Система для распределенной обработки данных с помощью корутин

Казначеев Д. В.

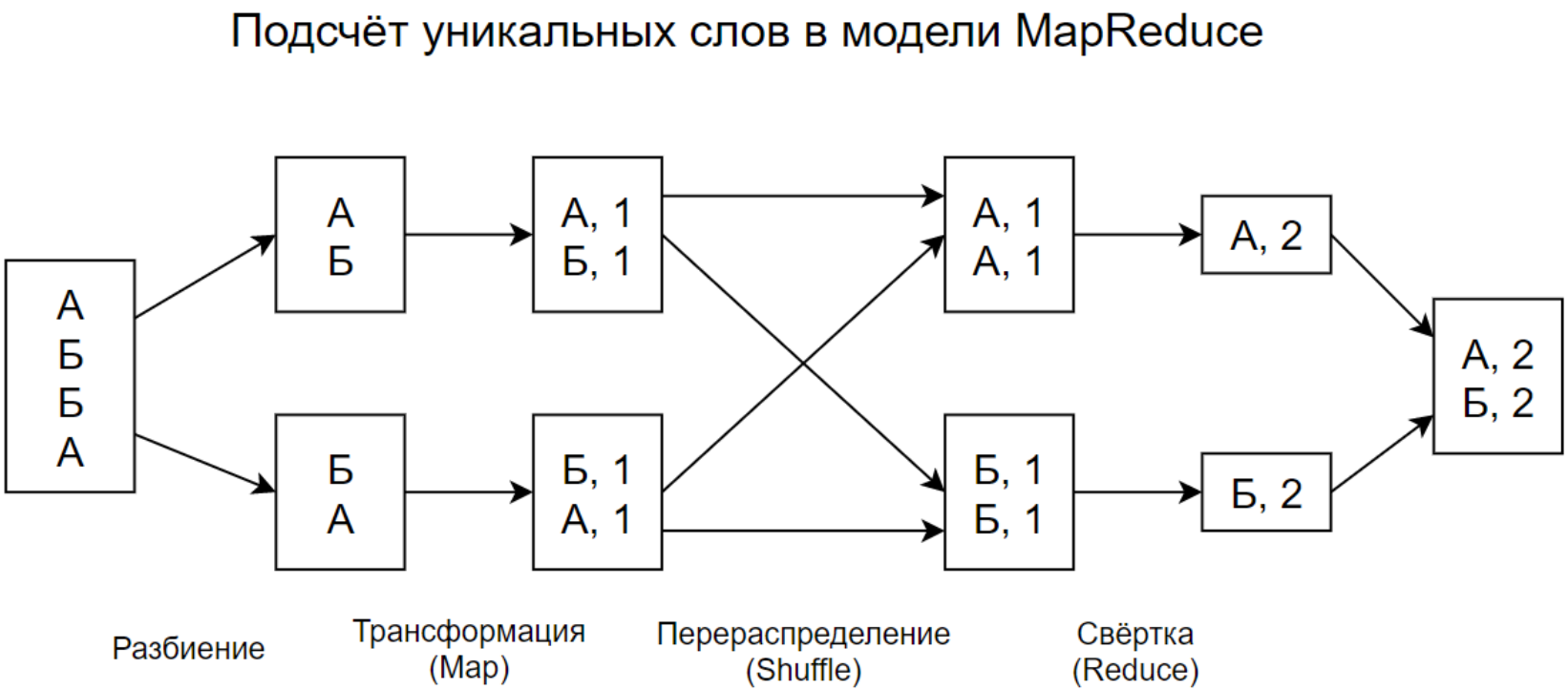
Научный руководитель: к.ф.-м.н. И. А. Близнец

