Большие данные на службе рекламного бизнеса

Илья Маркин

Курс “Специалист по большим данным”
Лаборатория новых профессий

13.10.2018

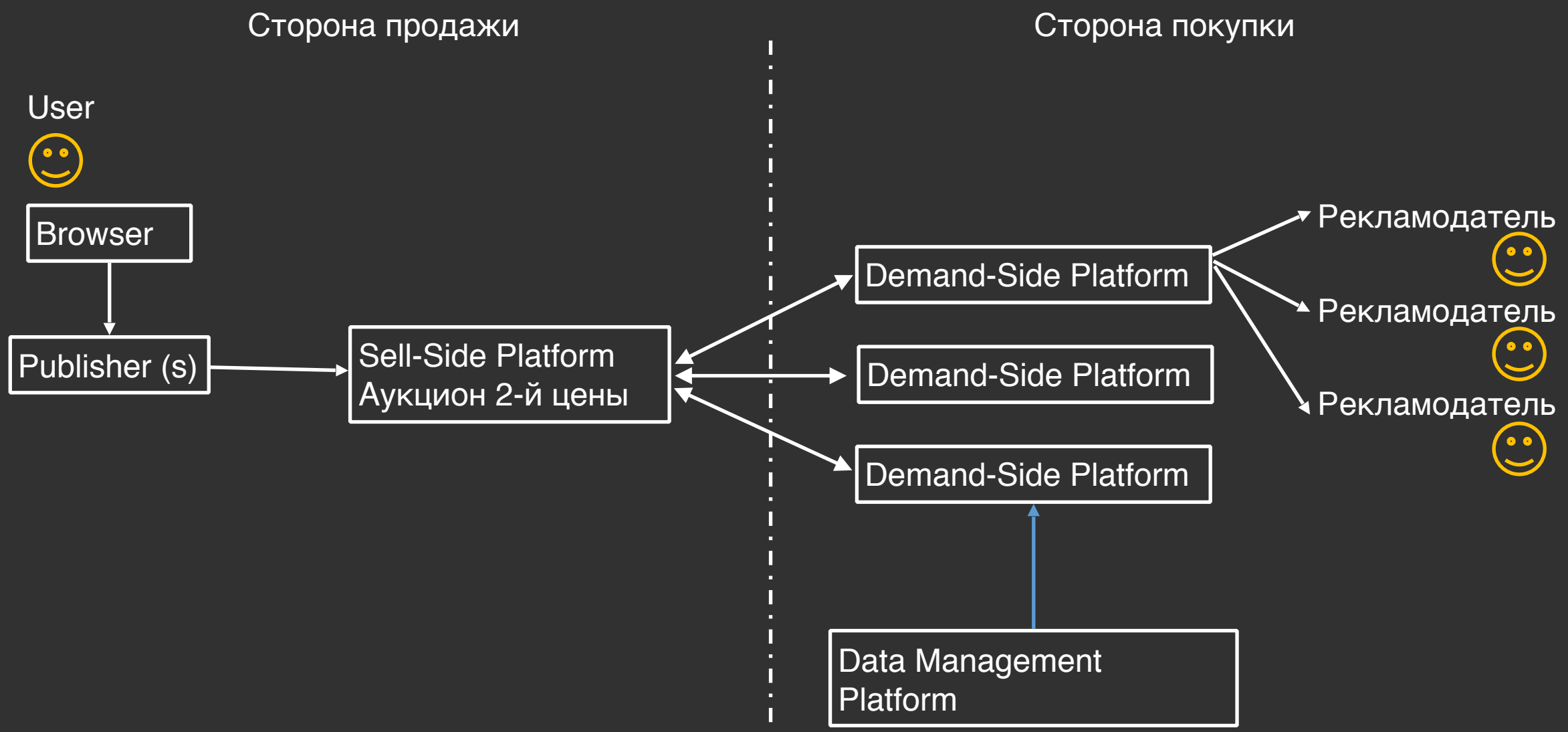
RTB (Real Time Bidding)

- Подход по продаже рекламы по модели аукциона в реальном времени
- Торги происходят за каждый конкретный показ баннера
- Конкуренция знаниями и алгоритмами

Основные понятия RTB

- **User** - произвольный пользователь интернета
- **Publisher** – сайт, который размещает рекламу
- **SSP** – Sell Side Platform - рекламная биржа, которая проводит аукцион. Как правило обслуживает большое количество Publisher’ов
- **Advertiser** – рекламодатель, заинтересованный в показе рекламы (CocaCola, Adidas, агентства)
- **DSP** – Demand Side Platform – платформа, которая представляет интересы рекламодателей. Как правило, агрегирует много Advertiser (брокер)
- **DMP** – Data Management Platform - обладает большим количеством знаний о пользователях

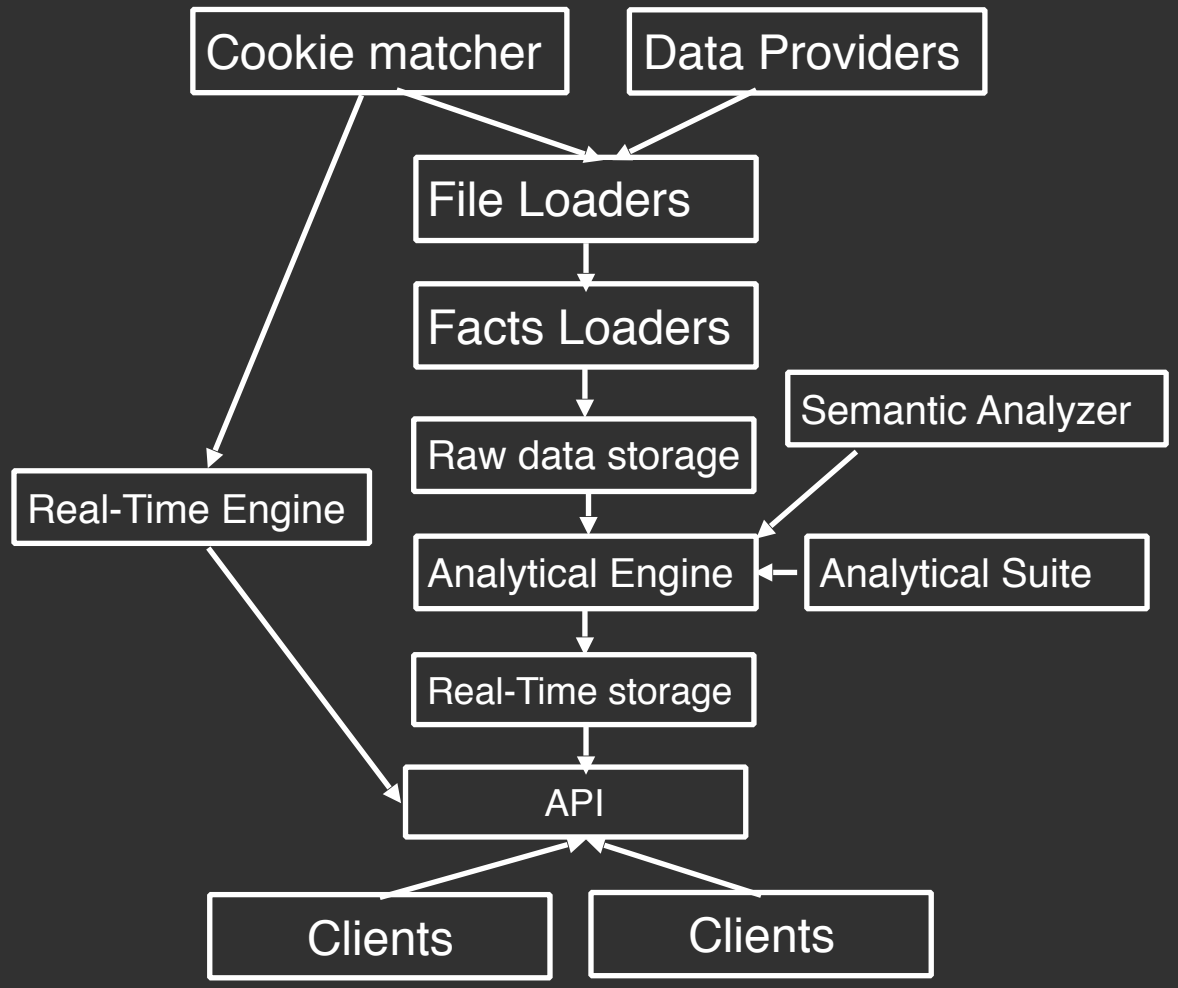
Как работает RTB



DMP: Основные концепции

- Не должно быть единой точки отказа
- Горизонтальная масштабируемость
- Возможность построения любых сколь угодно сложных сегментов над пользователями
- Минимизация ручного труда

