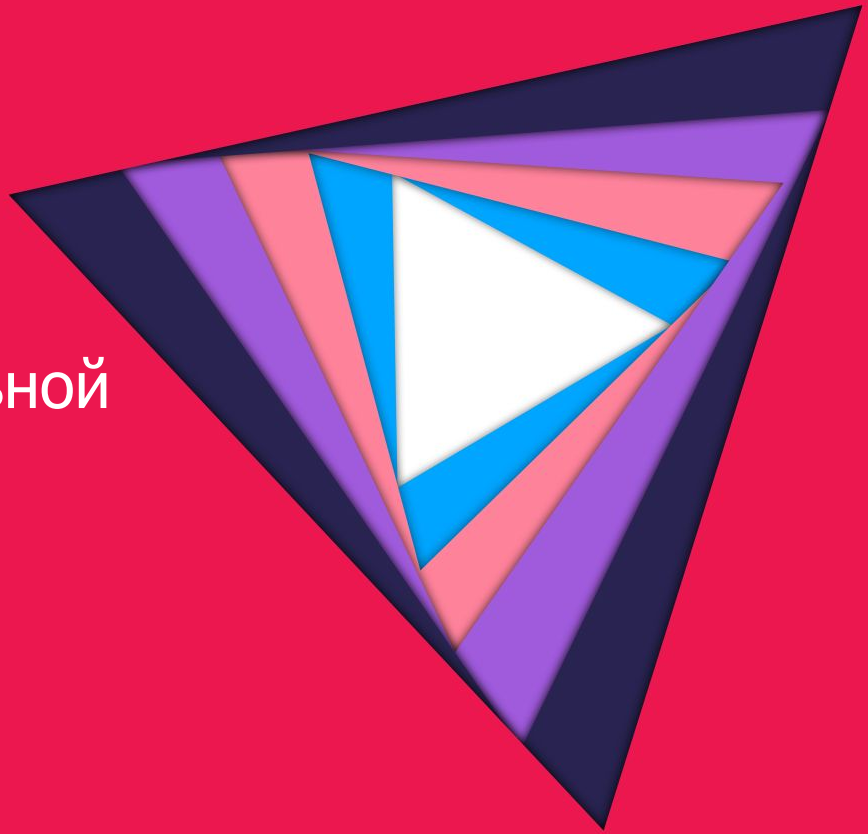
онлайн-кинотеатр

Распределённое ML на больших данных: опыт построения рекомендательной системы в ivi

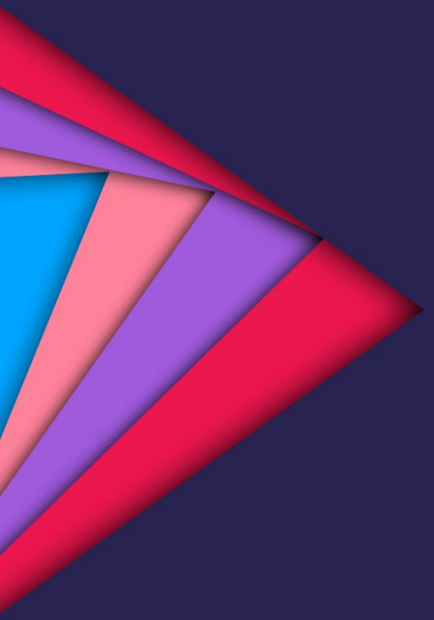


Борис Шминке
bshminke@ivi.ru

2017

О чём поговорим

- эволюция рекомендательной системы ivi
- ivi Data Lake
- Spark для Data Scientist: проблемы и их решения
- кейс: параллельный подбор метапараметров на Spark

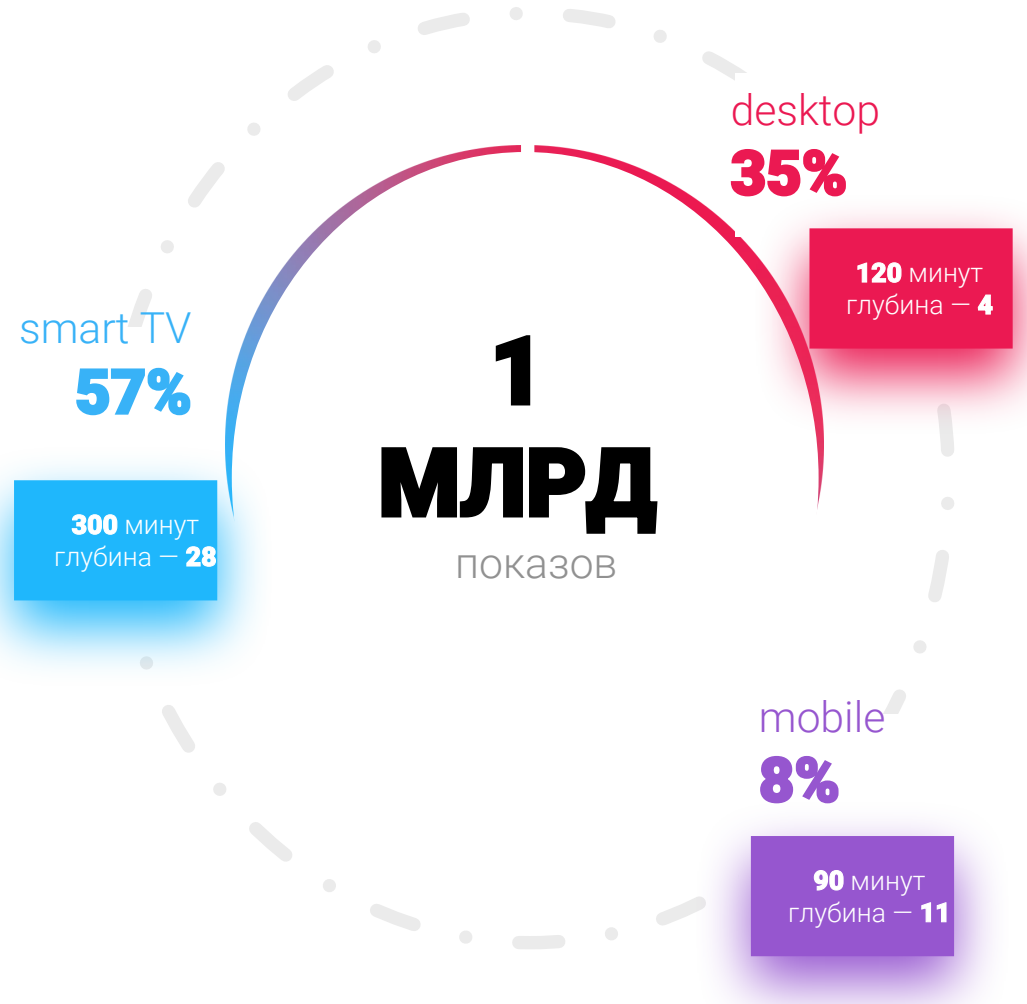


Коротко об ivi

СРЕДНЕМЕСЯЧНАЯ РЕКЛАМНАЯ АУДИТОРИЯ ivi

33 МЛН

пользователей



Источник: Пользователи – аналитика ivi data, люди – экстраполяция данных Mediascope. РФ, города 0+, возраст 12-64


