


Устройство и API Spark

Андрей Кузнецов
19.11.2022

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. Устройство и API Spark 
7. Approximate алгоритмы для больших данных
8. Поточковая обработка данных (Kafka, Spark Streaming, Flink)
9. Гостевая лекция VK
10. Гостевая лекция VK

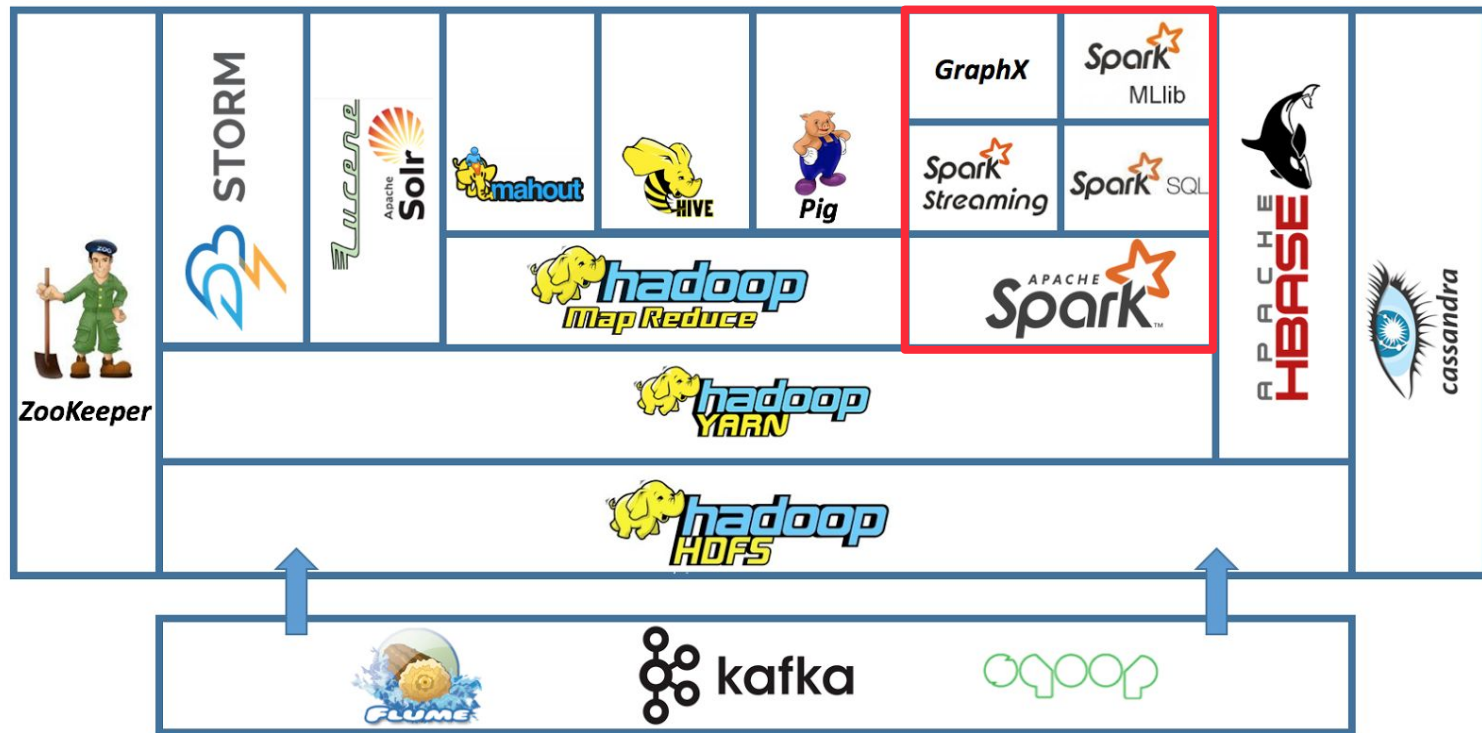
План лекции

1. Введение в Spark
2. RDD API
3. Dataset API
4. Workshop



Введение в Spark

Hadoop ecosystem



Apache Spark



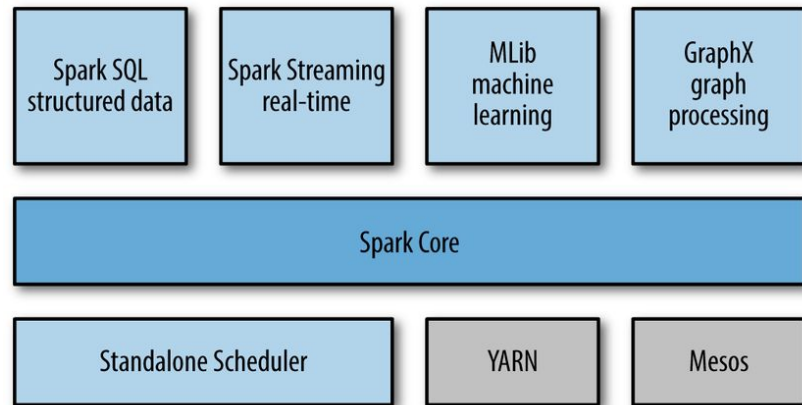
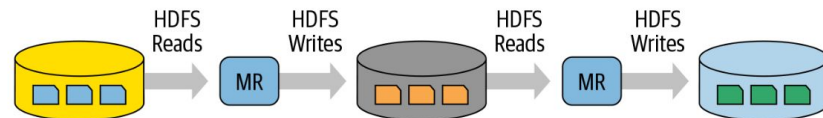
Spark - самый популярный фреймворк распределенных вычислений.