Архитектура реальной DMP

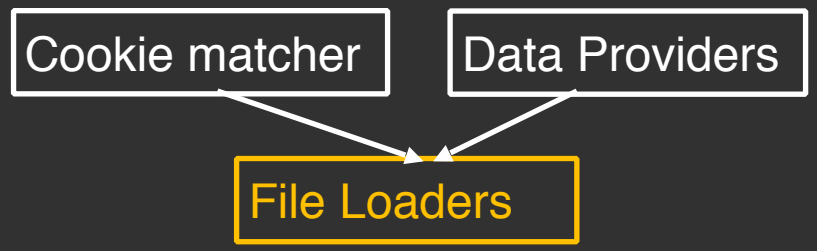


Cookie Matcher

Cookie matcher

- Выполняет cookie matching (механизм синхронизации cookie)
- Собирает данные по referer’ам по партнерской сети

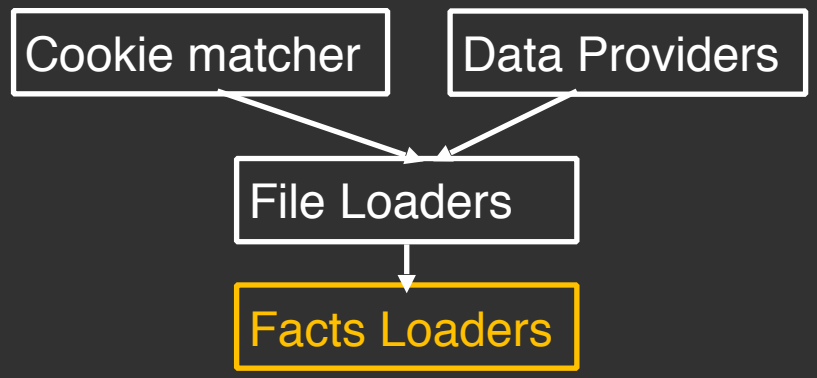
Files Loaders



- Универсальный framework позволяющий загрузить данные в максимально сыром виде
- Сохраняет данные в виде файлов на HDFS
- Сохраняем в максимально сыром виде не выполняя никаких преобразований

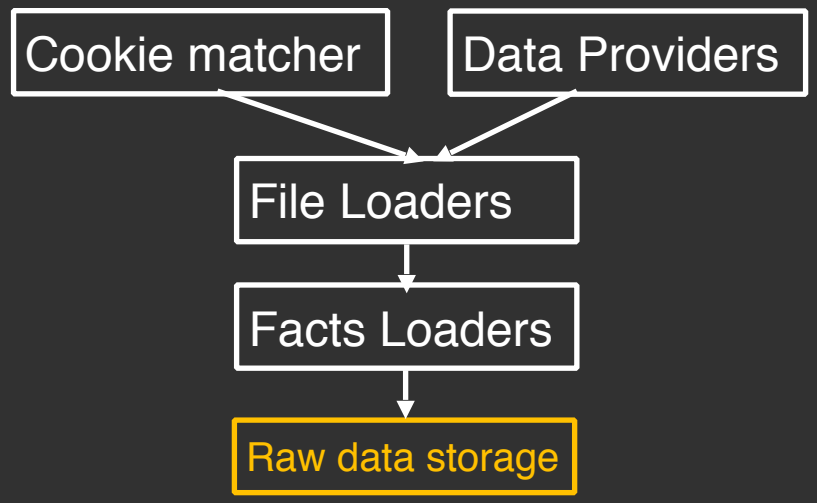
Facts Loaders

- Framework для загрузки данных в хранилище фактов посещений
- Периодически запускается и перекладывает накопившиеся данные



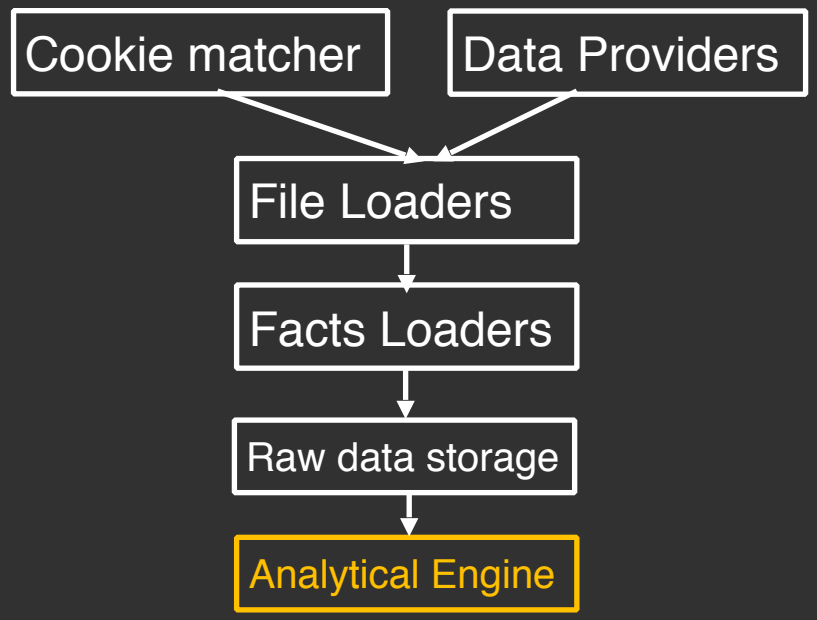
Raw data storage

- Основано на Hbase (колоночная база данных из стека hadoop)
- Данные отсортированы по пользователям
- Данные разобраны по типам (URL – отдельно, поисковики – отдельно)



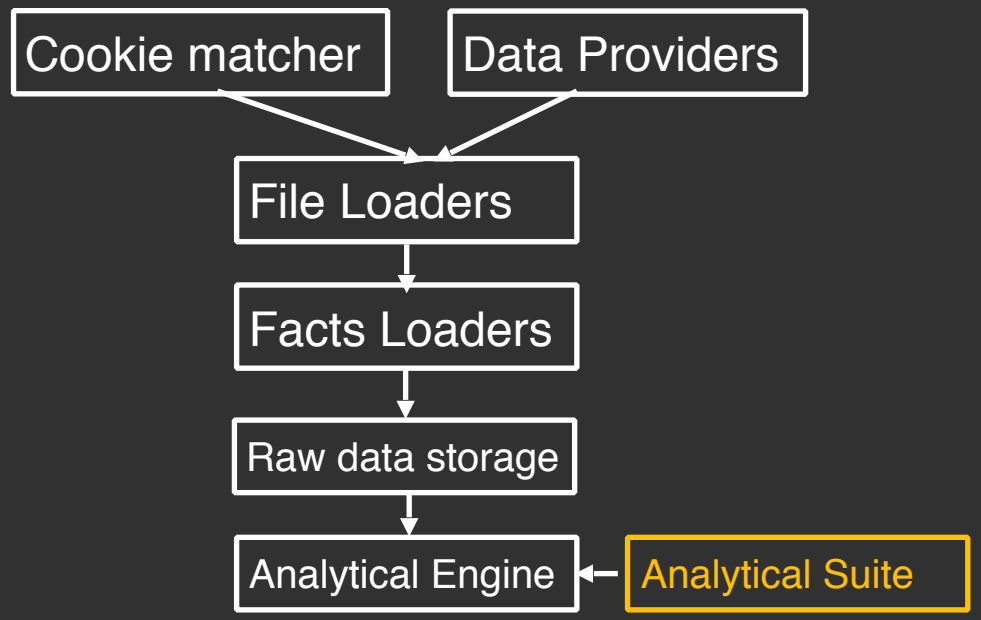
Analytical Engine

- Map-Reduce Job
- Загружает из внешнего хранилища (MongoDB) скрипты для разметки пользователей сегментами
- Результаты складывает в Real-Time storage
- Запускается раз в сутки



