Платформа инкрементальных вычислений на базе MapReduce

Федор Бочаров

Научный руководитель: Александр Юрьевич Юрченко, Яндекс

САНКТ-ПЕТЕРБУРГСКИЙ АКАДЕМИЧЕСКИЙ УНИВЕРСИТЕТ

13 июня 2017 г.

MapReduce[1] – парадигма программирования, которая позволяет удобно обрабатывать большие объемы данных.

Идея: отображаем данные в пары <ключ, значение> (map), затем аггрегируем значения с одинаковыми ключами (reduce).
Особенности:

- Позволяет обрабатывать произвольно большие объемы данных
- Обладает большой латентностью

[1] http://bit.ly/2sfzbO8

YT[2] – платформа распределенных вычислений, поддерживающая парадигму MapReduce. Разработанна в компании Яндекс.

Обеспечивает:

- Надежное выполнение операций на кластере
- Надежное хранение танных в *таблицах*
- Потабличные транзакции

[2] http://bit.ly/2raQKvs

Мотивация

Типичный MR-процесс при обновлениях обрабатывает весь объем данных, но со временем новые данные начинают поступать быстрее, чем успевают обрабатываться.

Вопрос: Как обрабатывать данные быстрее?
Ответ: Обрабатывать их *инкрементально*, т.е. не весь объем, а только изменившуюся часть.
Эта идея описана в статье о системе Percolator[3].

[3] http://bit.ly/2rSE3YL

Идея: пользователь реализует *триггеры* (функции-обработчики), которые работают с большой таблицей, и запускаются только при обновлении данных.

Особенности:

- Данные хранятся в KV-базе[4] и адресуются кортежем <ключ, колонка, время>
- Пользовательские триггеры могут транзакционно читать/писать любые наборы строк
- После записи новых данных в колонку срабатывают триггеры, которые «подписаны» на нее

[4] http://bit.ly/2raH9ow

Цель: сделать платформу, реализующую модель вычислений Percolator в модели MapReduce.

Задачи:

1. Разобраться с Percolator и понять, как реализовать его в MapReduce
2. Реализовать и протестировать платформу
3. Внедрить платформу в текущие бизнес-процессы и обеспечить ее поддержку

Цель

Почему MapReduce

1. На MR-кластерах хранится большинство данных компании, а значит работать с ними лучше на этих же кластерах
2. Хранение данных в общем MR-кластере дает возможность обрабатывать их другими MR-приложениями
3. YT – надежная протестированная система, использование которой позволит сэкономить большое количество ресурсов

Анализ Percolator

Основные проблемы, возникающие при реализации модели Percolator на MapReduce:

- Отсутствие построчных транзакций.
- Операция работает только с данными одного ключа.

Разработанная платформа

Модель вычислений

- Логически данные представляют из себя таблицу
- Триггеры выполняются в транзакциях с уровнем изоляции snapshot[5] и работают с произвольными ключами базы
- Запуск exactly once на новых данных