Научный консультант: к.т.н. В. А. Худобахшов

Санкт-Петербургская школа физико-математических и компьютерных наук

НИУ ВШЭ – Санкт-Петербург, 2020

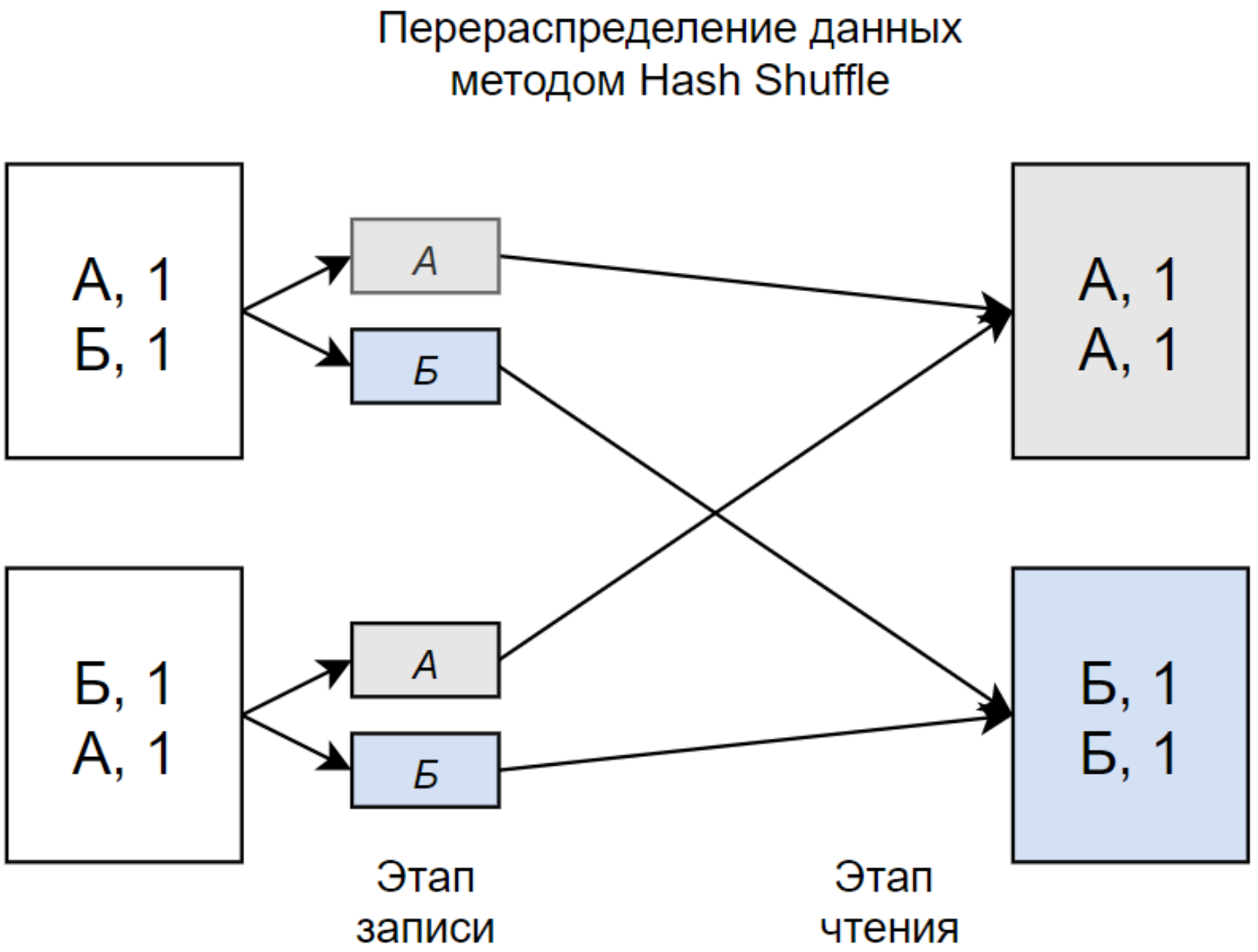
MapReduce



Map: Word -> (Word, 1)

Reduce: (Word, a), (Word, b) -> (Word, a + b)

Перераспределение



Возможные улучшения

Функции в map - блокирующие

- Пока не трансформировали один элемент, не можем приступить к следующим
- Пропустить набор данных через сторонний веб-сервис - нестандартная задача

Перераспределение может быть улучшено с помощью неблокирующих операций

- Пока ожидаем очередной блок данных, можем обрабатывать уже полученные

Обзор работ

Spark REST DataSource[1]

- Запросы блокирующие

Распределённое веб-извлечение данных[2]

- Рассчитаны на извлечение, а не обработку данных
- Сложнее встроить в аналитику

[1] Sourav Mazumder. “Using Apache Spark as a parallel processing framework for accessing REST based data services”. In: (Nov. 2017).

[2] Alexander Sibiryakov. “Distributed Frontera: Web Crawling At Scale”. In: (Aug. 2015).

