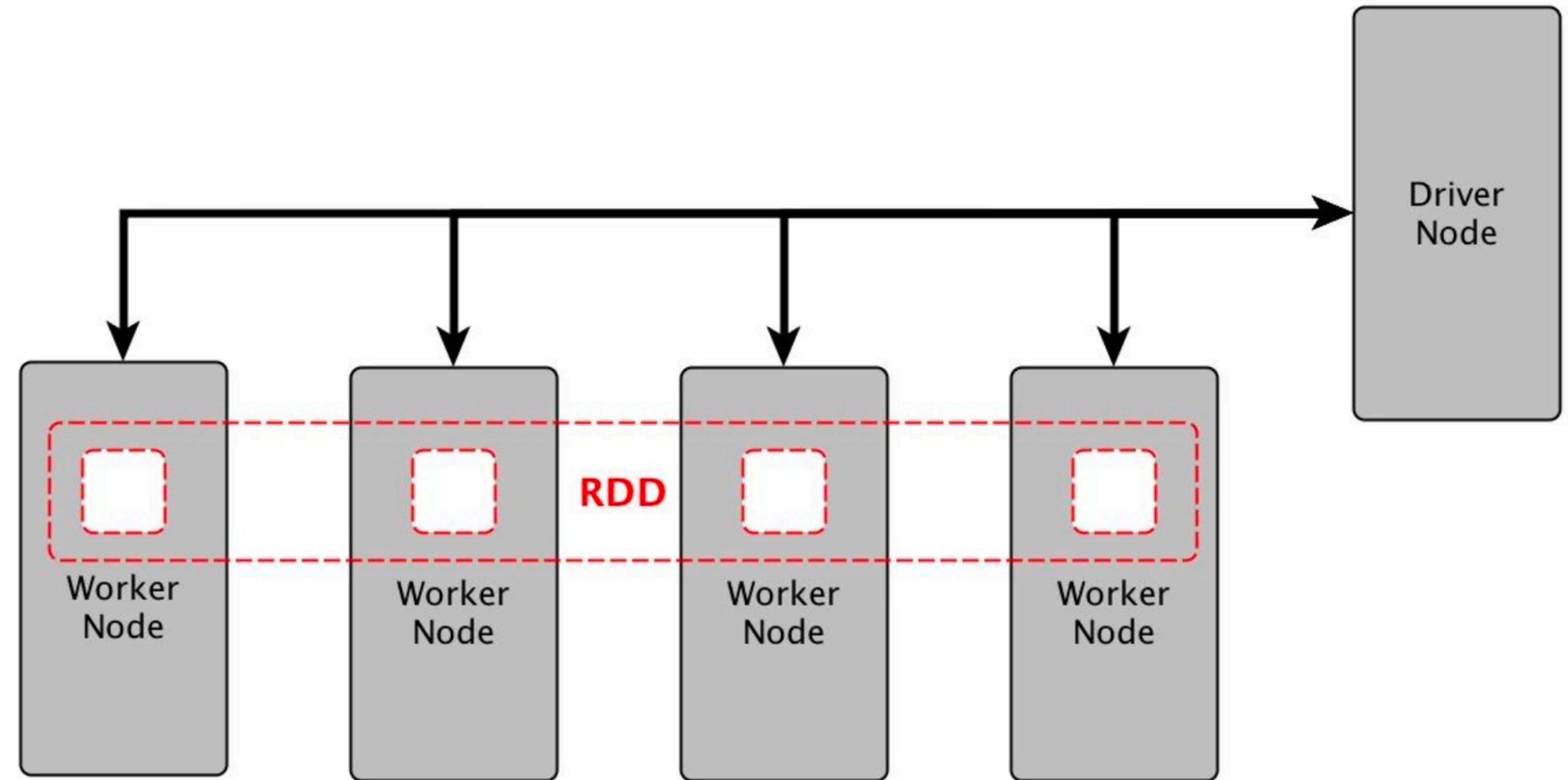
Кратко про прошлую лекцию

RDD

RDD (Resilient Distributed Dataset) — это базовая абстракция данных в Spark, представляющая собой отказоустойчивую распределённую коллекцию элементов, размещённую на кластере.