Paramount
A VIACOM COMPANY
UNIVERSAL
SONY
PICTURES
20th
CENTURY
FOX
Disney

ivi ПРИБЛИЖАЕТ КИНО И СОКРАЩАЕТ ОКНА ПРОКАТА

ПО НЕЗАВИСИМЫМ РЕЛИЗАМ

БЫЛО

6–11 недель

СТАЛО

2–4 недели

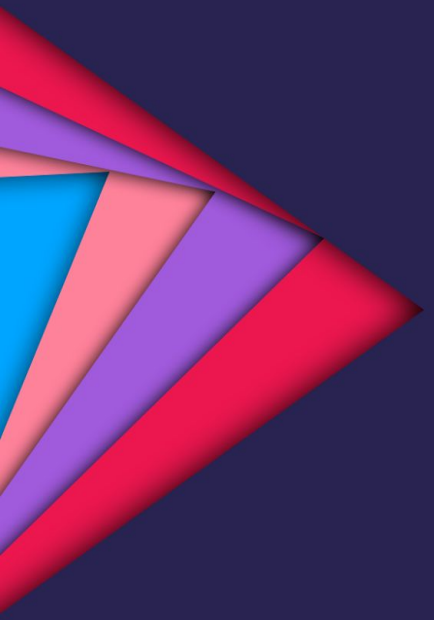
ПО СТУДИЙНЫМ РЕЛИЗАМ

БЫЛО

9–18 недель

СТАЛО

4–15 недель



Data Science в ivi

Data Science в ivi

- аналитическое хранилище данных (Tableau dashboards)
- маркетинговые модели (отток, потенциальные покупатели)
- финансовые модели (маржинальность закупок контента)
- рекомендательная система

Персональные рекомендации

Рекомендуем вам посмотреть >

Маша и Медведь

Фиксики

Доспехи бога: В поисках сокровищ

Молодежка

МиниФорс

Динамичные фильмы >

Механик: Воскрешение

Боги Египта

Елки 5

Молот

Экипаж

Американские комедии >

Рекомендации item-to-item

Меч короля Артура

2017, США, Фэнтези, Зарубежные, 121 мин., 16+

▶ Смотреть

ОПЛАЧЕНО

i Доступно неограниченное количество просмотров в любое время.

Детали платежа

Дата 12.10.2017 22:37
Номер транзакции [redacted]
Способ оплаты Личный счет на ivi
Сумма 399.00 руб.

Личные данные

Идентификатор в ivi [redacted]
Почта [redacted]
Телефон [redacted]

Вам также может понравиться

Сразу три модели монетизации

- ✓ рекламная: смотреть бесплатно и не весь каталог
- ✓ подписка: весь каталог без рекламы
- ✓ транзакционная: от аренды одного фильма до возможности скачать целый сезон сериала

От чёрного ящика к собственной рекомендательной системе

Recommendations as a Service

от Gravity (сейчас yusp.com)

плюсы

- как-то работает
- не нужны специалисты по ML

минусы

- чёрный ящик
- мало рычагов для кастомизации

Recommendations as a Service

от Gravity (сейчас yusp.com)

плюсы

- ~~как-то работает~~
- ~~не нужны специалисты по ML~~

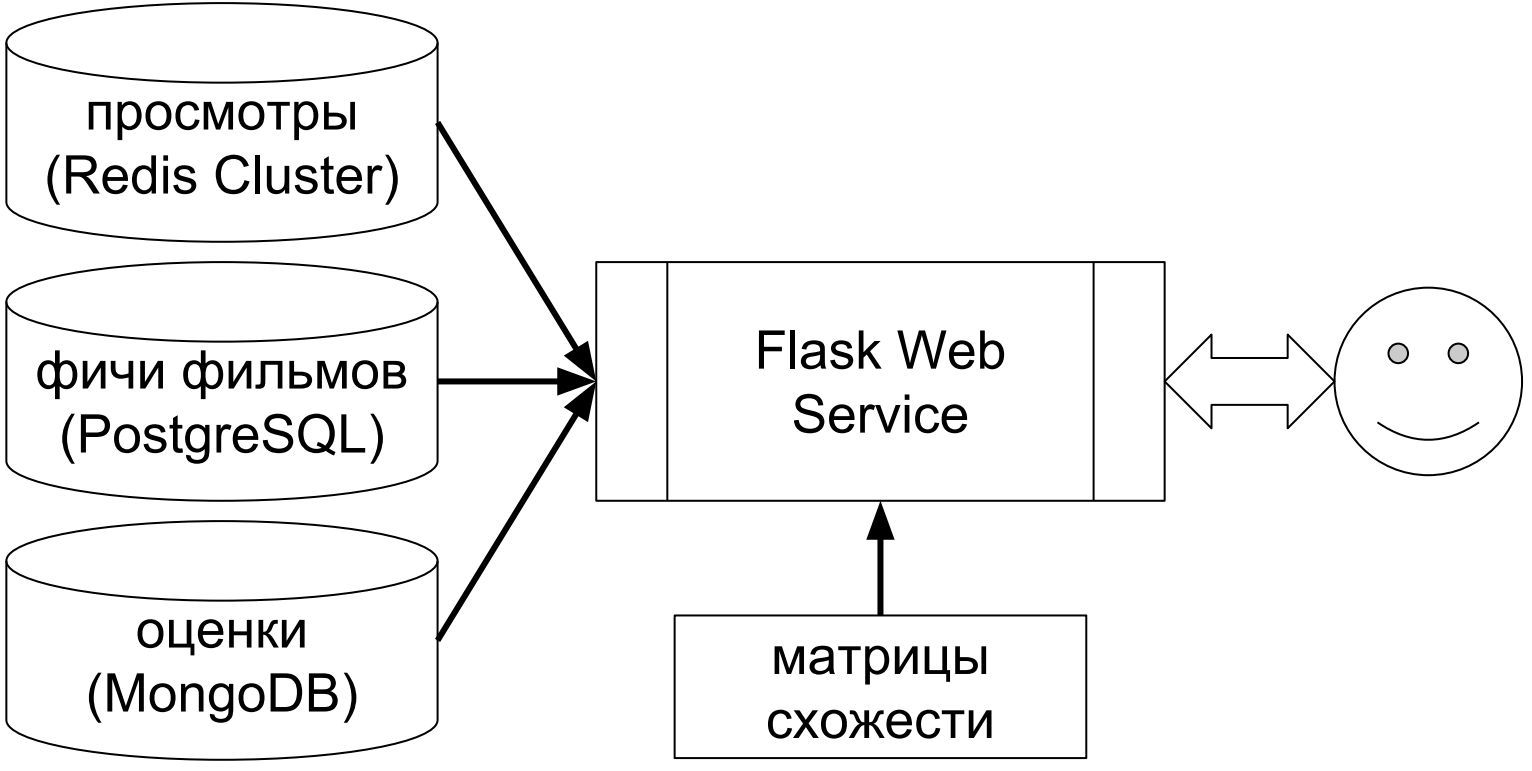
минусы

- чёрный ящик
- мало рычагов для кастомизации
- **невозможно развиваться дальше**