Написан на Scala, Java

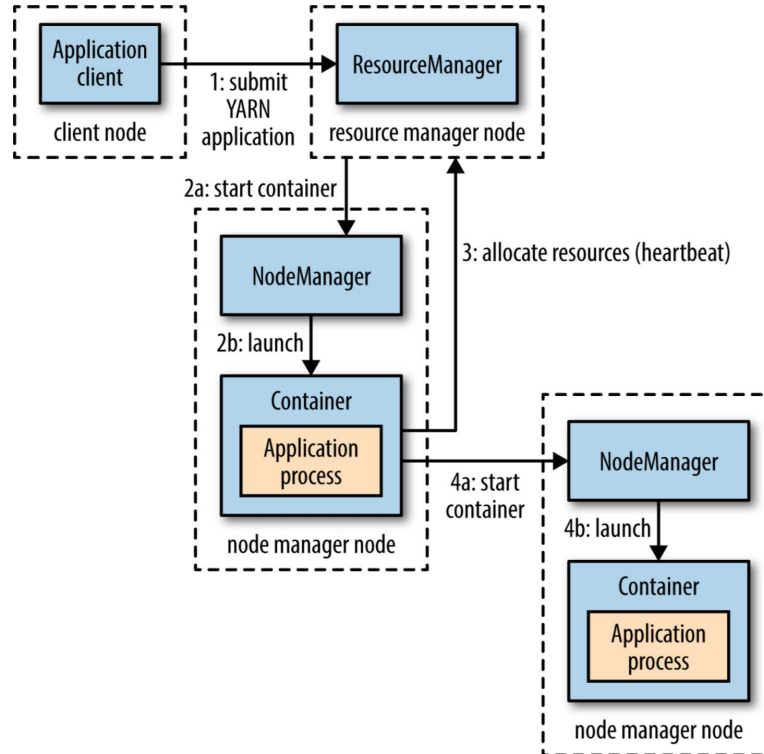
Поддерживает API на Scala, Java, SQL, Python, R, C#, F#

Apache Spark. Pros

- Классический MapReduce работает медленно
- Мапперы пишут данные на диск редьюсера
- Нет автоматической оптимизации запросов
- Spark работает значительно быстрее
- Хранение промежуточных результатов в памяти
- Оптимизирует запросы
- Не привязан к HDFS
- Не привязан к менеджерам ресурсов



YARN. Application workflow



Apache Spark. Architecture

Application

User program built on Spark. Consists of a driver program and executors on the cluster.