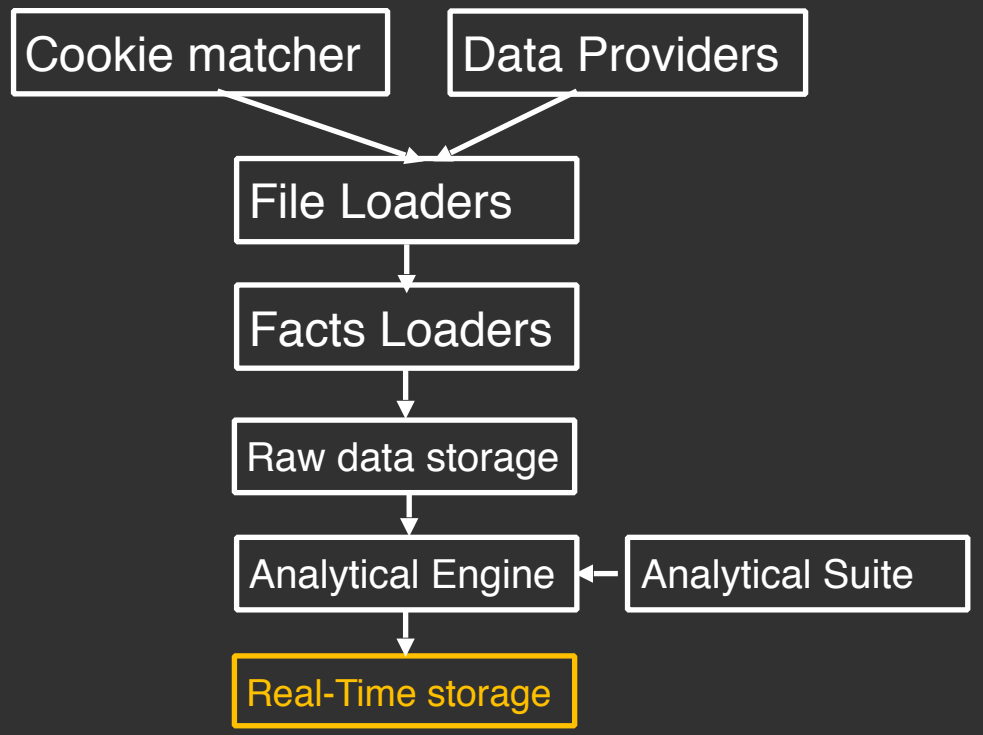
Analytical Suite

- Набор инструментов для аналитика
- Библиотеки для Python для доступа к данным
- Средства для отладки Groovy-скриптов
- Средства сериализации/ десериализации данных
- Средства визуализации данных



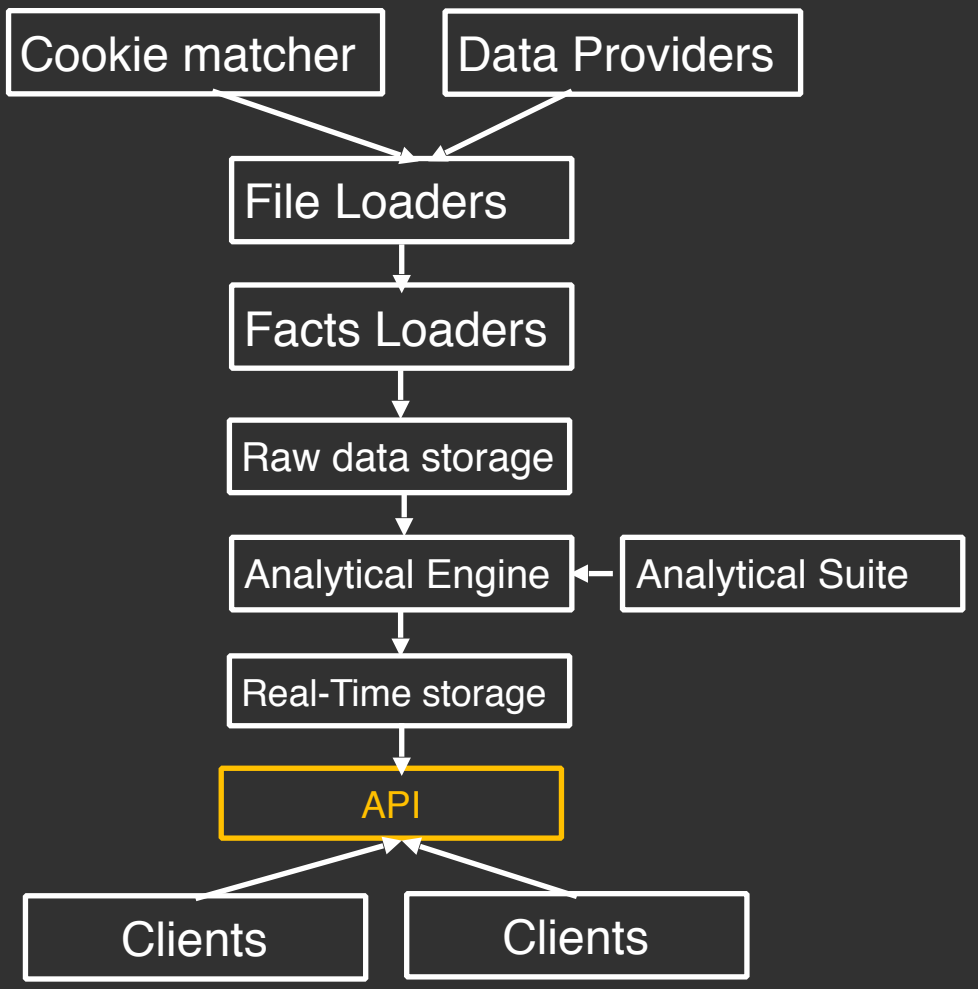
Real Time Storage

- Реализовано на основе Aerospike (очень быстрая и отказоустойчивая key-value база данных)
- Хранит сегменты в привязке к куке
- Хранит индексы
- Должен быть очень быстрым, отказоустойчивым и масштабируемым