Своя рекомендательная система (2014)

- item-based collaborative filtering
- kNN для предсказания оценки
- учёт просмотров как небольших положительных оценок
- тот самый пост на habr'e

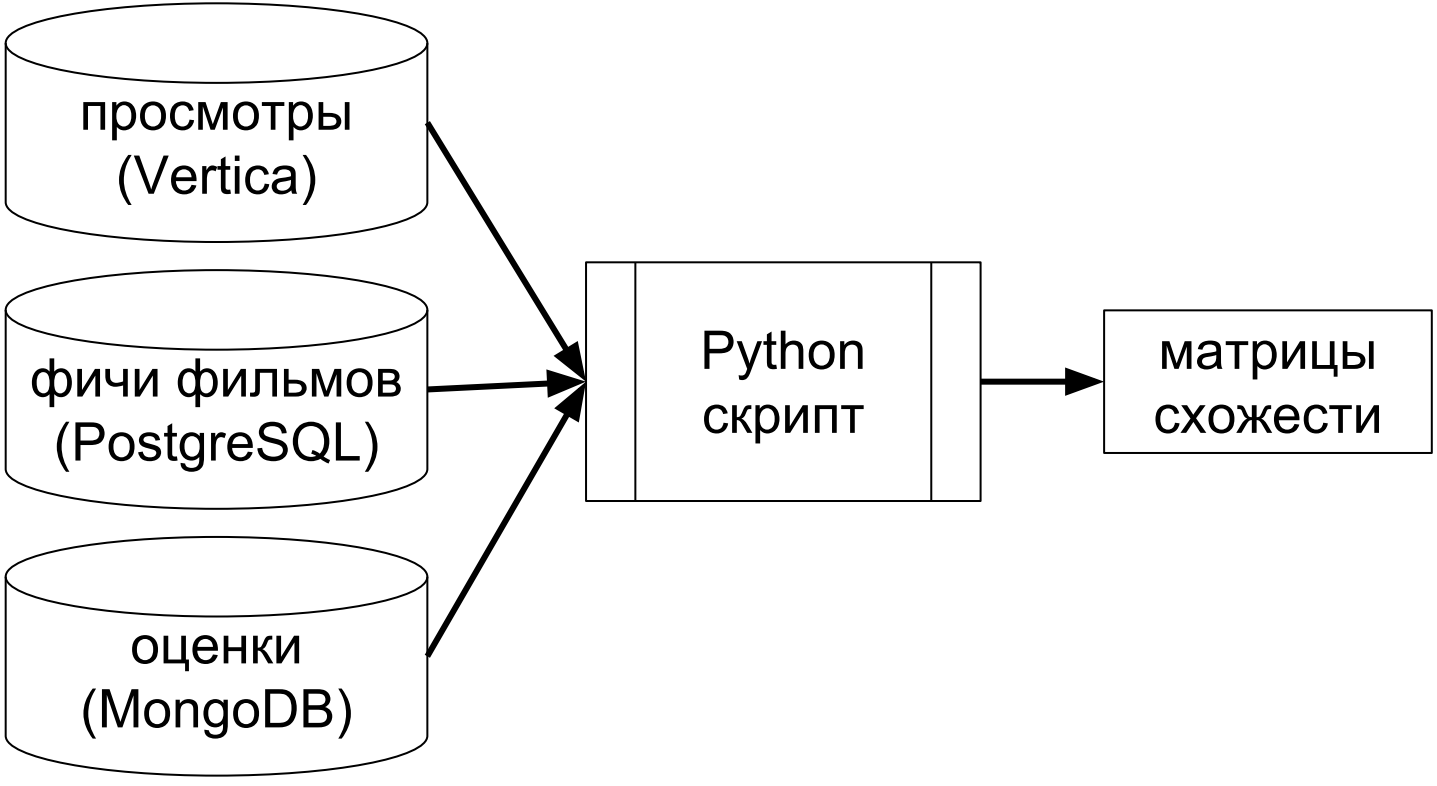
Выдача рекомендаций



Выдача рекомендаций

- легко добавлять бизнес-правила
- много потоков (uwsgi)
- несколько серверов (nginx)
- масштабируется до сотен RPS
- мгновенно реагирует на действия пользователя

Построение модели



Построение модели

- простой python-скрипт
- запускается раз в сутки
- делает крутое ML!
- работает всего полчаса

Данных становится больше

2014

- 1.4 млн пользователей в месяц
- матрицы строятся полчаса

2016

- 10.1 млн пользователей в месяц
- матрицы строятся 11 часов

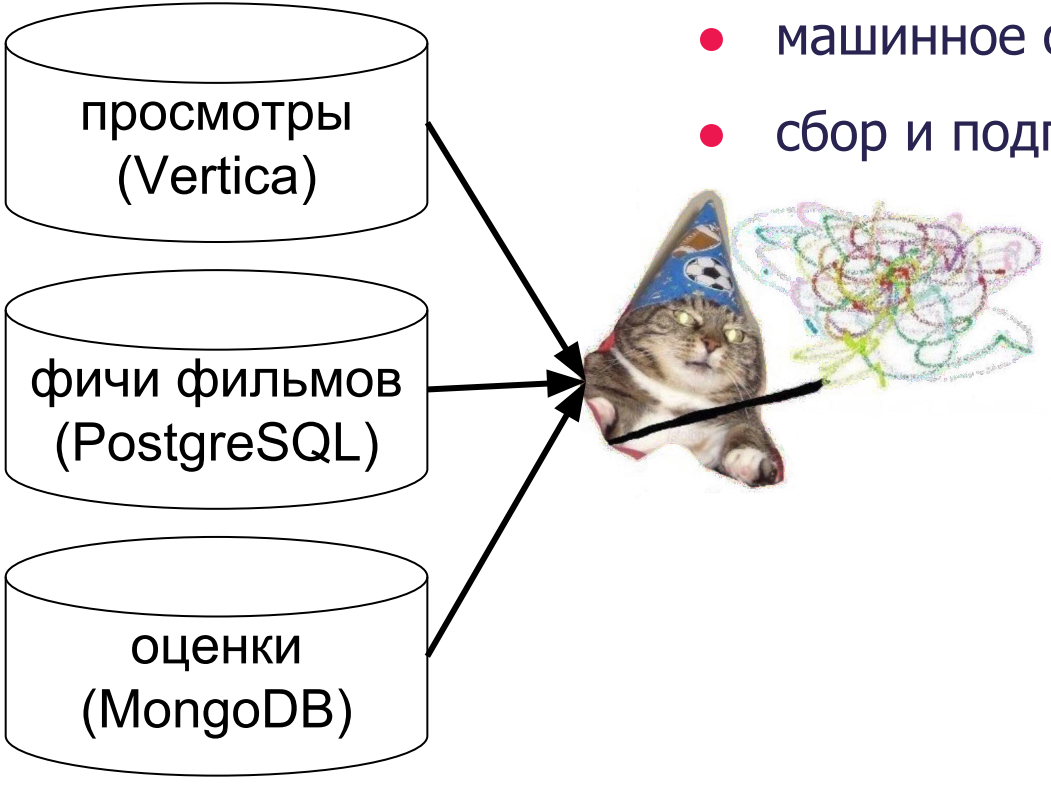
У нас проблема

- хотим более сложные модели
- хотим большую глубину истории
- хотим экспериментировать