Driver program

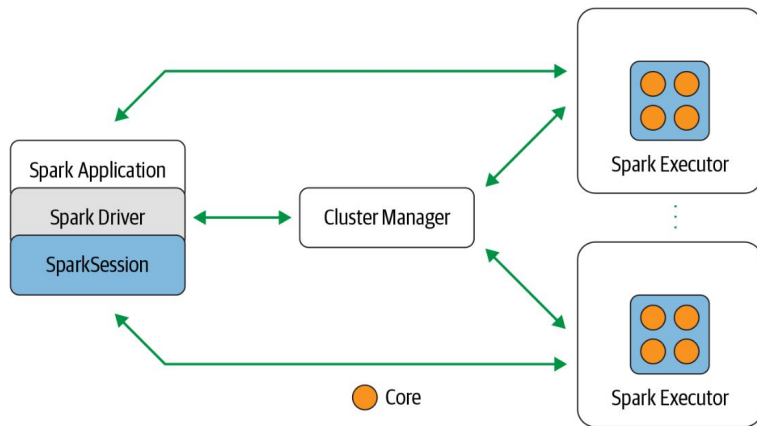
The process running the main() function of the application and creating the SparkContext

Deploy mode

"Cluster" driver inside of the cluster. "Client" outside of the cluster.

Executor

A process launched for an application on a worker node.



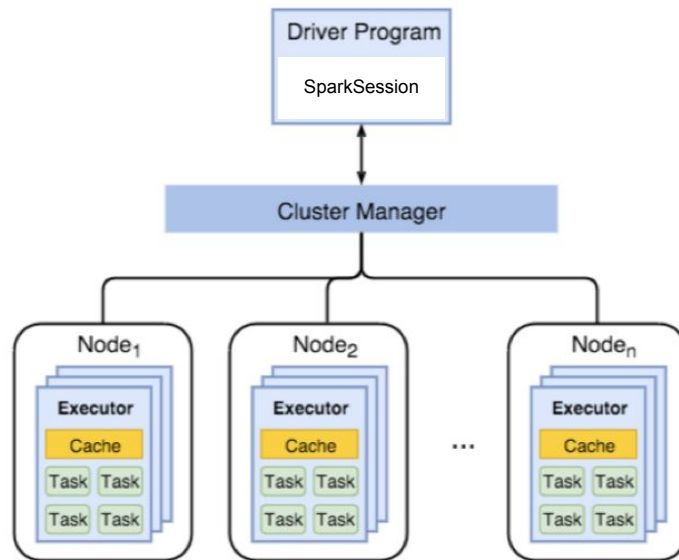
Spark. App workflow

Driver:

1. Инициализирует приложение
2. Запрашивает ресурсы
3. Формирует физический план
4. Передает сериализованный код задач
5. Отслеживает их выполнение
6. Завершает приложение
7. Запрашивает освобождение ресурсов

Executors

1. Получает от драйвера конкретную задачу
2. Отправляет статус выполнения на драйвер
3. Завершает выполнение по команде драйвера

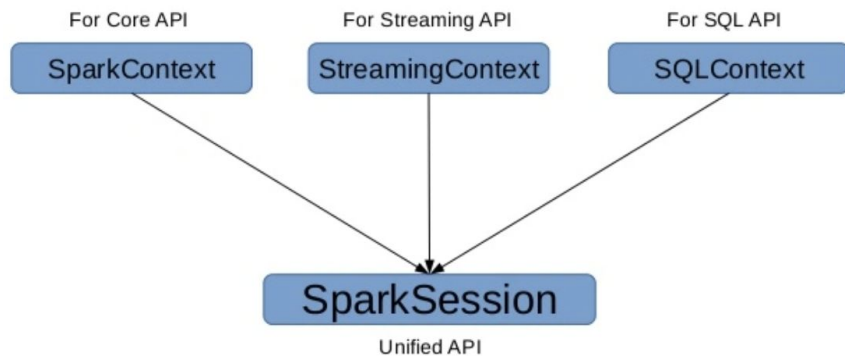


Apache Spark. SparkSession

```
import org.apache.spark.sql.SparkSession

val spark = SparkSession
  .builder
  .appName("BlablaSession")
  .getOrCreate()

val people =
  spark.read.parquet("/some/path")
```