Обзор работ

Оптимизация производительности shuffle[3]

- Консолидация (reduce на map-стороне перед отправкой)
- Смена файловой системы
  - ext4 быстрее работает с консолидацией, ext3 - без консолидации
- Сжатие

[3] Aaron Davidson and Andrew Or. “Optimizing shuffle performance in Spark”. In: (2013).

Корутины

Удобный способ управлять многозадачностью

- Легковесные потоки
- Не требуют переключения контекста процессора
- Удобный интерфейс для неблокирующих операций

Корутины в Kotlin

Основная абстракция - suspending функции

- Способны останавливать выполнение, не блокируя свой поток
- Могут быть вызваны только внутри корутины
- Позволяют писать асинхронный код в синхронном стиле

```
longFunction(a, onComplete = { b ->
    longFunction(b, onComplete = { c ->
        println(c)
    })
})
```

```
launch {
    val b = longFunction(a)
    val c = longFunction(b)
    println(c)
}
```

Цель и задачи

Цель: Исследовать возможность использования корутин для создания системы распределённой обработки данных на основе MapReduce

Задачи:

- Спроектировать систему, использующую корутины в потоковой обработке данных
- С помощью корутин поддержать неблокирующие операции в map
- Реализовать этап перераспределения (shuffle), используя корутины
- Создать прототип системы и измерить его производительность

Архитектура системы

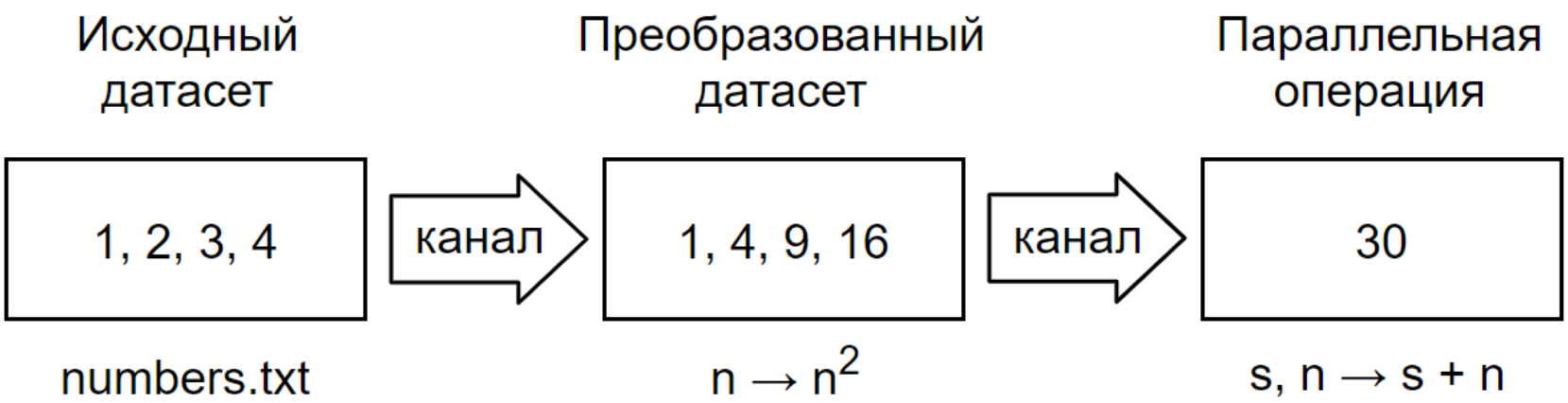
Абстракция распределённого датасета взята из Apache Spark
RDD - Resilient Distributed Dataset:

- getPartitions() - разделы датасета
- getPreferredLocations(partition) - узлы, на которых лучше обрабатывать данный раздел
- getIterator(partition) - итератор для последовательного получения элементов