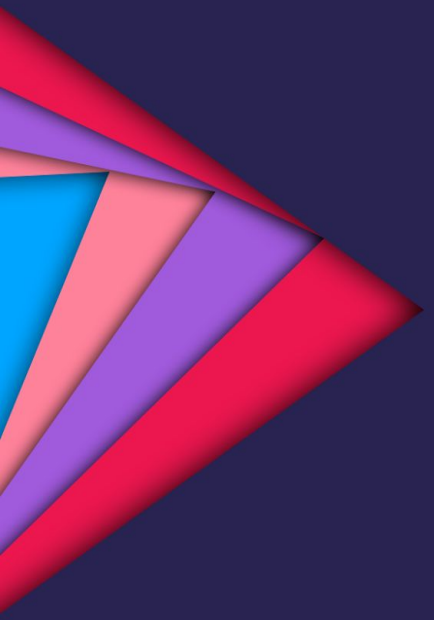
но

- текущая модель строится 11 часов
- работает только на одном сервере

На что уходит 11 часов



- машинное обучение - 2 мин
- сбор и подготовка данных - 11 ч

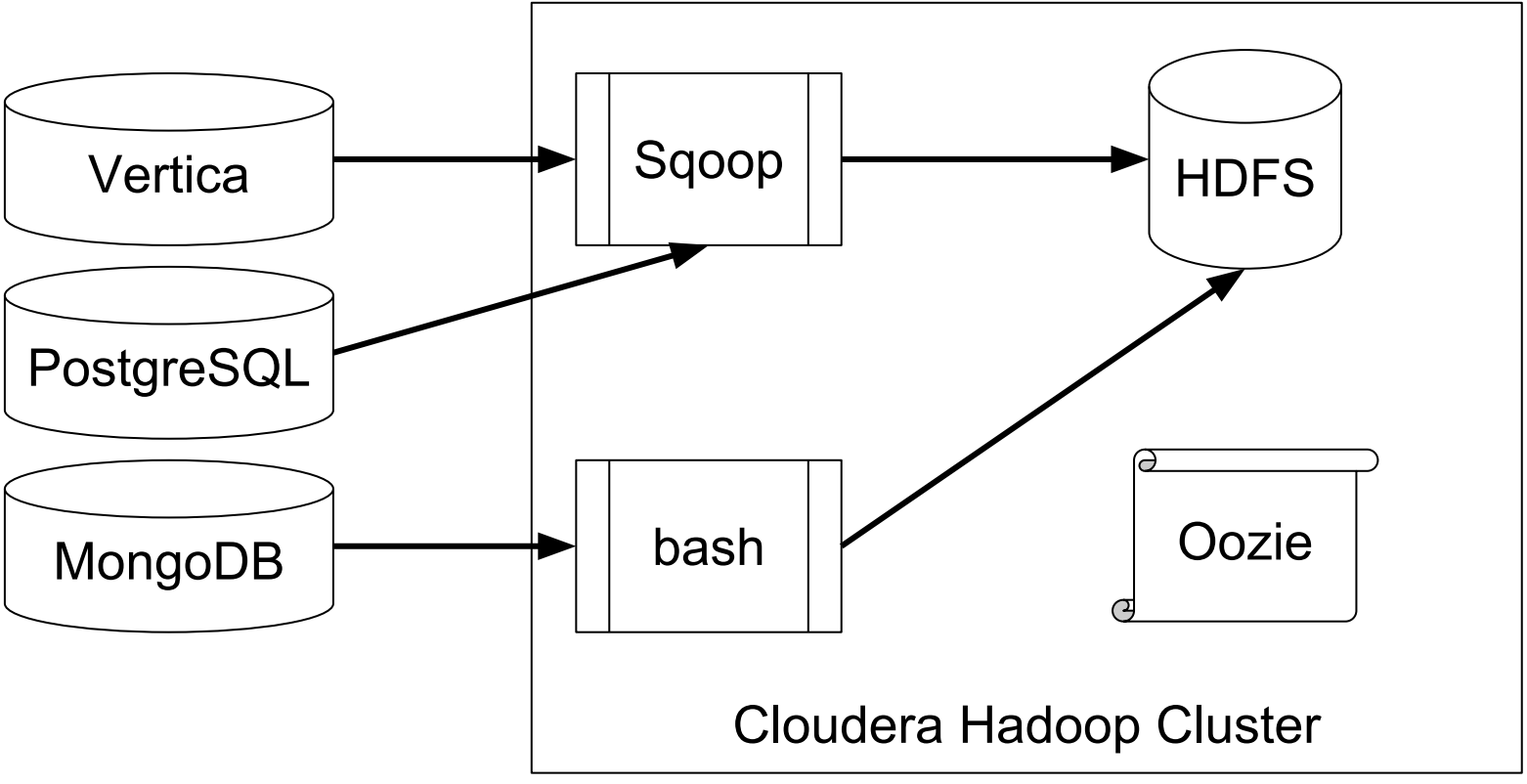


ivi Data Lake

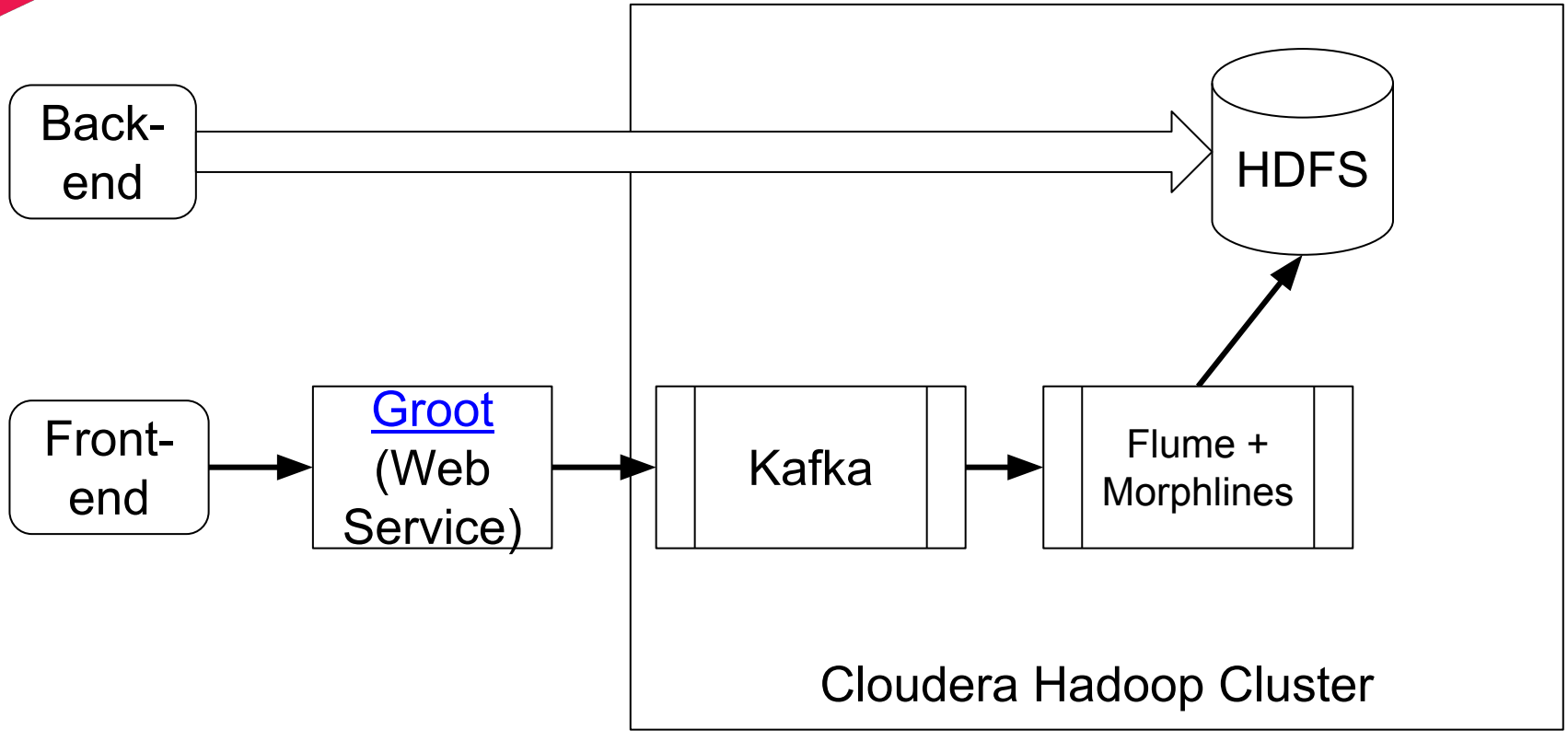
Отделяем DWH от ML

- Vertica - хорошо, но дорого
- Hadoop - много места почти даром
- Cloudera - удобный UI, можно бесплатно