[5] http://bit.ly/1SSxglD

Разработанная платформа

Физическое устройство

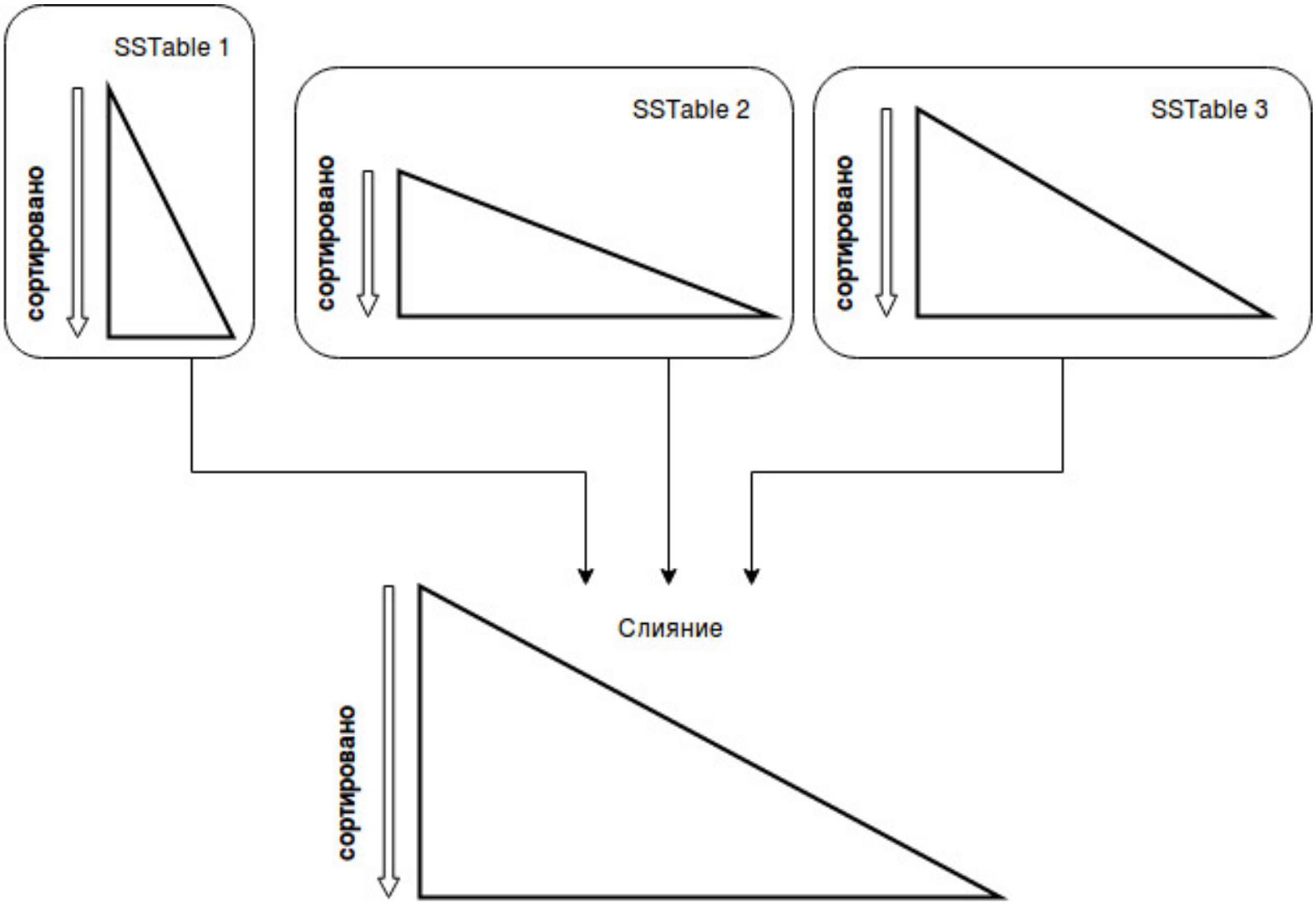
- Физически база представляет из себя LSM-дерево[6], каждая SSTable которого – таблица на YT.
- Операций с базой – Reduce таблиц с данными и некоторых вспомогательных таблиц

Например: для запуска триггеров только на новых данных достаточно сделать Reduce таблиц с информацией о том, какие колонки изменились, и таблиц базы.

[6] http://bit.ly/2rnGVJV

Физическое устройство

LSM-дерево



Свой ключ – ключ, который передан в Reduce-операцию, запускающую триггер.

Чужие ключи – все остальные ключи в базе.

Предположение: обычно, триггеры читают небольшое количество чужих ключей, поэтому их чтение можно реализовать с помощью *зеркалирования* данных.

Разработанная платформа

Чтение чужих ключей

- Каждый ключ хранит «кэш» чужих ключей – дополнительная колонка в таблице
- При попытке прочитать ключ, который отсутствует в кэше, генерируется «запрос на чтение» – запись во вспомогательную таблицу
- Перед запуском триггеров происходит разрешение запросов на чтение
- При обновлении данных по ключу *Key*, обновление *зеркалируется* в кэши всех ключей, которые читают *Key*

Разработанная платформа

Пример триггера

```
def trigger(url, ctx):
    content = ctx.get_bytes(url, "content")
    rank = 0
    for link in extract_links(content):
        rank += ctx.get_int(link, "rank")
    ctx.set_int(url, "rank", rank)
```

Разработанная платформа

Что еще есть

- Типизированный доступ к данным из триггеров
- Импорт/экспорт данных с произвольными форматами
- Автоматический конфигурируемый compaction и сборка мусора
- Мониторинг системы (сколько времени исполнялись триггеры, сколько использовали памяти, ...)