Каждый RDD ссылается на предыдущий или является источником

Архитектура системы

Блокирующий итератор заменён на неблокирующий канал



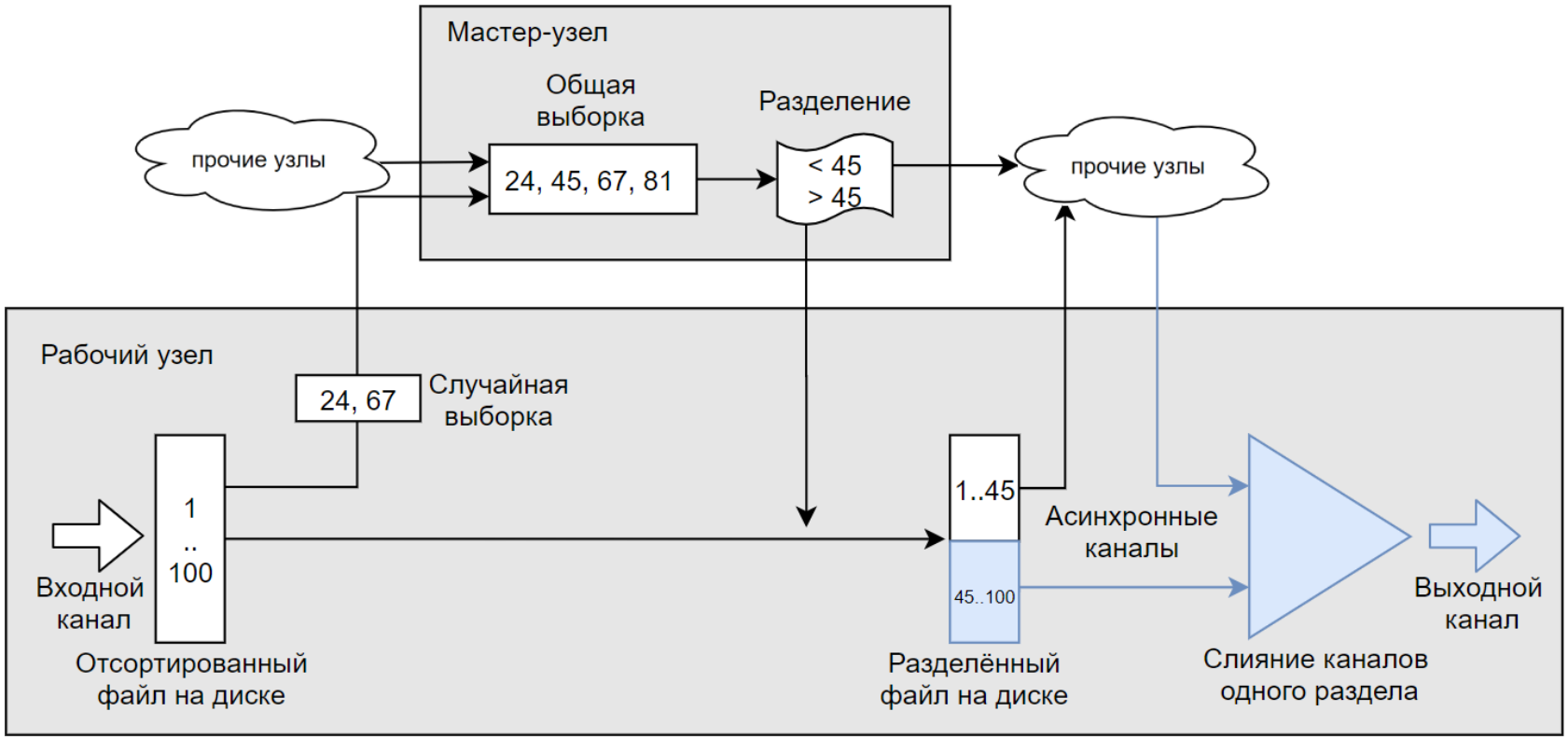
Асинхронный map

Цель: на каждом узле трансформировать suspending функцией f не более N элементов датасета одновременно

Алгоритм:

- Получить от родителя канал на чтение R
- Открыть канал на запись W
- Запустить N корутин:
  - Пока R не пуст:
    - Получить элемент e из R
    - Дождаться выполнения $f(e)$
    - Отправить результата $f(e)$ в W
- Дождаться завершения всех корутин
- Закрыть канал W

Shuffle


Мастер-узел
Общая выборка
Разделение
прочие узлы
24, 45, 67, 81
< 45
> 45
прочие узлы
Рабочий узел
24, 67
Случайная выборка
1
..
100
Входной канал
Отсортированный файл на диске
1..45
45..100
Асинхронные каналы
Разделённый файл на диске
Слияние каналов одного раздела
Выходной канал

Прототип

Режимы:

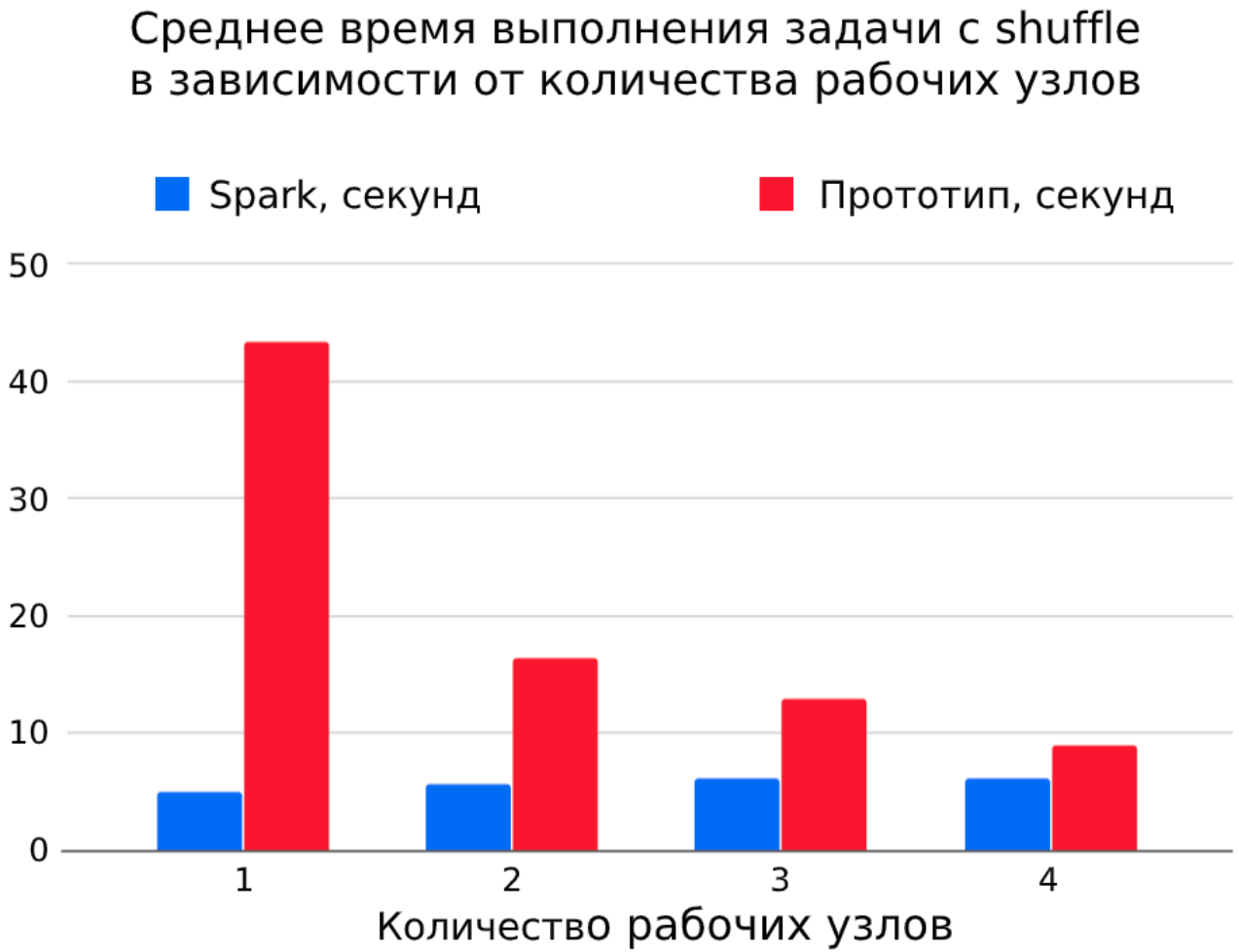
- Локальный
- Распределённый

```
val master = LocalMaster()
val translator = TranslatorService( address: "localhost:8080")

fileRdd<String>(master,  filename: "words.txt")
        .map { word -> translator.translate(word) }
        .saveAsText( filename: "translated.txt")
```