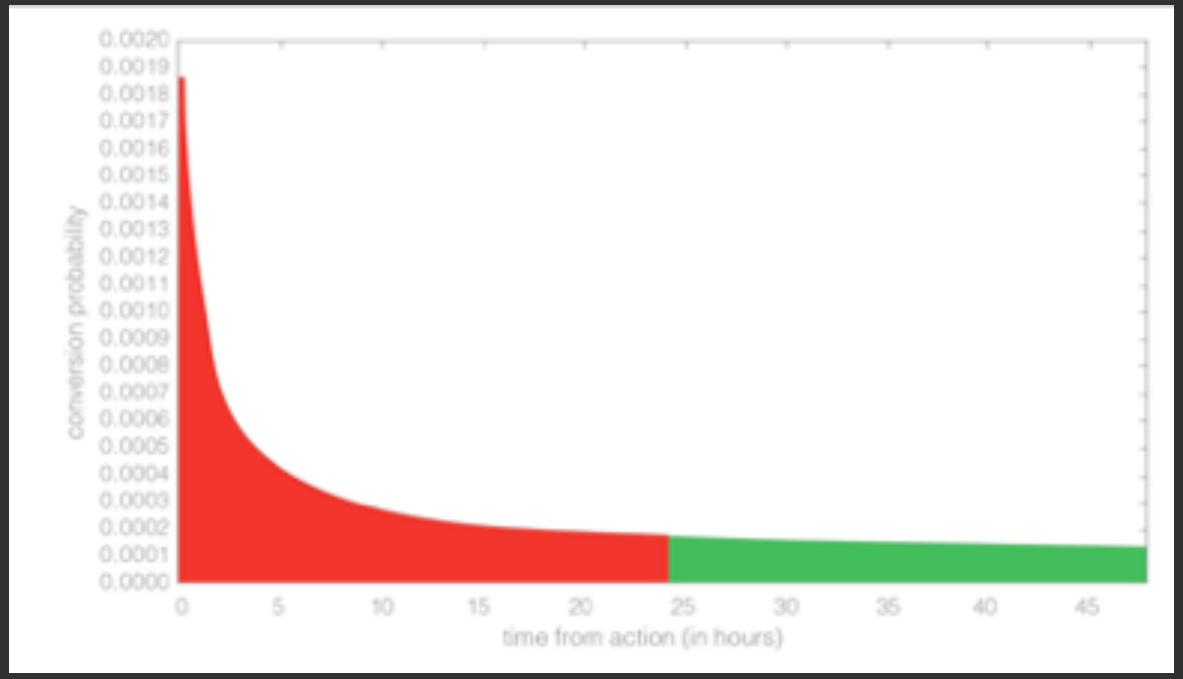
API

- HTTP-сервис, точка доступа к DMP для клиентов
- Основное назначение – обогатить данными запрос
- Запросы во внешние DMP
- Главное - ответить



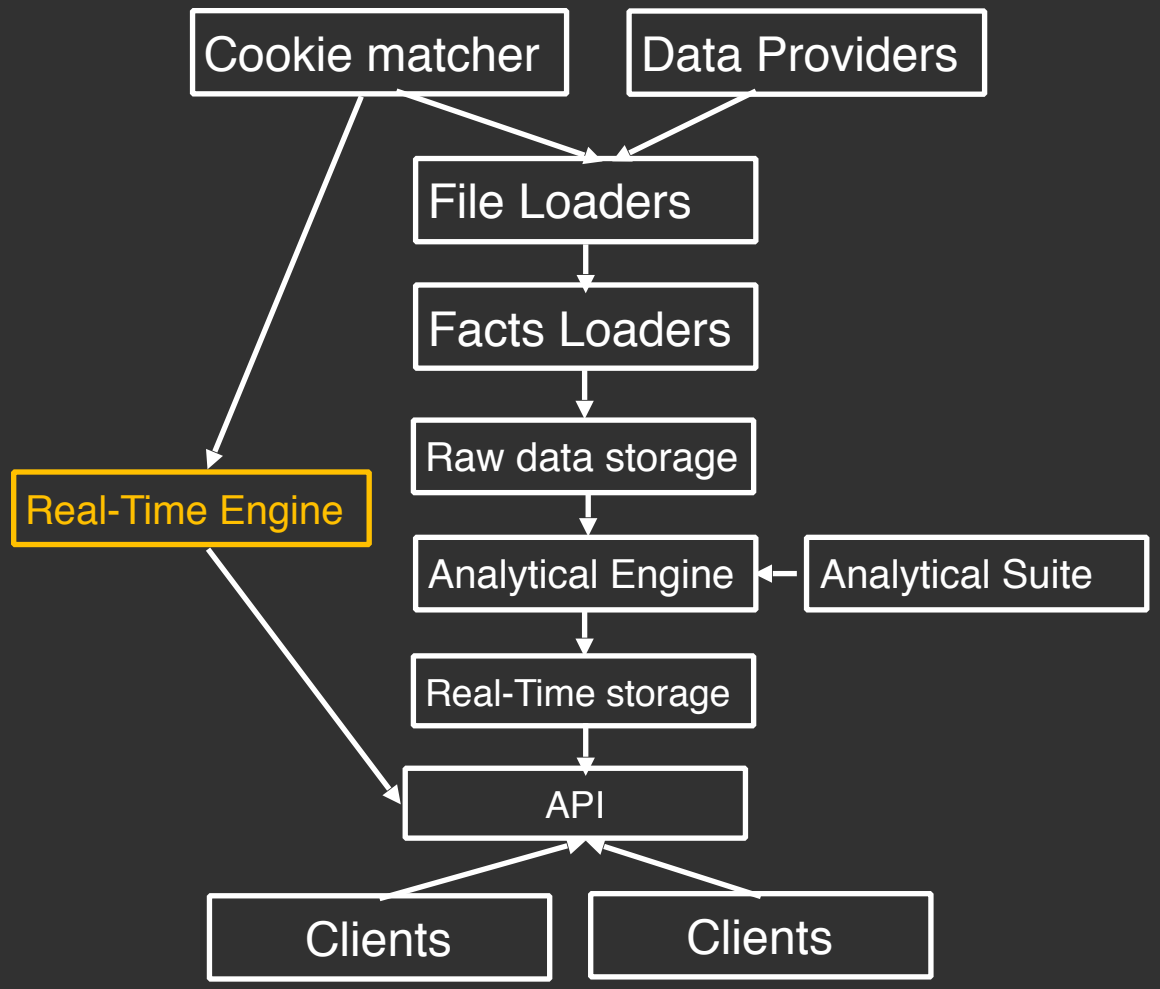
Real-Time engine

- Сутки – это очень долго
- Некоторые сегменты актуальны только в течение нескольких часов
- Пример: вероятность покупки телефона через N часов после просмотра