Apache Spark. App runtime

Task

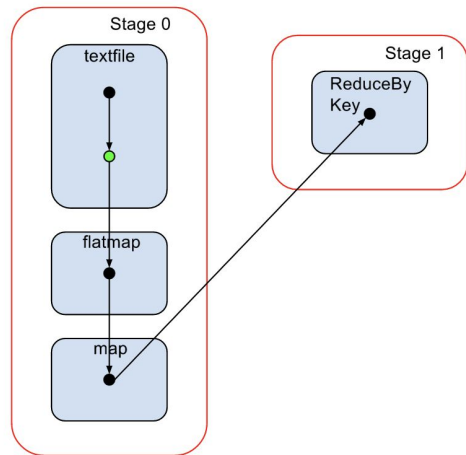
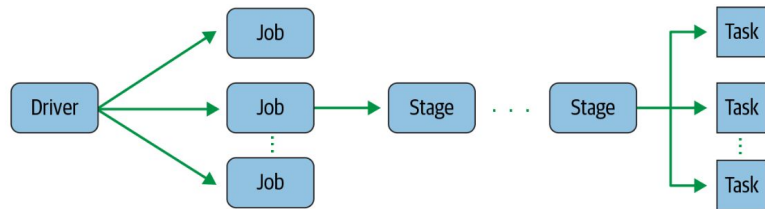
A unit of work that will be sent to one executor

Job

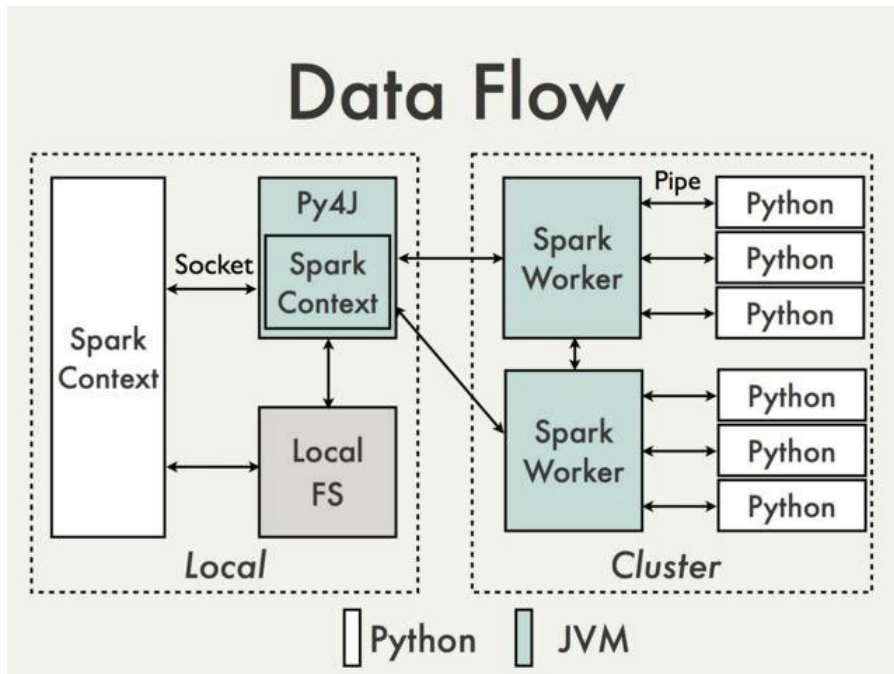
A parallel computation consisting of multiple tasks that gets spawned in response to a Spark action (e.g. save, collect)

Stage

Each job gets divided into smaller sets of tasks called stages that depend on each other (similar to the map and reduce stages in MapReduce)



Apache Spark. PySpark





RDD API

Spark. RDD

RDD (Resilient Distributed Dataset) – это простая, неизменяемая, распределенная коллекция объектов

Включает в себя три характеристики:

- Dependencies
- Partitions (with some locality information)
- Compute function: Partition => Iterator[T]

Может создавать распределенные коллекции из локальной ФС, HDFS, Cassandra, HBase, Amazon S3, etc.

Spark поддержка текстовых файлов, SequenceFiles, и других Hadoop InputFormat.