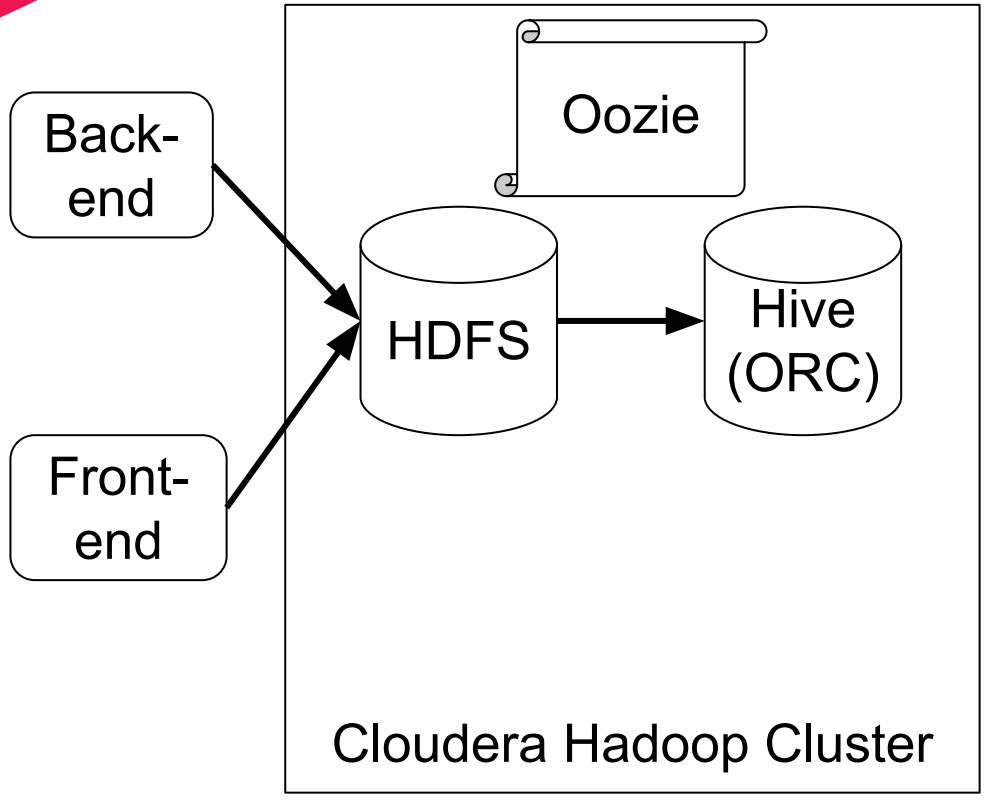
Собираем данные back-end'а



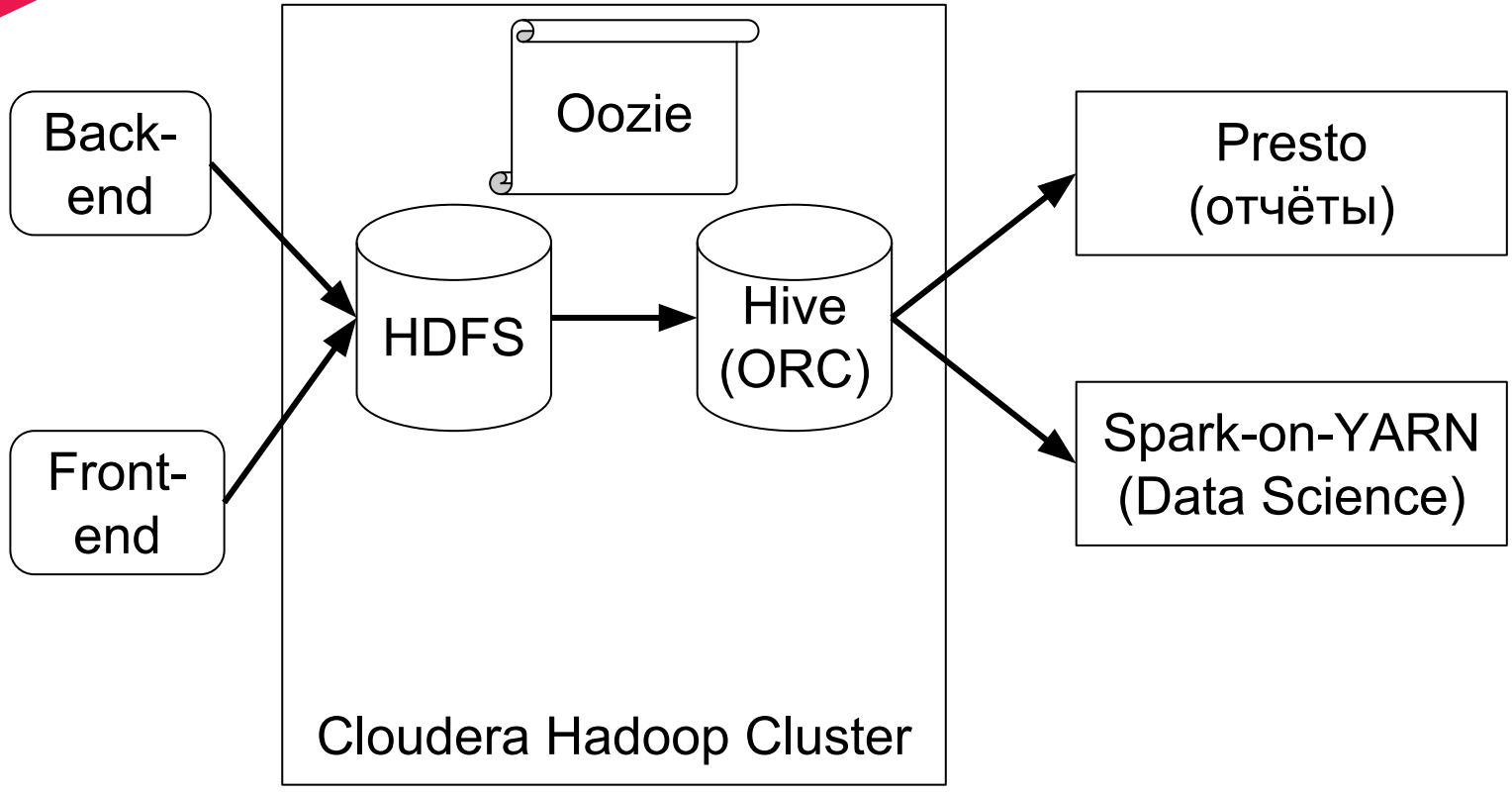
Добавляем события приложений



Добавляем метаданные



Работаем с данными в хранилище



Что изменил Spark

со Spark:

- все узлы кластера
- год исторических данных
- находит SVD с помощью ALS

без Spark:

- один сервер
- месяц исторических данных
- умножает матрицу на матрицу

‘Lightning-fast cluster computing’?

со Spark:

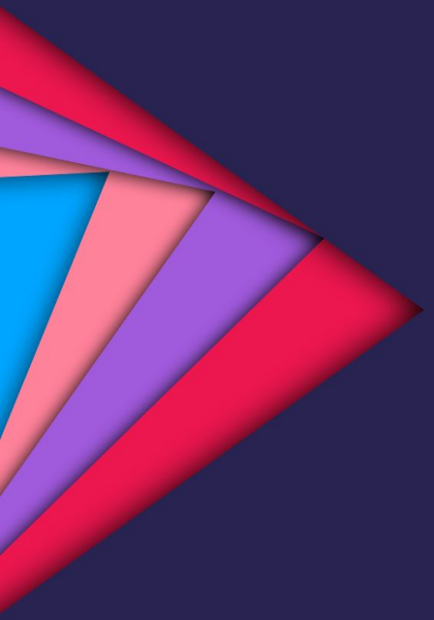
- один источник данных
- работает час
- ML + feature engineering

без Spark:

- разрозненные источники
- работает 11 часов
- сбор данных + ML

Чего добились с помощью Spark

- больше данных
- более сложные модели
- конверсия в смотрение выросла на несколько пунктов



Spark для Data Scientist: проблемы и их решения

2017

За что мы (не) любим Spark

- можно писать на Python

За что мы (не) любим Spark

- можно писать на Python
- правда, он падает с Java Stack Trace

```
java.lang.OutOfMemoryError: Java heap space
        at scala.collection.mutable.ArrayBuilder$ofInt.mkArray(ArrayBuilder.scala:323)
        at scala.collection.mutable.ArrayBuilder$ofInt.resize(ArrayBuilder.scala:329)
        at scala.collection.mutable.ArrayBuilder$ofInt.ensureSize(ArrayBuilder.scala:341)
        at scala.collection.mutable.ArrayBuilder$ofInt.$plus$eq(ArrayBuilder.scala:346)
        at scala.collection.mutable.ArrayBuilder$ofInt.$plus$eq(ArrayBuilder.scala:316)
        at scala.collection.generic.Growable$$anonfun$$plus$plus$eq$1.apply(Growable.scala:59)
        at scala.collection.generic.Growable$$anonfun$$plus$plus$eq$1.apply(Growable.scala:59)
        at scala.collection.IndexedSeqOptimized$class.foreach(IndexedSeqOptimized.scala:33)
        at scala.collection.mutable.ArrayOps$ofInt.foreach(ArrayOps.scala:234)