Свойства RDD:

- Распределённость:** Данные хранятся и обрабатываются параллельно на нескольких узлах.
- Отказоустойчивость:** В случае отказа узла данные восстанавливаются автоматически.
- Неизменяемость (Immutable):** Любое преобразование RDD создаёт новый RDD.
- Отложенные вычисления (Lazy Evaluation):** RDD вычисляются только при выполнении действий (actions).

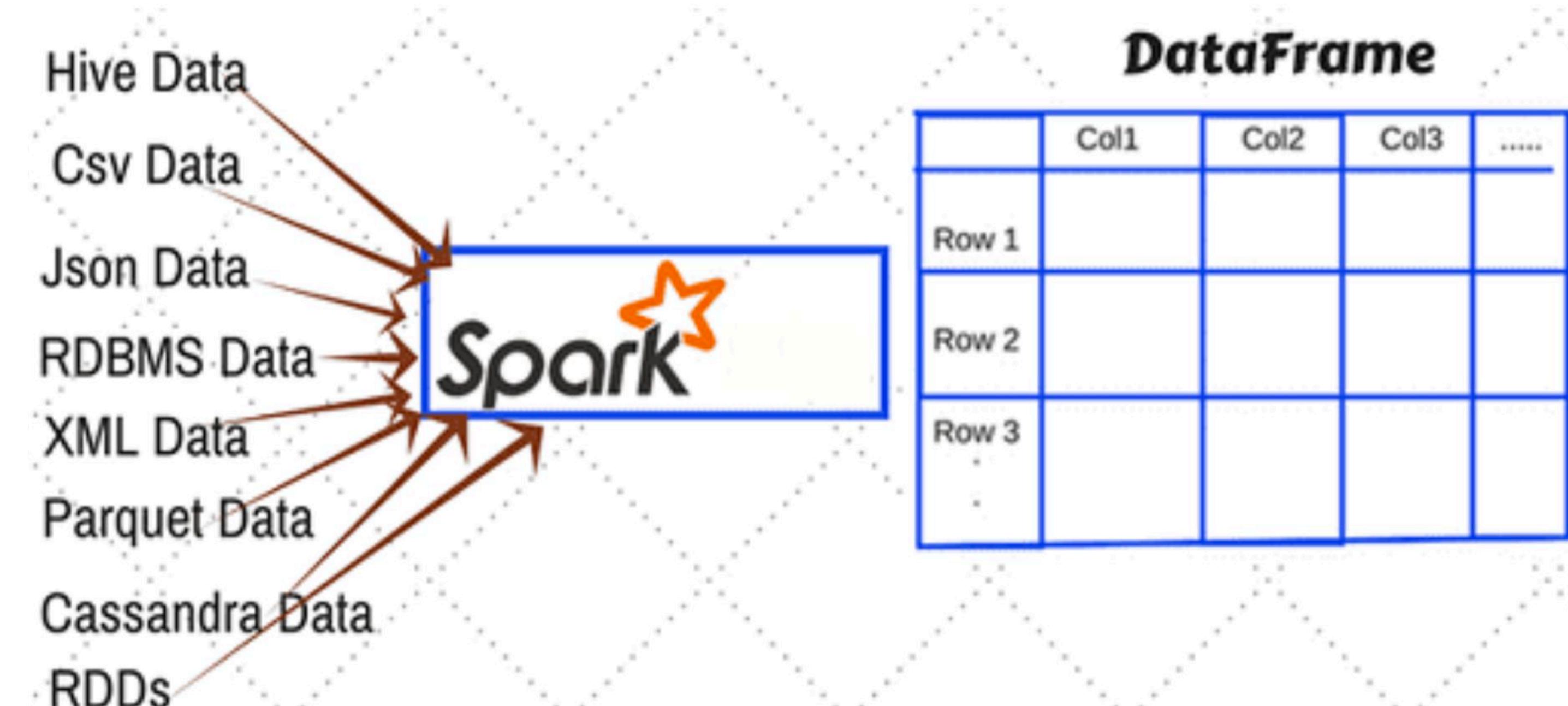


Spark DataFrame

Spark DataFrame — это распределённая коллекция данных, организованная по столбцам, похожая на таблицу в реляционной базе данных.