Spark. RDD

1. Распределенная коллекция (данные хранятся в партициях)
2. Неизменяемый (Immutable)
3. Хранит информацию о родителях и цепочке вычислений
4. Каждый элемент – сериализуемый объект
5. Ленивые трансформации (transformations) над RDD
6. Действия (actions) над RDD – запускают граф вычислений (Job)

Transformations	Actions
<code>orderBy()</code>	<code>show()</code>
<code>groupBy()</code>	<code>take()</code>
<code>filter()</code>	<code>count()</code>
<code>select()</code>	<code>collect()</code>
<code>join()</code>	<code>save()</code>

Spark. Типы трансформаций

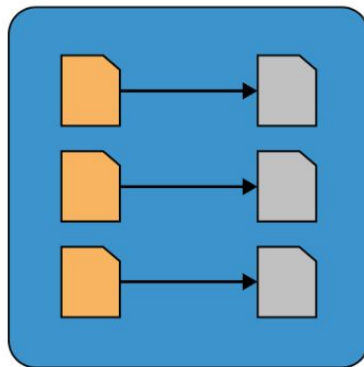
Узкая (Narrow Dependency)

Каждая партиция родительского RDD поступает на вход ровно одной дочерней

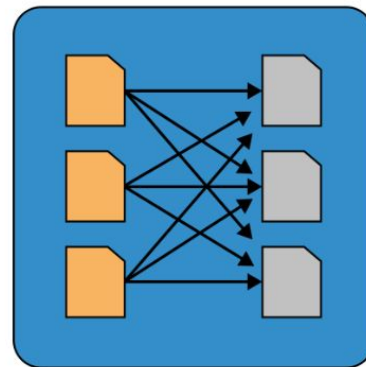
Широкая (Wide Dependency)

Партиция родительского RDD может поступать сразу в несколько дочерних партиций (нужен shuffle)