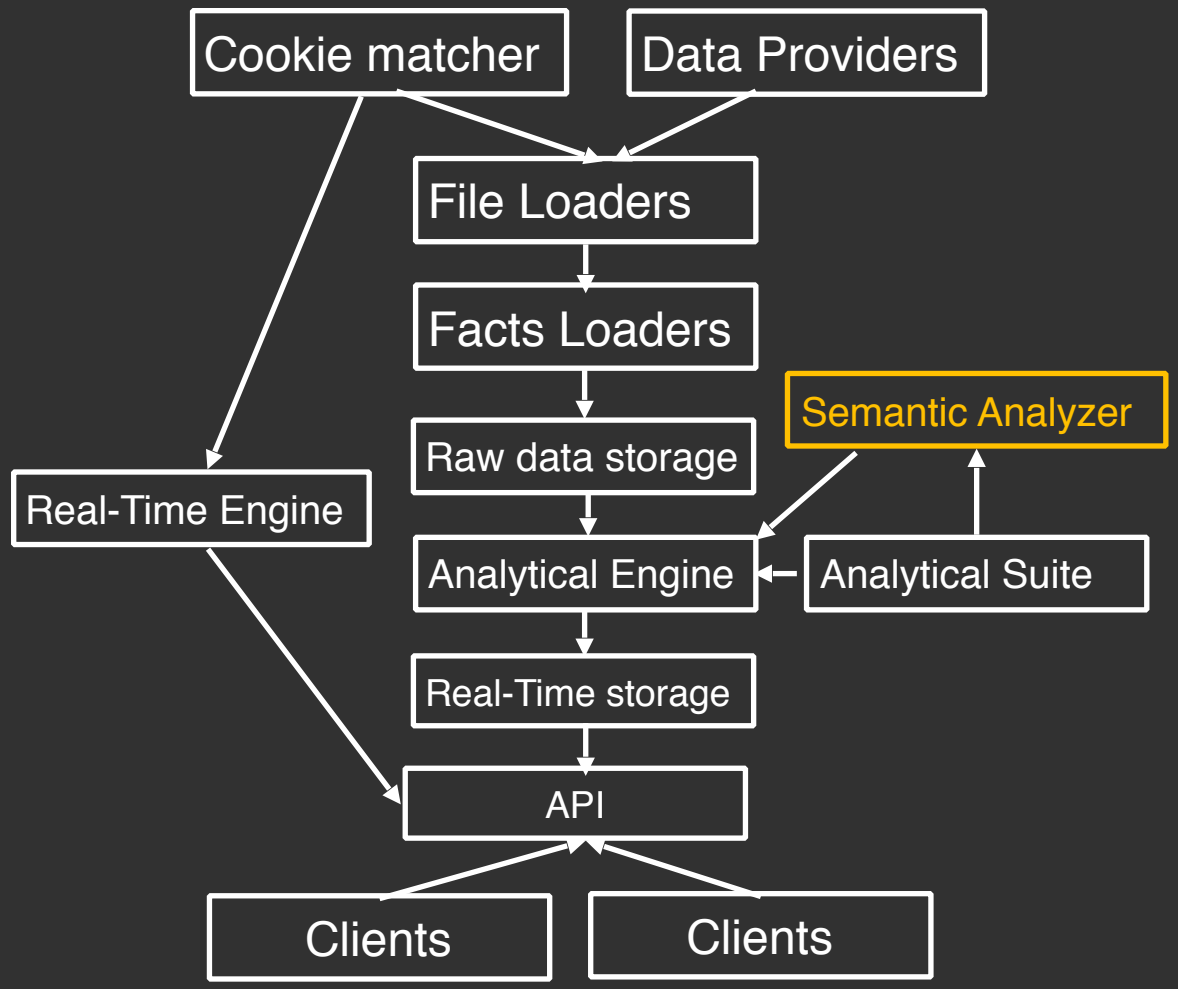
Real-Time Engine

- Выполняются те же скрипты, что и в Analytical Engine
- Можно рассчитывать сегменты произвольной сложности



Семантический анализатор

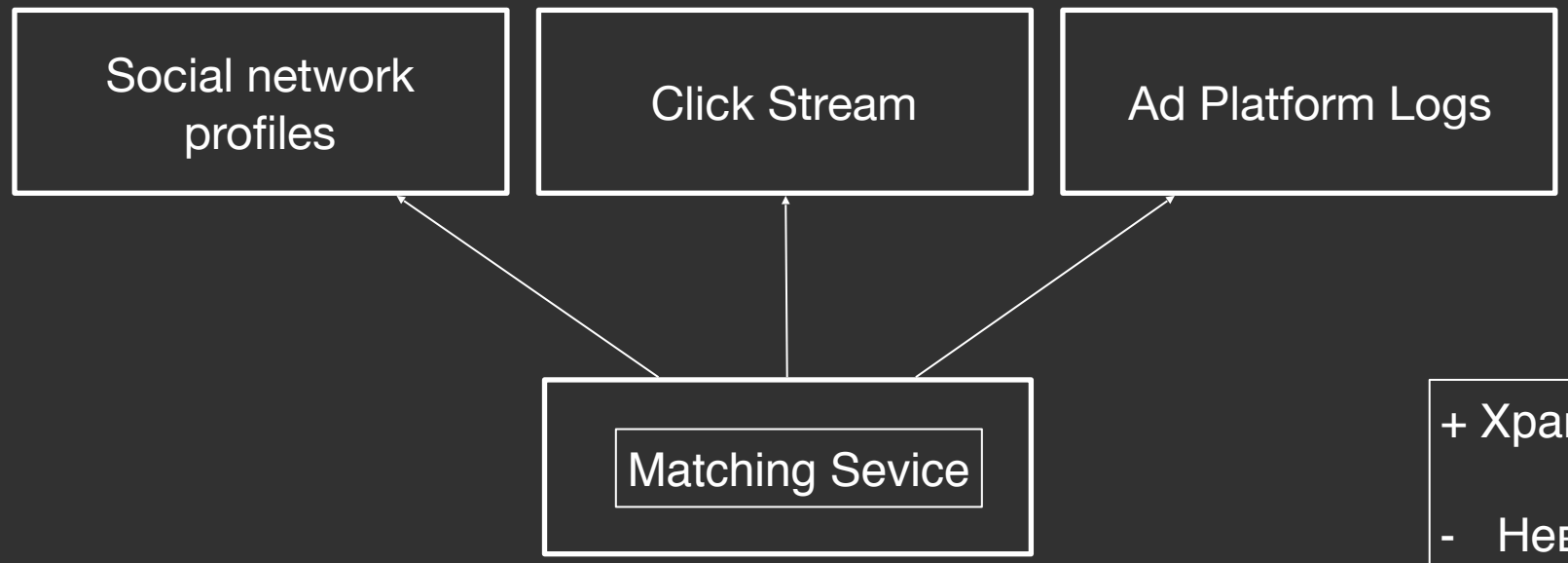
- Составление обучающей выборки на основе поисковых запросов
- Иерархическая классификация
- 15 % выборка для feature selection, 70% обучающая, 15% контрольная
- Взвешенная F-мера для оценки качества классификации.
- Визуализация качества.
- Подробности: http://habrahabr.ru/company/dca/blog/261677/



Challenge #1. Хранение данных

- Задача - сохранить информацию о поведении пользователей
- Сотни миллионов пользователей
- Десятки миллиардов фактов
- Десятки поставщиков, данные каждого немного разные.

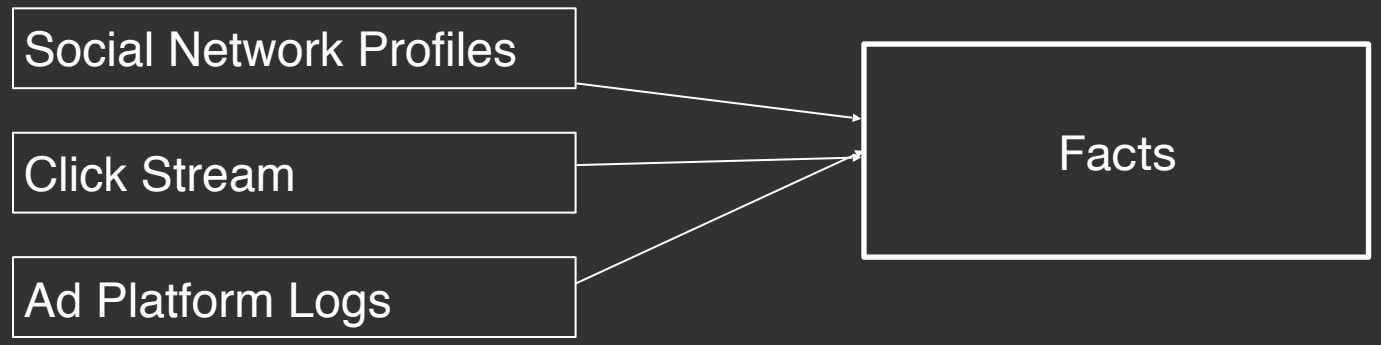
Challenge #1. Много таблиц в Hbase



+ Хранятся полные данные

- Невозможно запомнить где какие данные
- Необходимость постоянных JOIN' ов

Challenge #1. Одна большая таблица в hbase



+ Все данные хранятся вместе, однотипные данные хранятся в одних колонках

+ Легко достать все данные относящиеся к одному пользователю

- При отсутствии синхронизации на момент заливки – теряем данные

Challenge #2. Запись данных в Hbase

- Десятки тысяч событий в секунду
- Ограниченный размер кластера (~10 машин в HBASE – кластере)

Challenge #2. Запись данных. Put

+ Данные доступны в Hbase в реальном времени(сразу как пришли)

- Нужно много памяти для MemStore

- Не очень высокая скорость записи

    Отставание от realtime-потока примерно в 1.5 раз

Challenge #2. Запись данных. Bulk Load

+ Можно записывать практически неограниченные обхемы

+ Меньше требований к оперативной памяти.

