



# Введение в Spark

Дмитрий Киселёв

24.10.2020

# План лекции

- Что такое Spark?
- RDD API
- Выполнение кода на Spark
- DataFrame API
- Когда применять?

# Обо мне



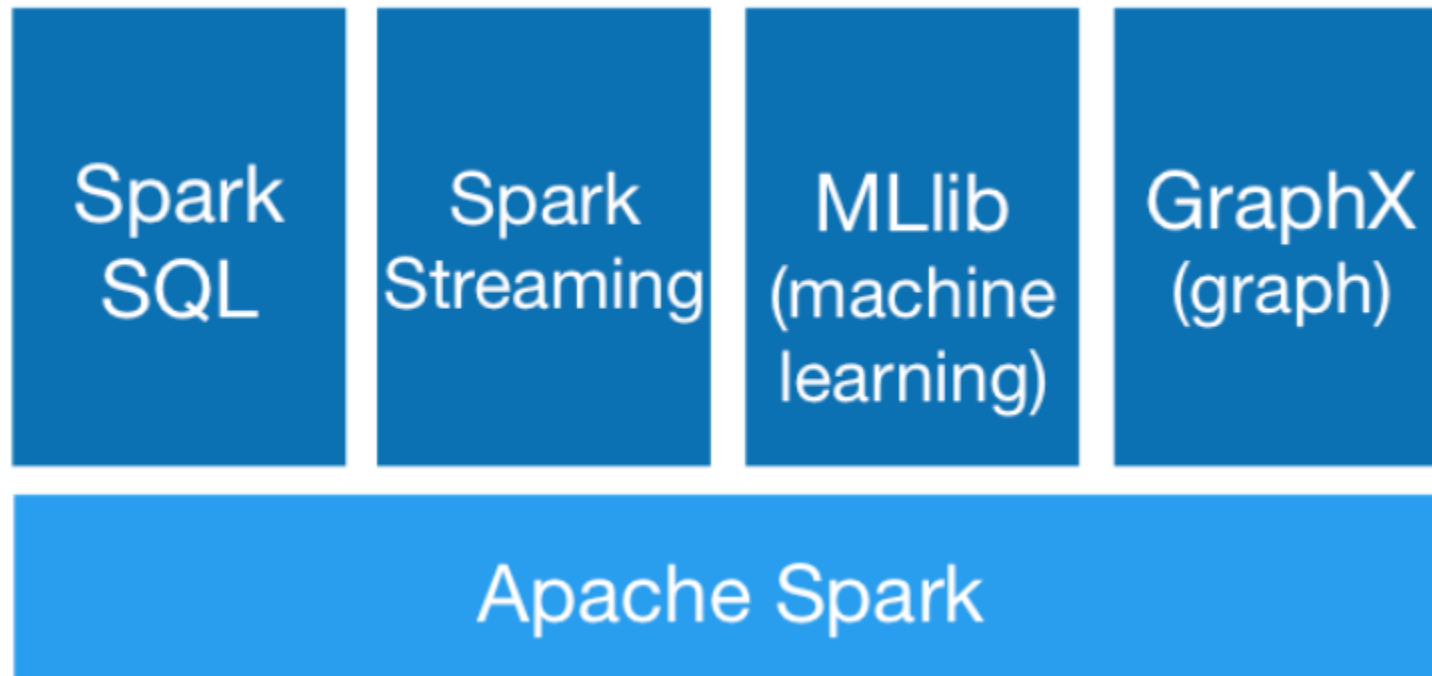
- Data Scientist в Платформе  
Рекомендательных систем Сбера
  - Занимаюсь контекстными многорукими бандитами
- Аспирант ФКН НИУ ВШЭ
  - Занимаюсь графовыми непрерывными представлениями и их применением в рекомендательных системах



- Контакты: [dkiseleff@gmail.com](mailto:dkiseleff@gmail.com), tg: dkiselev

# Apache spark

- Apache Spark is a unified analytics engine for large-scale data processing.