Narrow Dependencies



Wide Dependencies

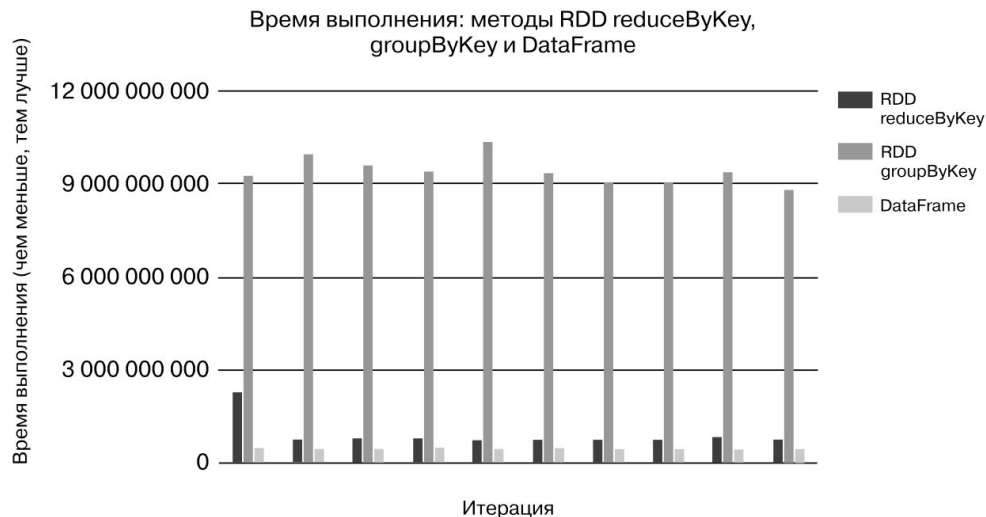




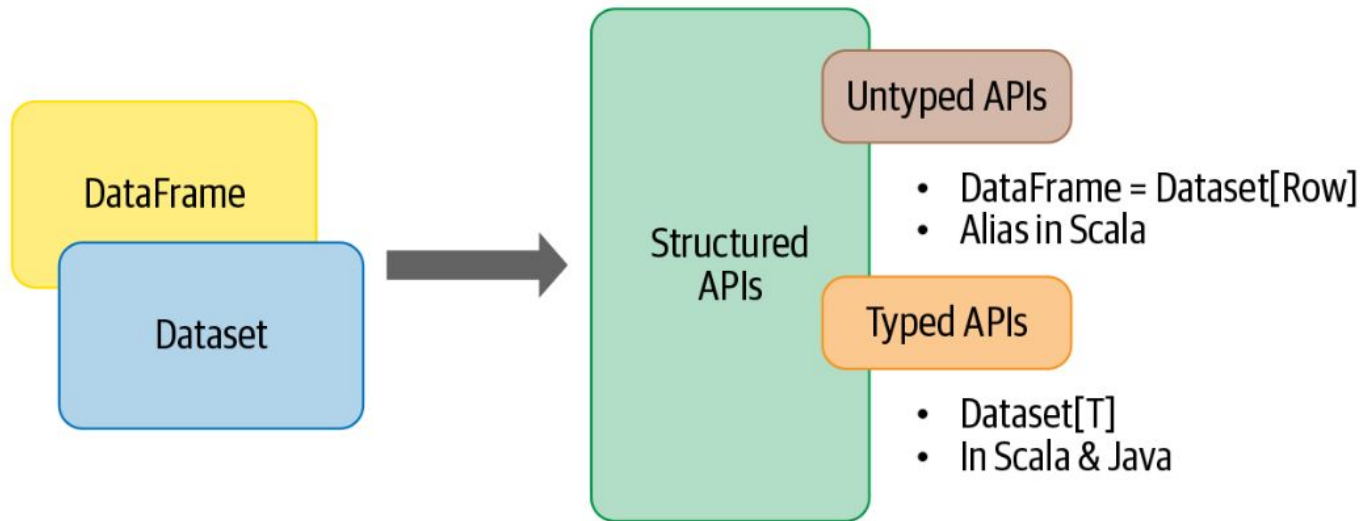
Dataset API

Spark. Structured APIs

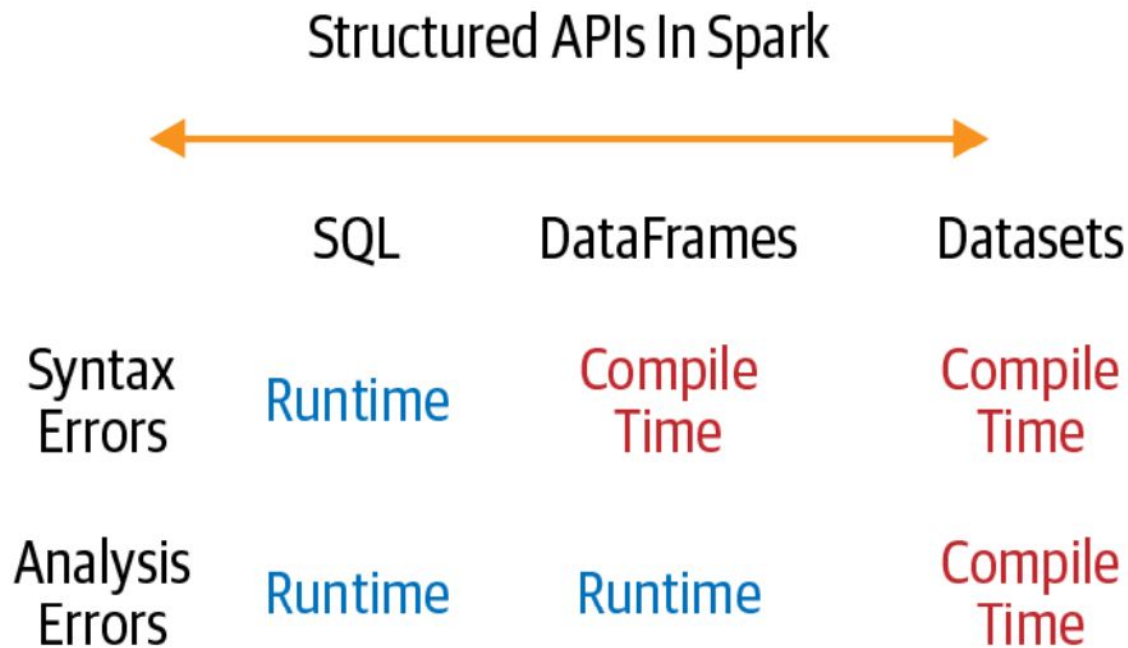
Подобно наборам RDD, наборы DataFrame и Dataset представляют собой распределенные коллекции с отсутствующей в RDD дополнительной информацией о схеме.