Заключение

Результаты

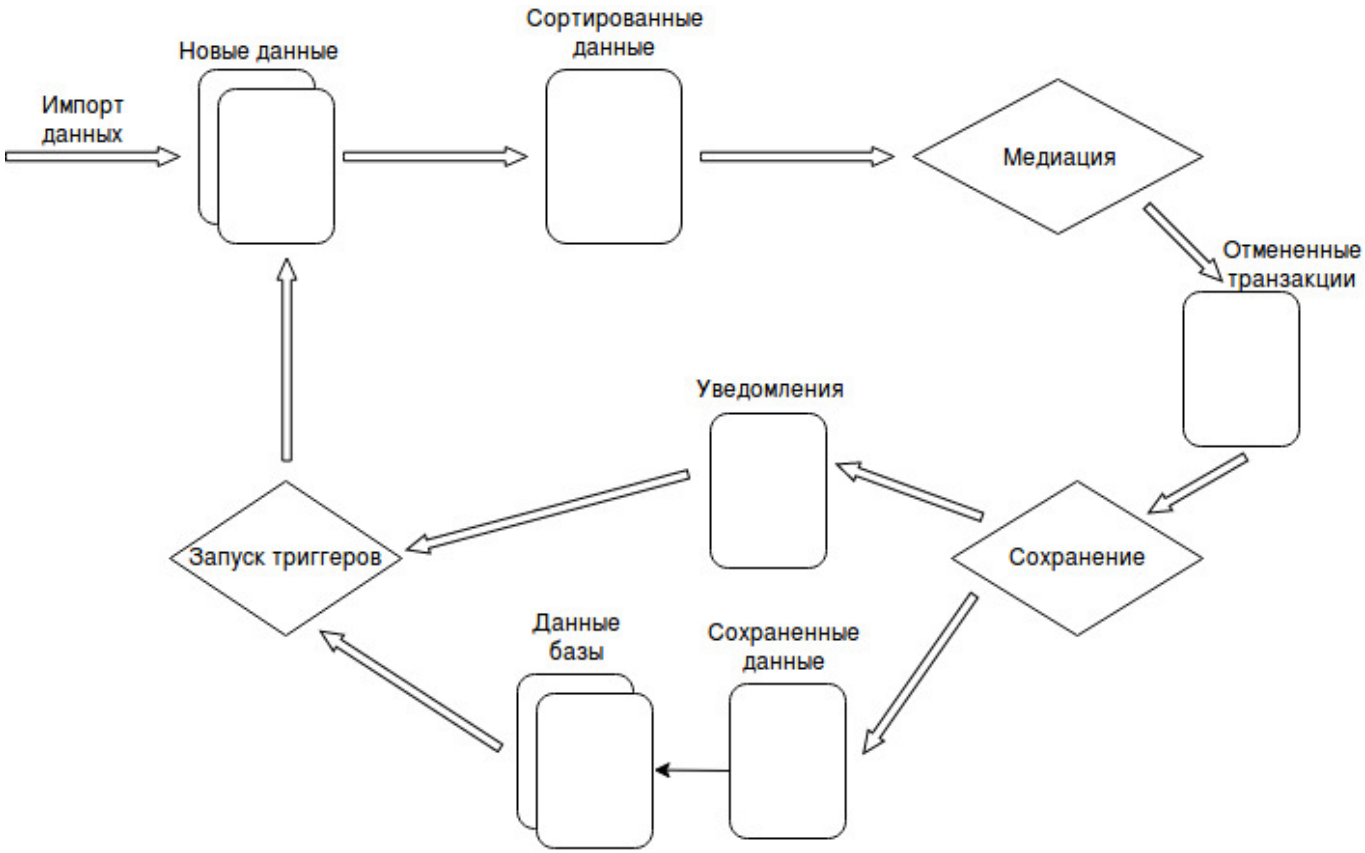
- Выявлены особенности модели Percolator в ограничениях модели MapReduce
- Платформа реализована и протестированная на MapReduce
- Внедрена в несколько production процессов

Заключение

Спасибо за внимание!

Заключение

Общий pipeline



Заключение

Запуск триггеров