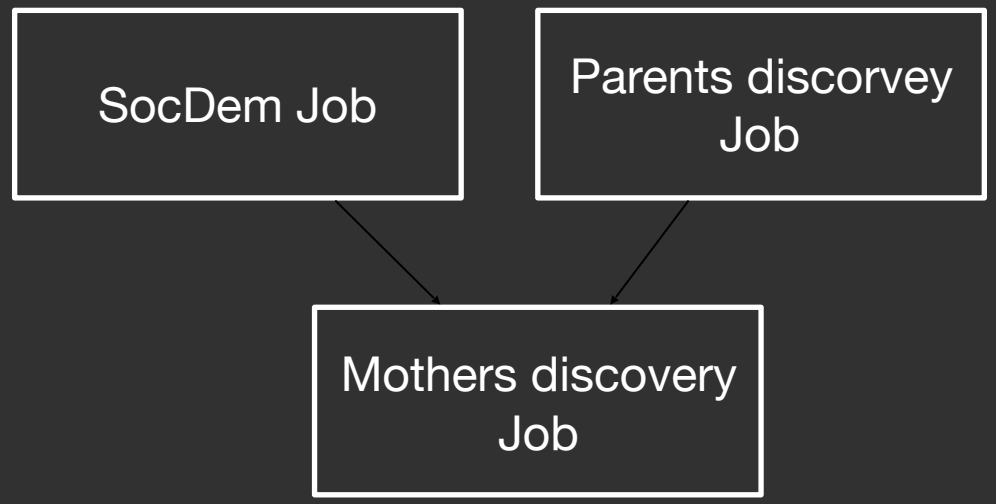
- Задержка по сравнению с Real Time

-Мелкие Data-файлы для небольших поставщиков, увеличение интенсивности Compaction’ов

Challenge #3. Analytical Engine

- Задача – делать выводы о профиле пользователя на основе фактов хранящихся в Hbase
- Сотни миллионов пользователей
- Сотни характеристик пользователей

Challenge #3. Analytical Engine: Цепочка Map-Reduce job-ов

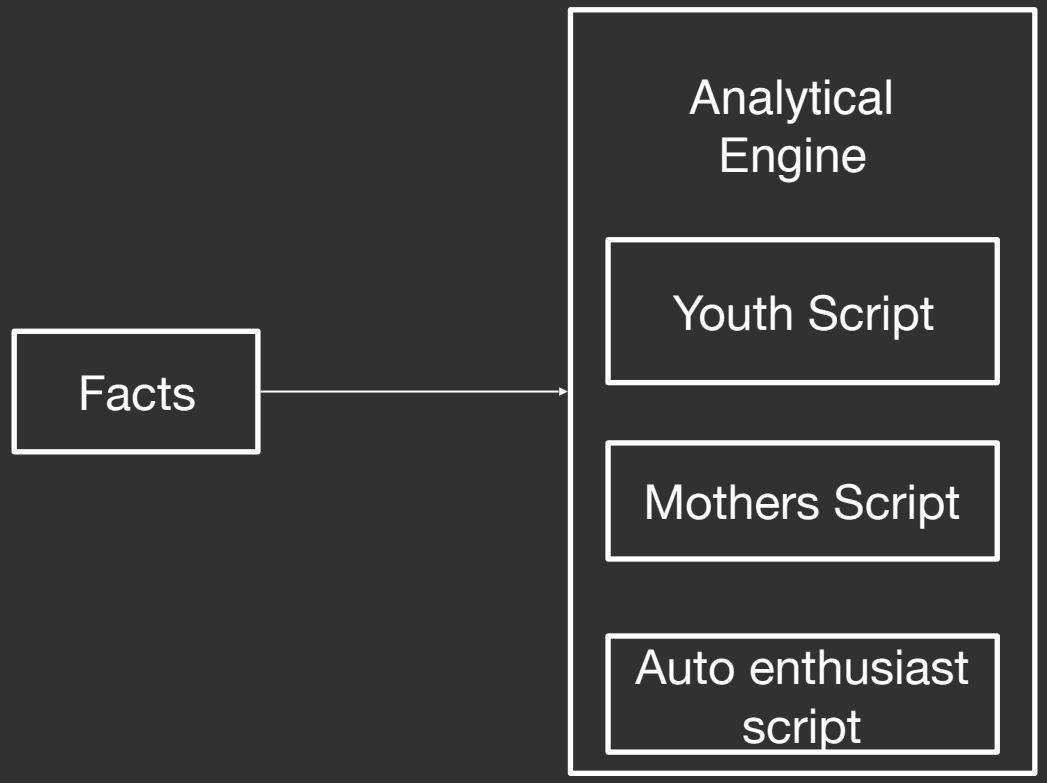


+ Универсальность, вычисление сколько угодно сложных комбинированных сегментов

- Необходимость постоянных Join-ов

- Много MapReduce, большое время выполнения

Challenge #3. 1 большой Map-Only. 1 сегмент = 1 скрипт



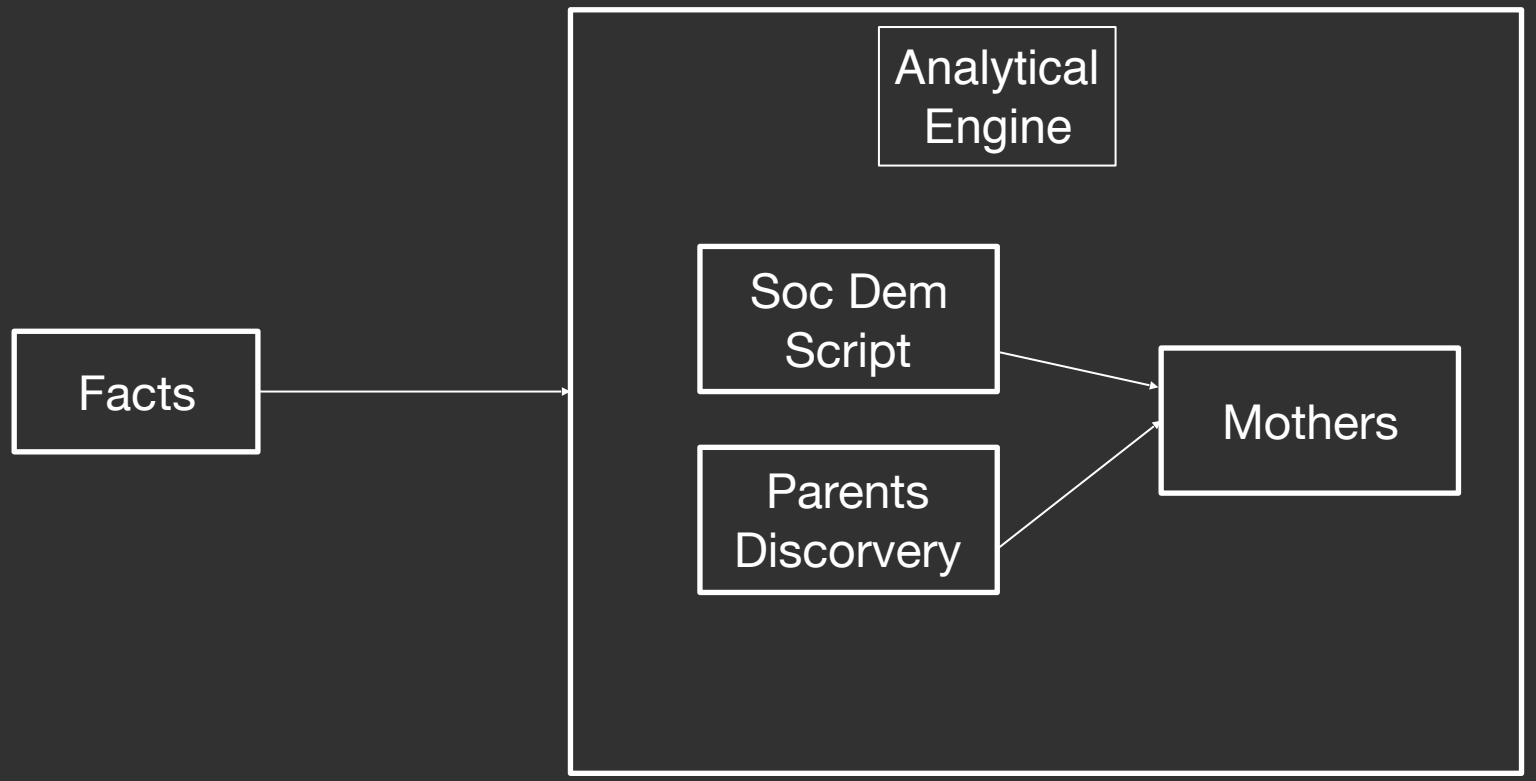
+ Разметка за 1 проход

+ Отсутствие Join-ов

- Невозможность разметки в несколько сегментов в одном скрипте

- Невозможность построения производных сегментов

Challenge #3. Текущий подход



+ Один map-only

+ Нет Join'ов

+ Производные сегменты любой сложности

+ Можно размечать несколько сегментов из одного скрипта

- Задержка между фактом и разметкой в сегмент.