        at scala.collection.generic.Growable$class.$plus$plus$eq(Growable.scala:59)
        at scala.collection.mutable.ArrayBuilder$ofInt.$plus$plus$eq(ArrayBuilder.scala:359)
        at scala.collection.mutable.ArrayBuilder$ofInt.$plus$plus$eq(ArrayBuilder.scala:316)
        at org.apache.spark.ml.recommendation.ALS$UncompressedInBlockBuilder.add(ALS.scala:1158)
        at org.apache.spark.ml.recommendation.ALS$$anonfun$23$$anonfun$apply$16.apply(ALS.scala:1376)
        at org.apache.spark.ml.recommendation.ALS$$anonfun$23$$anonfun$apply$16.apply(ALS.scala:1375)
        at scala.collection.Iterator$class.foreach(Iterator.scala:893)
        at org.apache.spark.util.collection.CompactBuffer$$anon$1.foreach(CompactBuffer.scala:115)
```

За что мы (не) любим Spark

- можно писать на Python
- правда, он падает с Java Stack Trace
- Spark DataFrames такие же удобные, как в pandas

За что мы (не) любим Spark

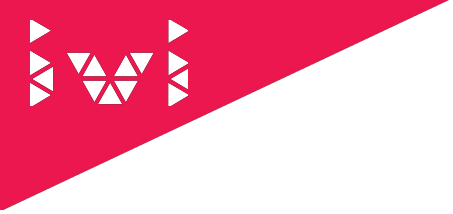
- можно писать на Python
- правда, он падает с Java Stack Trace
- Spark DataFrames такие же удобные, как в pandas
- правда, иногда доступно только RDD API

За что мы (не) любим Spark

- можно писать на Python
- правда, он падает с Java Stack Trace
- Spark DataFrames такие же удобные, как в pandas
- правда, иногда доступно только RDD API
- есть своя библиотека ML

За что мы (не) любим Spark

- можно писать на Python
- правда, он падает с Java Stack Trace
- Spark DataFrames такие же удобные, как в pandas
- правда, иногда доступно только RDD API
- есть своя библиотека ML
- ...или MLLib?:) SPARK-4591



OOM

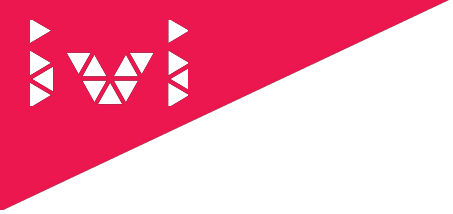
- ✓ добавить памяти executor’ам

OOM

- ✓ ~~добавить памяти executor’ам~~
- ✓ много маленьких executor’ов
- ✓ настройки партицирования:
  - ✓ spark.sql.shuffle.partitions
  - ✓ spark.default.parallelism