# Зачем?

- Классический MapReduce работает медленно
  - Мапперы пишут данные на диск редьюсера
  - Нет оптимизации запросов
- Spark работает значительно быстрее
  - Хранение промежуточных результатов в памяти
  - Восстанавливаемые распределенные наборы данных (RDD)
  - Представление заданий в виде направленных ациклических графов (DAG)
- Не привязан к HDFS
- Не привязан к менеджерам ресурсов

# Resilient Distributed Dataset

- Распределенная коллекция (данные хранятся в партициях)
- Неизменяемый (Immutable)
- Хранит информацию о родителях и цепочке вычислений
- Каждый элемент – сериализуемый объект
- Ленивые трансформации (transformations) над RDD
  - map, join, ...
- Действия (actions) над RDD – запускают граф вычислений (Job)
  - save, collect, ...

# Зависимость между RDD

- 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
- По факту различие между зависимостями – необходимость шафла

# Физический план

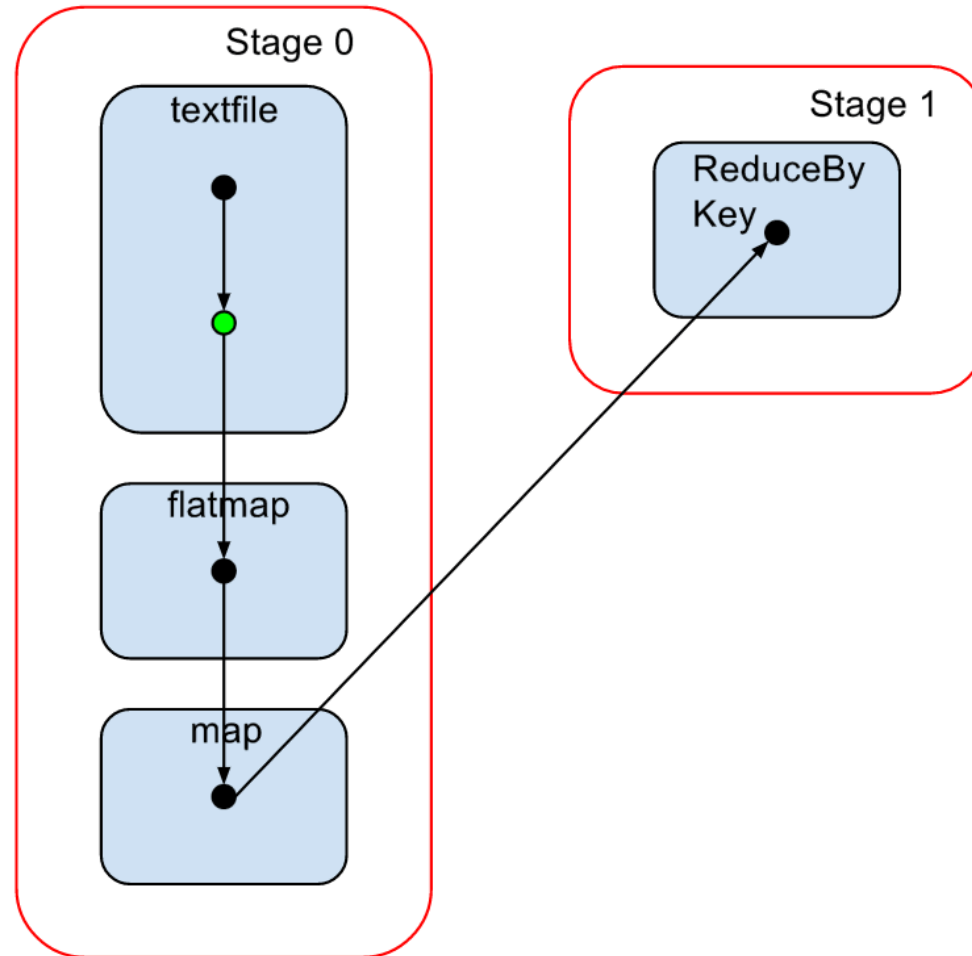
- Физический план представляет собой DAG
- Физический план описывает выполнение задания (Job)
- Физический план состоит из этапов (Stages)
- Этапы состоят из задач (Tasks) над конкретным RDD