Свойства:

1. Имеет именованные столбцы.
2. Данные структурированы и распределены по узлам кластера.
3. Обладает оптимизацией запросов (Catalyst Optimizer).



RDD vs DataFrame

RDD (Resilient Distributed Dataset)

- Низкоуровневая структура данных.
- Не имеет схемы (нужна ручная обработка).
- Операции менее оптимизированы.
- Подходит для нестандартных задач и низкоуровневой обработки данных.

DataFrame

- Высокоуровневая структура данных, основанная на RDD.
- Имеет четко определённую схему (столбцы, типы данных).
- Использует Catalyst Optimizer для оптимизации операций.
- Лучше подходит для структурированных данных и SQL-запросов.

RDD vs DataFrame

Задача: Имеются данные о сотрудниках. Нужно отобрать сотрудников старше 30 лет, сгруппировать по отделу и посчитать, сколько сотрудников в каждом отделе.

RDD

```
# Отбираем сотрудников старше 30 лет (фильтрация)
filtered_rdd = employees_rdd.filter(lambda x: x[2] > 30)

# Группируем по отделу и считаем количество сотрудников
department_counts_rdd = (filtered_rdd
    .map(lambda x: (x[3], 1))
    .reduceByKey(lambda a, b: a + b))

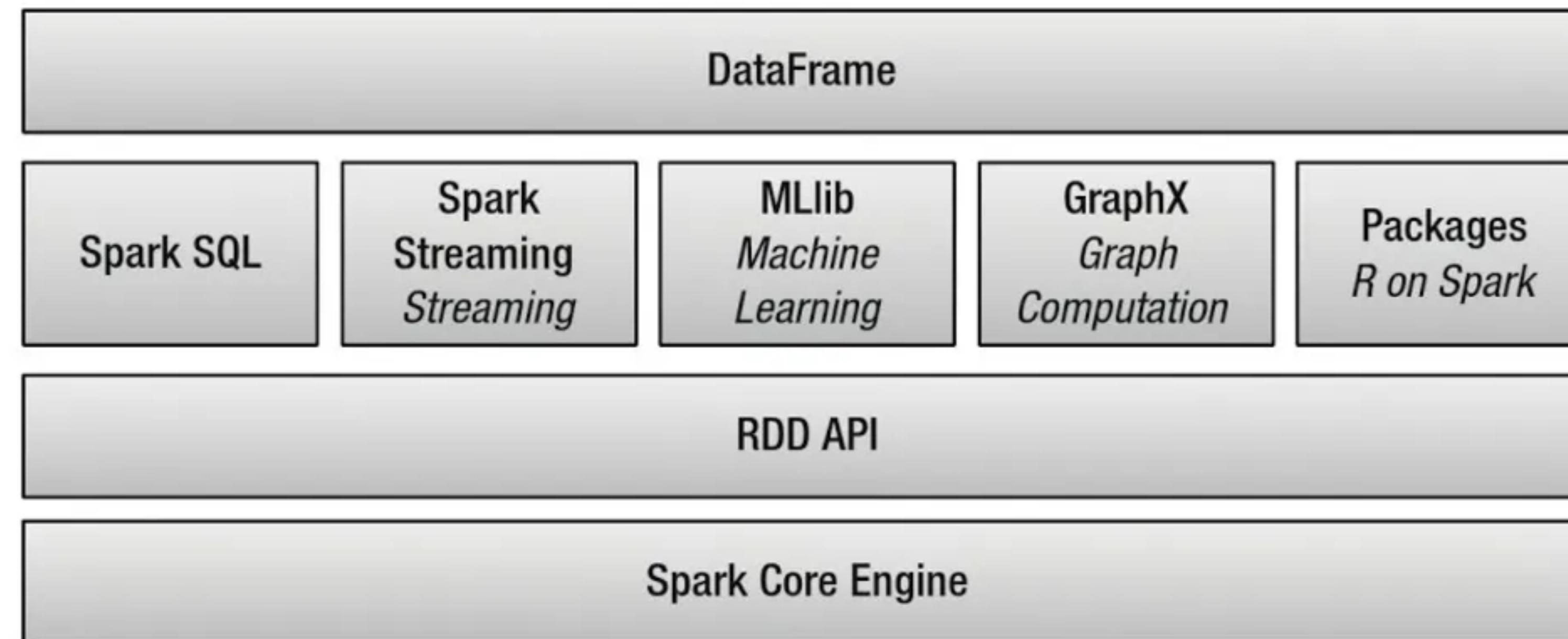
# Действие: собираем результат
result_rdd = department_counts.collect()
```