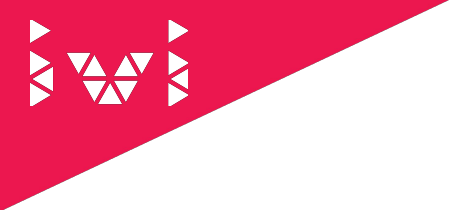
OOM

- ✓ ~~добавить памяти executor’ам~~
- ✓ много маленьких executor’ов
- ✓ настройки партицирования:
  - ✓ spark.sql.shuffle.partitions
  - ✓ spark.default.parallelism
  - ✓ ALS.numItemBlocks
  - ✓ ALS.numUserBlocks



Stack Overflow

- ✓ ставим check point’ы:
  - SparkContext.setCheckpointDir
  - ALS.setCheckpointInterval
  - начиная с 2.1 есть и для Dataframe’ов



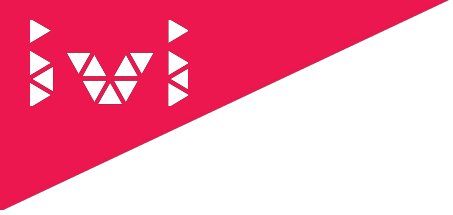
RDD vs DataFrame

проблема

- DataFrame в pyspark быстрее RDD в разы
- использование python UDF всё портит

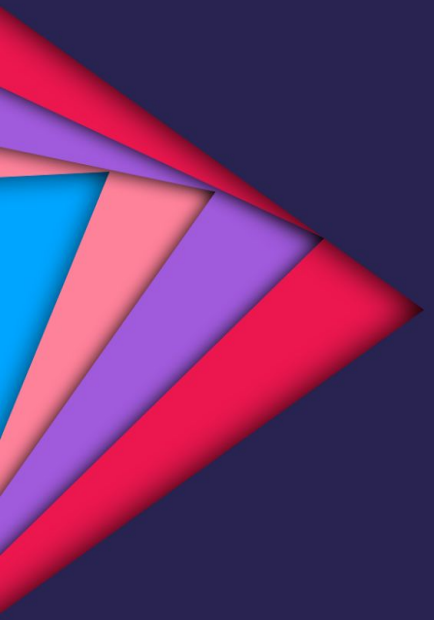
решение

- писать UDF на Java/Scala (начиная со Spark 2.1)



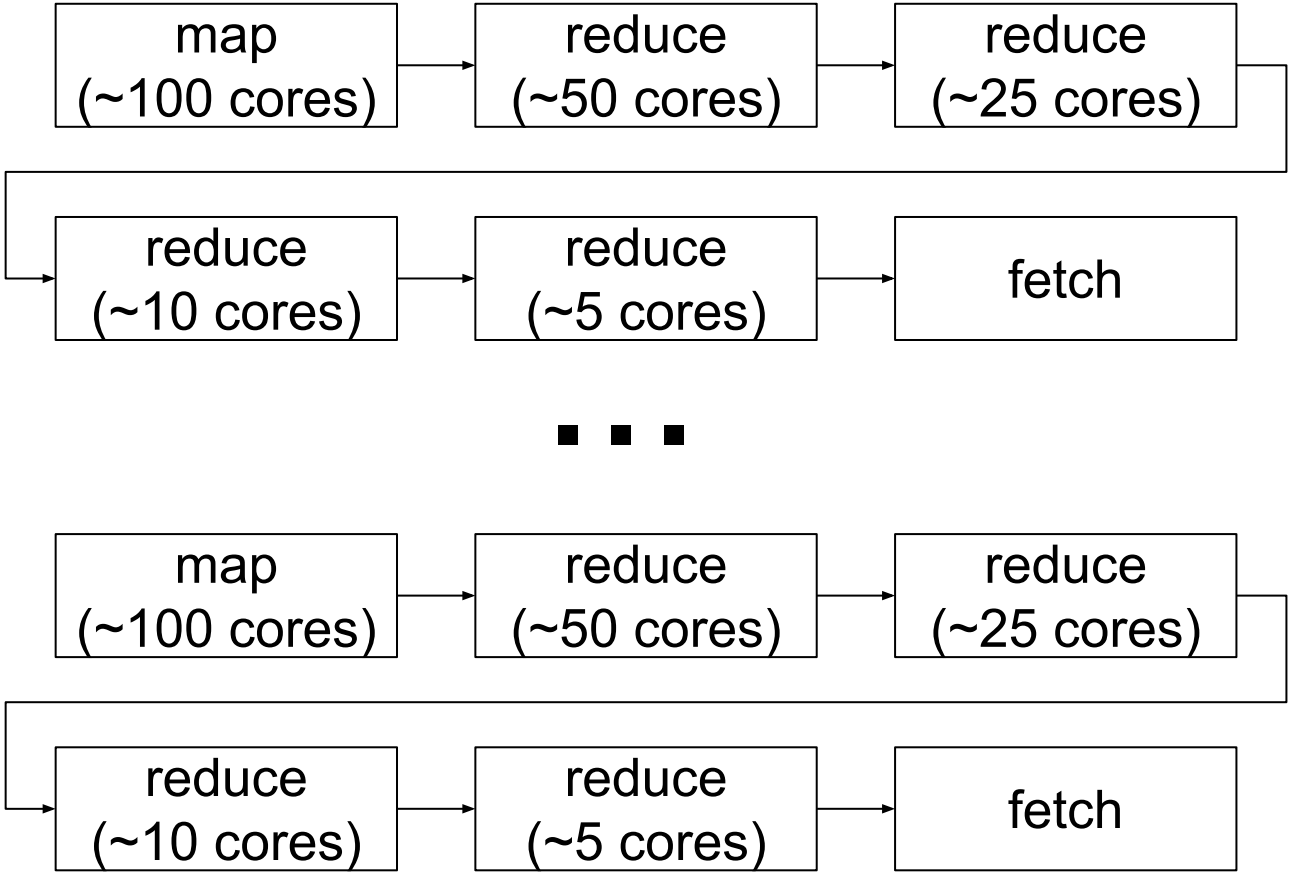
ML vs MLLib

- хотим: рекомендации item-to-item по схожести
- нужны: большие разреженными матрицами
- в Spark есть **распределённые** разреженные матрицы
- но только pypspark.**mllib**.linalg.distributed
- Scala?



Кейс: параллельный подбор метапараметров на Spark

Распределённое машинное обучение



Распределённое машинное обучение

- происходит долго
- потребность в ресурсах по времени неравномерна
- сложно запустить даже две модели параллельно

Параллельный подбор метапараметров

“А где тут писать njobs=10?”