Challenge #4. Real Time Storage

- Нужно хранить информацию о характеристиках сотен миллионов пользователей интернета
- Нагрузка на чтение – десятки тысяч запросов в секунду
- Необходимое время ответа < 15ms
- Полное обновление раз в сутки -> ~10000 в секунду на запись

Challenge #4. Real Time Storage: MongoDB

+ Шардинг из коробки
+ Репликация из коробки
+ Простота использования в клиенте, хорошая документация

- Сложность администрирования шардированного кластера
- Деградация скорости при одновременном чтении и записи
- Деградация скорости в случае когда объем хранимых данных превышает объем оперативной памяти

Challenge #4. Real Time Storage: Aerospike

+ Шардинг и репликация из коробки
+ Простота в администрировании
+ Очень быстрый, даже в случае когда данные не влазят в память (заточен на работу с SSD)

- Не очень хорошо работающие клиенты под разные языки программирования
- Не самая лучшая документация

Challenge #5. BI

- Задача - уметь быстро анализировать логи рекламных кампаний и отвечать на вопросы на вроде “Сколько кликов был вчера по рекламной кампании Mazda в разбивке по регионам”
- Десятки измерений
- Десятки показателей эффективности
- Желание агрегировать и фильтровать данные по произвольному срезу

Challenge #5. BI: Предагрегаты на MapReduce

- Заранее понимаем какие срезы нам потребуются
- Рассчитываем на MapReduce.

+ достаточно простой и дешевый способ
+ отображение сформированного отчета за доли секунды

- Необходимость заранее определить необходимые срезы
- Комбинаторный взрыв количества отчетов
- Задержка между попаданием записи в лог и ее появлением в статистике

Challenge #5. BI Google Big Query

- Псевдо-SQL база данных
- Произвольные срезы и агрегации на полных данных

- Оплата за отпроцешенные терабайты
- - Дорого

Challenge #5.BI Impala

- Псевдо-SQL, процессинг данных в памяти
- Opensource, оплата только за железо
- Интерактивность
  - Процессинг сотен миллионов строк занимает несколько секунд

Challenge #6. Сбор тематических сегментов

- Задача: собирать сегменты по интересам пользователей интернета
- Древовидные категории
- Интересы в первую очередь на основе URL.