# Построение физического плана

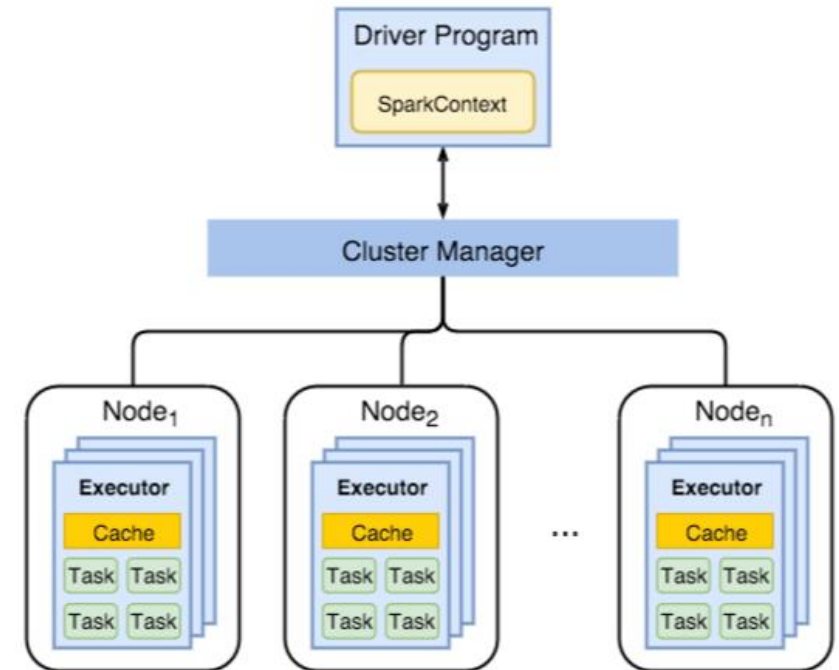
- Вызов операции действия (Action)
- Построение логического плана
  - Строится из графа зависимостей RDD
- Разделение логического плана на задачи (Tasks)
- Конвейеризация задач (Pipelining) в этапы (Stages)
  - Spark объединяет узко зависимые RDD в один этап

# Пример WordCount



# Исполнение кода на Spark

- Driver
  - Инициализирует приложение
  - Запрашивает ресурсы
  - Формирует физический план
  - Передает сериализованный код задач
  - Отслеживает их выполнение
  - Завершает приложение
  - Запрашивает освобождение ресурсов
- Executors
  - Получает от драйвера конкретную задачу
  - Отправляет статус выполнения на драйвер
  - Завершает выполнение по команде драйвера



# DataFrame API

- Работает с таблицами (DataFrame)
  - Таблицы имеют схему (колонки и их типы)
- Ограничен SQL операторами
  - Sql операции генерируют java код (ок использовать в PySpark)
  - Но можно делать user-defined functions
- Единое API для всех типов данных
- Знание о типах позволяют делать оптимизацию

# Оптимизация запросов (Catalyst)

- Catalyst преобразует синтаксическое дерево в физический план выполнения
- Catalyst состоит из четырех этапов
  - Анализ – проверяет метаданные таблиц (колонки, типы)
  - Логическая оптимизация
    - constant folding (считаем константы заранее и заменяем везде)
    - predicate pushdown (берем только нужные куски данных)
    - другие
  - Физическое планирование – cost-based оптимизация
  - Генерация кода

# Joins

- Shuffle join
  - Бьем все по ключу и кладем в одну партицию (Map + Shuffle)
  - Джойним по хэшам (Hash join)
  - Или слиянием (Merge join)
- Broadcast joint
  - Если таблица достаточно маленькая, то можем сделать по ней хэш таблицу и передать на все мапперы
- Spark делает для них CBO

# Когда применять Spark?

- Когда много данных
- Когда много источников
- Когда нужна гибкость

Workshop



# Домашнее задание

- По данным [TripAdvisor hotel reviews](#) посчитать [Tf-Idf](#) с помощью Spark DataFrame / Dataset API без использования Spark ML
- Этапы
  - Привести все к одному регистру
  - Удалить все спецсимволы
  - Посчитать частоту слова в предложении
  - Посчитать количество документов со словом
  - Взять только 100 самых встречаемых
  - Сдвоить две полученные таблички и посчитать Tf-Idf (только для слов из предыдущего пункта)
  - Запайвотить табличку