два ответа:

- параллелизм на уровне YARN
- параллелизм на уровне Spark

Параллелизм на уровне YARN

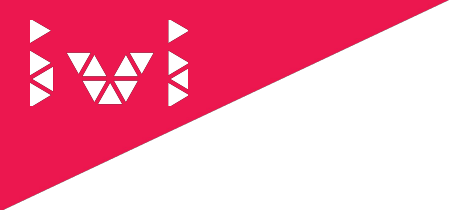
- один набор параметров - одно Spark Application
- YARN выделяет ресурсы заранее
- но требуются они неравномерно по времени
- Dynamic Resource Allocation?

multiprocessing в pyspark

- одно Spark приложение
- одна модель - один процесс Python
- модели в разных Job’ах строятся последовательно
- а надо было параллельно:(

threading в pyspark

- одно Spark приложение
- одна модель - один поток в Python
- все отправляют обновление одному JVM-объекту
- а надо было несколько моделей:(

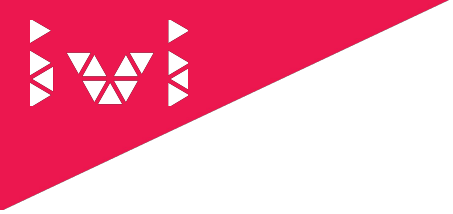


Два варианта решения

- разобраться, почему pyspark так работает
- написать патч
- сделать pull request

или

- написать подбор параметров на Scala



Два варианта решения (на самом деле, нет)

- разобраться, почему pyspark так работает (**Scala**)
- написать патч (Python + **Scala**)
- сделать pull request

или

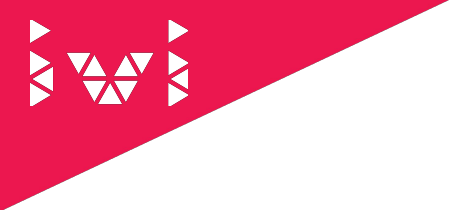
✓ написать подбор параметров на **Scala**

Простой и понятный код на Python

```
from pyspark.ml.recommendation import ALS


data = spark.sql("""
    SELECT user, item, rating
    FROM user_item_view
""").cache()
model = ALS().fit(data)
```

Сложный и непонятный код на Scala

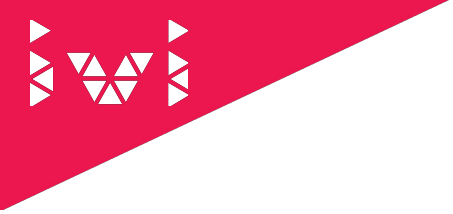
```
import org.apache.spark.ml.recommendation.ALS


val data = spark.sql("""
    SELECT user, item, rating
    FROM user_item_view
""").cache()
val model = (new ALS).fit(data)
```



Обычный код на Scala

```
paramSets.map(
  paramSet =>
    buildAndEvaluateModel(
      trainDataFrame,
      testDataFrame,
      paramSet
    )
)
```

выполняется последовательно как цикл foreach



Параллельный код на Scala

```
paramSets.par.map(
  paramSet =>
    buildAndEvaluateModel(
      trainDataFrame,
      testDataFrame,
      paramSet
    )
)
```

выполняется в несколько JVM Threads

Важные настройки

- `spark.scheduler.mode: FAIR`
- `spark.driver.extraJavaOptions:`

  `"-Dscala.concurrent.context.maxThreads=10"`

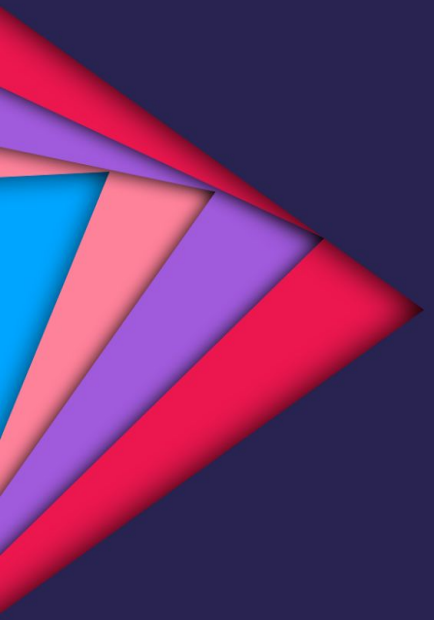
Что выяснили в этом кейсе

- pyspark может работать *очень* странно
- программировать на Scala в Spark несложно
- параллельное обучение моделей экономит время до 5 раз

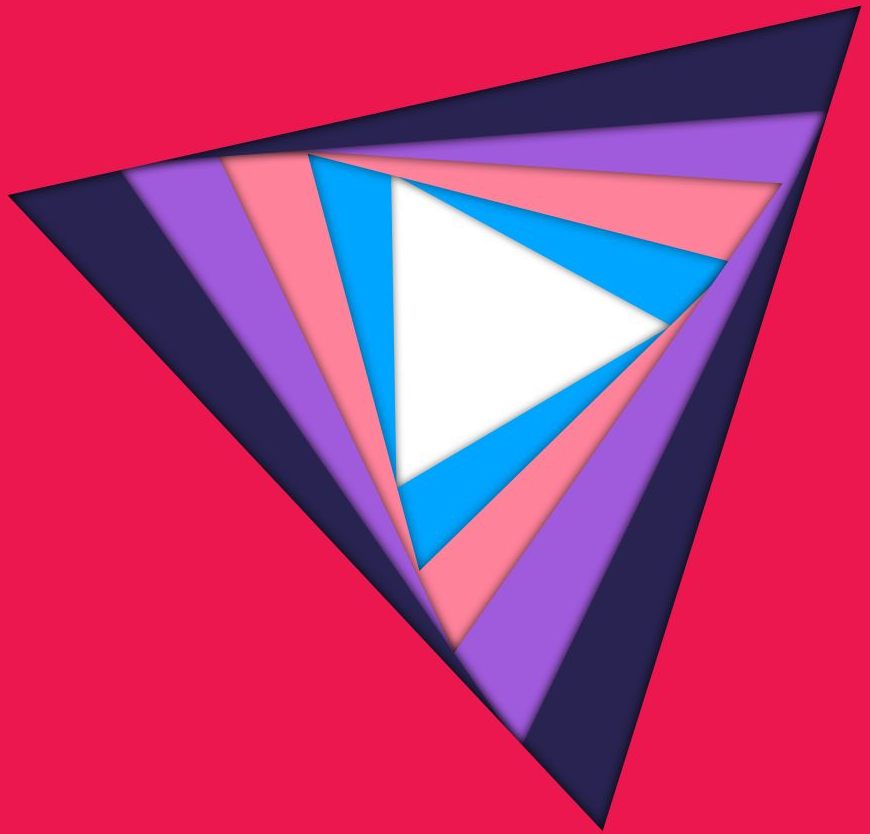
Резюме

Распределённое ML на Big Data

- позволяет забыть о проблеме увеличения объёма данных
- но его действительно непросто приготовить
- проще делать на Scala, чем на Python
- по крайней мере, если использовать Spark
- Spark вполне подходит для рекомендательных систем
- если, конечно, уже есть Big Data инфраструктура



Вопросы и ответы



Спасибо
за внимание!

Борис Шминке
bshminke@ivi.ru