DataFrame

```
# Отбираем сотрудников старше 30 лет и группируем по отделу
result_df = (df.filter(col("age") > 30)
    .groupBy("department")
    .count())

# Действие: собираем результат
result_df.show()
```

Spark DataFrame



Spark SQL

Spark SQL – это модуль Spark для обработки структурированных данных с использованием **SQL-запросов** и **DataFrame API**. Он позволяет работать с данными аналогично обычным таблицам в реляционных БД.

Spark SQL предоставляет:

1. Выполнение SQL-запросов напрямую.
2. DataFrame API, оптимизированный через Catalyst Optimizer.
3. Поддержку форматов данных: Parquet, CSV, JSON и других.

DataFrame API (Трансформации)

| Метод | Описание | Основные параметры |
|----------------------------------|---|---|
| <code>select()</code> | Выбирает нужные столбцы из DataFrame | <code>cols</code> (<i>имена столбцов или выражения</i>) |
| <code>filter() / where()</code> | Отбирает строки по заданному условию | <i>условие фильтрации</i> |
| <code>withColumn()</code> | Создает новый столбец или изменяет существующий | <i>имя столбца, выражение</i> |
| <code>withColumnRenamed()</code> | Переименовывает столбец | <i>старое имя, новое имя</i> |
| <code>drop()</code> | Удаляет столбцы из DataFrame | <i>имена удаляемых столбцов</i> |
| <code>groupBy()</code> | Группирует данные по указанным столбцам | <i>имена столбцов</i> |
| <code>agg()</code> | Выполняет агрегации по заданным столбцам | <i>функции агрегации (avg, sum и др.)</i> |
| <code>join()</code> | Объединяет два DataFrame по ключу | <i>другой DataFrame, условие соединения</i> |
| <code>distinct()</code> | Убирает дубликаты записей | — |
| <code>orderBy() / sort()</code> | Сортирует данные по указанным столбцам | <i>имена столбцов, порядок сортировки</i> |
| <code>limit()</code> | Ограничивает количество возвращаемых строк | <i>количество строк (число)</i> |
| <code>drop()</code> | Удаляет указанный столбец из DataFrame | <i>имя столбца</i> |
| <code>join()</code> | Соединяет два DataFrame по указанному условию | <i>(DataFrame, условие, тип join)</i> |