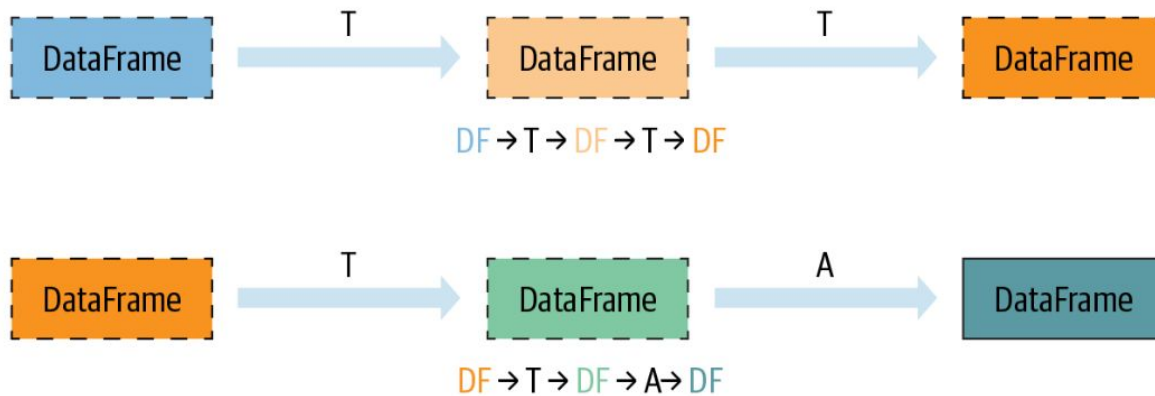
Spark. Structured APIs



Spark. Structured APIs



Spark. Lazy operations in DataFrames



DataFrame. Types

Строка в Spark is это объект Row, который состоит из одной или более колонок

Data type	Value assigned in Python	API to instantiate
ByteType	int	DataTypes.ByteType
ShortType	int	DataTypes.ShortType
IntegerType	int	DataTypes.IntegerType
LongType	int	DataTypes.LongType
FloatType	float	DataTypes.FloatType
DoubleType	float	DataTypes.DoubleType
StringType	str	DataTypes.StringType
BooleanType	bool	DataTypes.BooleanType
DecimalType	decimal.Decimal	DecimalType

Data type	Value assigned in Scala	API to instantiate
BinaryType	Array[Byte]	DataTypes.BinaryType
TimestampType	java.sql.Timestamp	DataTypes.TimestampType
DateType	java.sql.Date	DataTypes.DateType
ArrayType	scala.collection.Seq	DataTypes.createArrayType(ElementType)
MapType	scala.collection.Map	DataTypes.createMapType(keyType, valueType)
StructType	org.apache.spark.sql.Row	StructType(ArrayType[fieldTypes])
StructField	A value type corresponding to the type of this field	StructField(name, dataType, [nullable])

Dataset API

1. Работает с таблицами (DataFrame)
2. Таблицы имеют схему (колонки и их типы)
3. Ограничен SQL операторами
4. SQL операции генерируют Scala код
5. Но можно делать user-defined functions
6. Единое API для всех типов данных
7. Знание о типах позволяют делать оптимизацию

Построение физического плана

Catalyst преобразует синтаксическое дерево в физический план выполнения

Catalyst состоит из четырех этапов:

1. Анализ – проверяет метаданные таблиц (колонки, типы)
2. Логическая оптимизация
 - a. constant folding (считаем константы заранее и заменяем везде)
 - b. predicate pushdown (берем только нужные куски данных)ё
 - c. другие
3. Физическое планирование – cost-based оптимизация
4. Генерация кода



Workshop

Lecture summary

1. **Spark** - текущая SOTA в распределенных вычислениях
2. **Driver** - управляет процессом, может размещаться в аппмастере,
Executor - выполняет работу.
3. Приложение разбивается на **джобы, стейджи и задачи**
4. **RDD** - для низкоуровневой работы, **DataFrame** для PySpark, **Dataset** - для Scala

Recommended literature