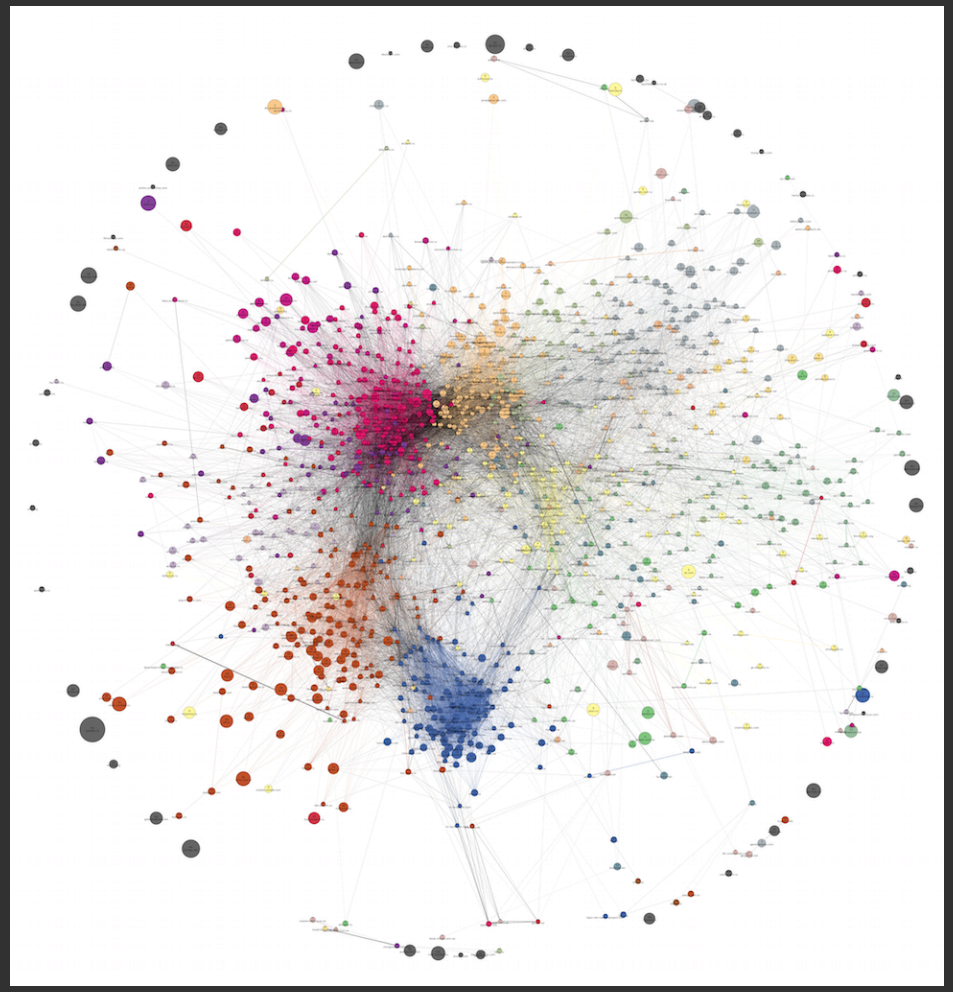
Challenge #6. Ручная сборка

+ Простота, остсутствие формализации

- Скорость работы

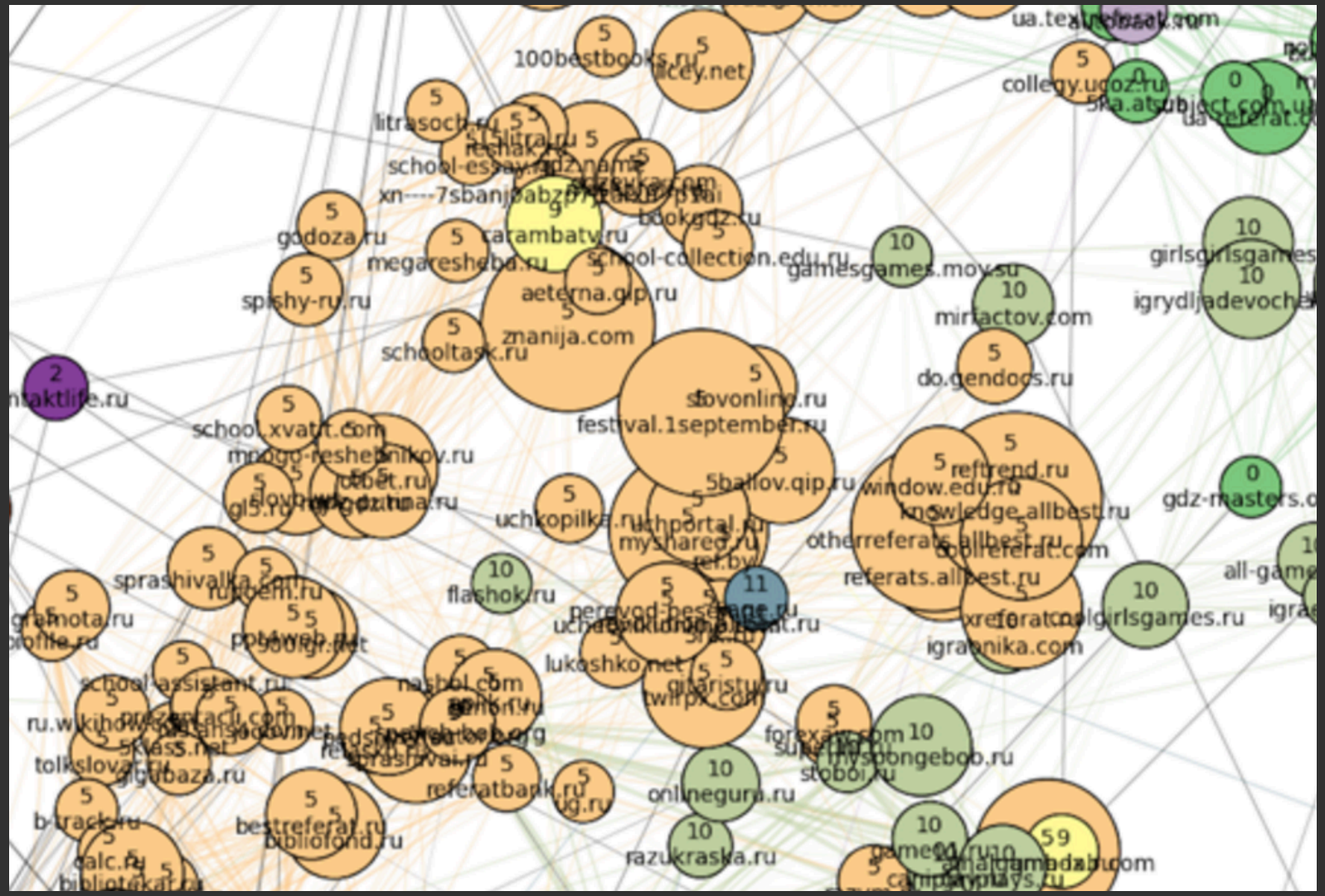
- Удобство

Challenge #6. Кластеризация доменов

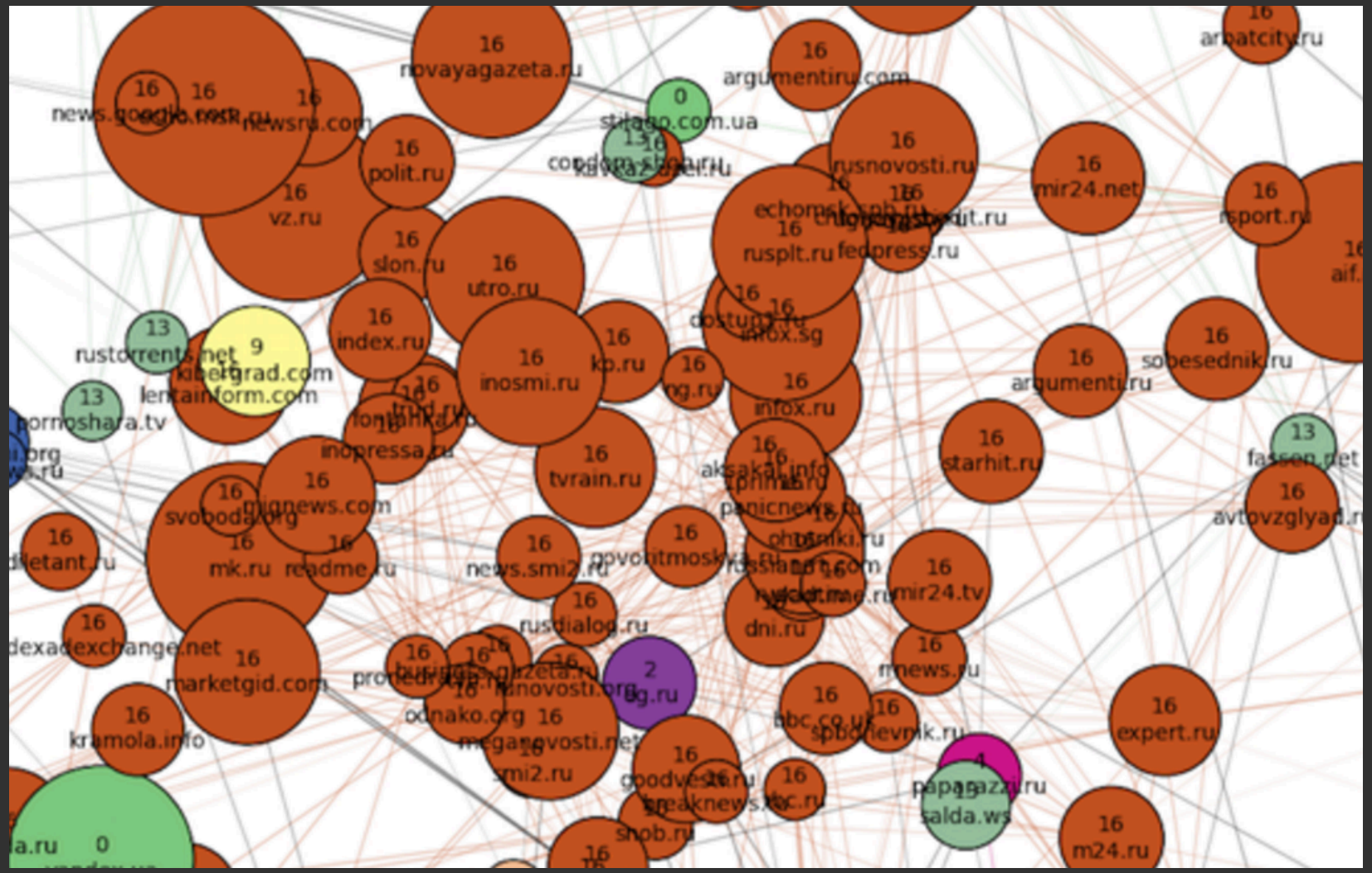


- Строим граф из всех сайтов в интернете
- Если 2 сайта посещают чаще чем их по отдельности – добавляем ребро
- Находим области графа, где вершины более плотно объеденены ребрами
- Даем области адекватное имя

Challenge #6. Кластеризация доменов



Challenge #6. Кластеризация доменов



Challenge #6. Семантический анализ URL.

- Внешнее стороннее решение
  + “Поставил и работает”
  + Простой API
  + Готовое дерево категорий

  - Дорого
  - Заточенность категорий под другие задачи
  - Невозможность влиять на список категорий

Challenge #6. Семантический анализ URL. Собственное решение

- Каждая категория описывается несколькими поисковыми запросами
- Обучающая выборка формируется как выдача поисковика по этим запросам
- Multi-Label классификатор в каждом узле дерева

+ Возможность формирования собственного дерева категорий
+ Простота в использовании
+ Бесплатное использование

Challenge #7. Machine Learning

- Задача 1: По имеющейся обучающейся выборке восстанавливать характеристики пользователей.
  - Пол, возраст, интересы.

- Задача 2: По имеющемуся классу построить группу “похожих” пользователей – пользователей которые ведут себя похожим образом в интернете. “look-a-like”

Challenge #7. Machine Learning: Weka

- Библиотека машинного обучения для Java

+ Удобно интегрируется в hadoop-экосистему

- Не очень быстрая
- Нету таких удобных средств отладки моделей как Ipython, sklearn, pandas, matplotlib

- Аналитики отлаживают модели в python, а потом переводят на weka.

Challenge #7. Machine Learning: Python Daemon

- На каждой ноде поднимаем сервер содержащий python-модели
- В mapper-е кидаем объект User в сервер и получаем в ответ список сегментов
- Можно использовать python-модели

Спасибо за внимание!