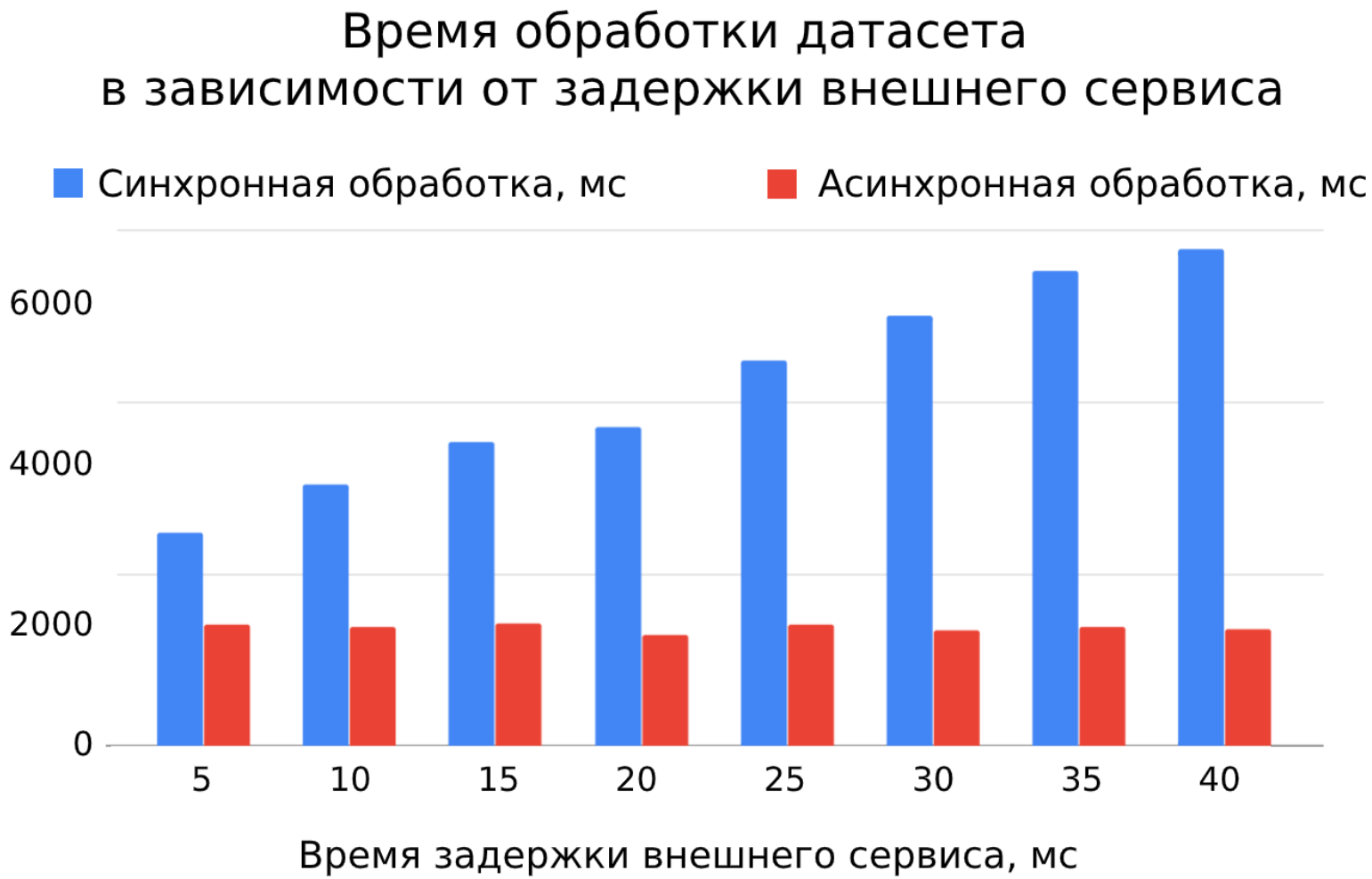
Сравнение с Apache Spark в скорости shuffle

Поиск самого частого из 1 млн случайных чисел
1-4 EC2 m5.xlarge инстансов, 4 ядра, 16 GB RAM



Тестирование с внешним сервисом

Отправка 200 запросов к внешнему сервису, отвечающему с задержкой 5-40 мс



Результаты

- Разработан прототип MapReduce-подобной системы обработки данных на основе корутин.
- В системе реализована трансформация коллекций неблокирующими функциями
- Реализован алгоритм shuffle на основе асинхронных каналов
- Разработана среда для интерактивного использования системы
- Измерена производительность прототипа в сравнении с Apache Spark на кластере с 4 рабочими узлами

Вывод: Использование корутин позволило ускорить параллельное выполнение неблокирующих операций, но не дало прироста в производительности shuffle.