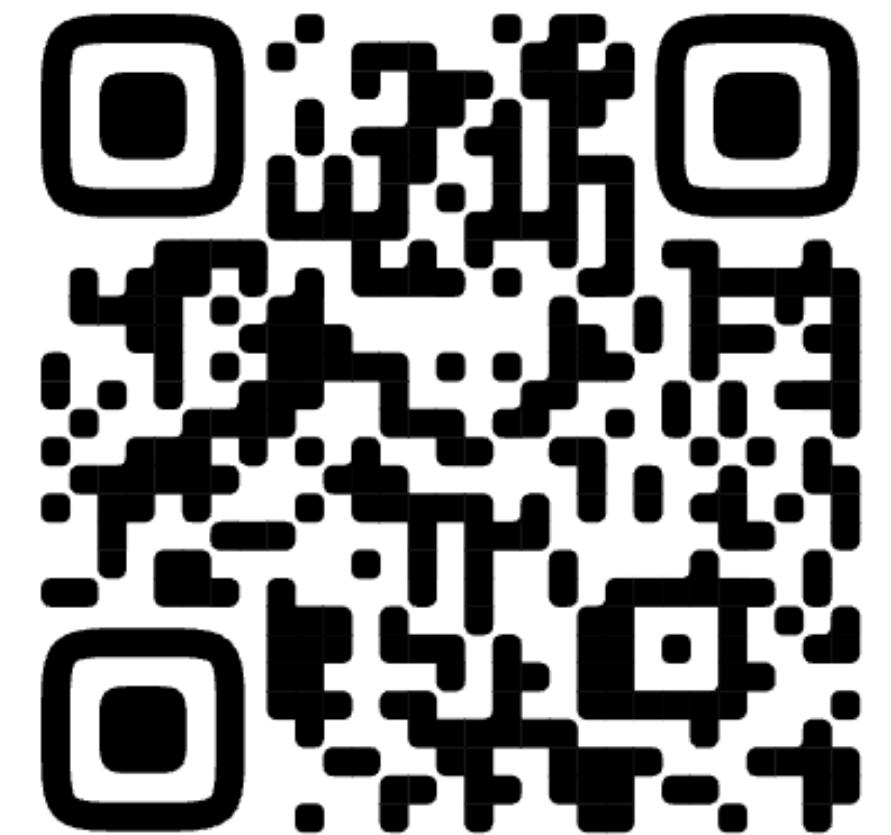
DataFrame API (Действия)

| Метод | Описание | Параметры |
|-------------------------|---|---|
| <code>show()</code> | Отображает первые строки DataFrame в виде таблицы | (число строк, <i>truncate=True/False</i>) |
| <code>collect()</code> | Возвращает все строки DataFrame в виде массива | — |
| <code>count()</code> | Возвращает количество строк DataFrame | — |
| <code>take()</code> | Возвращает указанное число первых строк | (число строк) |
| <code>head()</code> | Возвращает первую или первые несколько строк | (число строк, по умолчанию 1) |
| <code>first()</code> | Возвращает первую строку DataFrame | — |
| <code>write()</code> | Записывает DataFrame в файл или базу данных | (<i>format</i> , <i>mode</i> , <i>path</i> и другие опции) |
| <code>describe()</code> | Возвращает статистику (среднее, мин, макс и т.д.) | (имена столбцов, optionalno) |

Практика DataFrame

Репозиторий на GitHub с python-ноутбуком и файлом с данными
Рекомендуется его загрузить в Google Collab (вместе с данными)
и запускать от туда

<https://clck.ru/3HGT8Q>



Литература для доп погружения в тему

O'REILLY®

ВЫСОКО- НАГРУЖЕННЫЕ ПРИЛОЖЕНИЯ

Программирование
масштабирование
поддержка



ДИТЕР®

Мартин Клеппман

O'REILLY®

ЭВОЛЮЦИОННАЯ АРХИТЕКТУРА

ПОДДЕРЖКА НЕПРЕРВНЫХ ИЗМЕНЕНИЙ



Нил Форд, Ребекка Парсонс, Патрик Кью

ДИТЕР®

O'REILLY®

Data Governance The Definitive Guide

People, Processes, and Tools to Operationalize
Data Trustworthiness



Evren Eryurek, Uri Gilad,
Valliappa Lakshmanan,
Anita Kibunguchy-Grant
& Jessi Ashdown

Спасибо за внимание!