Курсовой проект
Фреймворк и файловая система для распределённой обработки больших данных в рамках концепции map-reduce

Выборнов А.И.

МГТУ им. Н. Э. Баумана

art-vybor@ya.com

26 декабря 2014 г.

Обзор

Постановка задачи

Концепция map-reduce

Архитектура

Тестирование

Постановка задачи

- Анализ требований и проектирование архитектуры системы. Реализация нераспределённого map-reduce. Решение проблемы RPC.
- Реализация распределённой файловой системы. Реализация фреймворка. Тестирование на примерах.

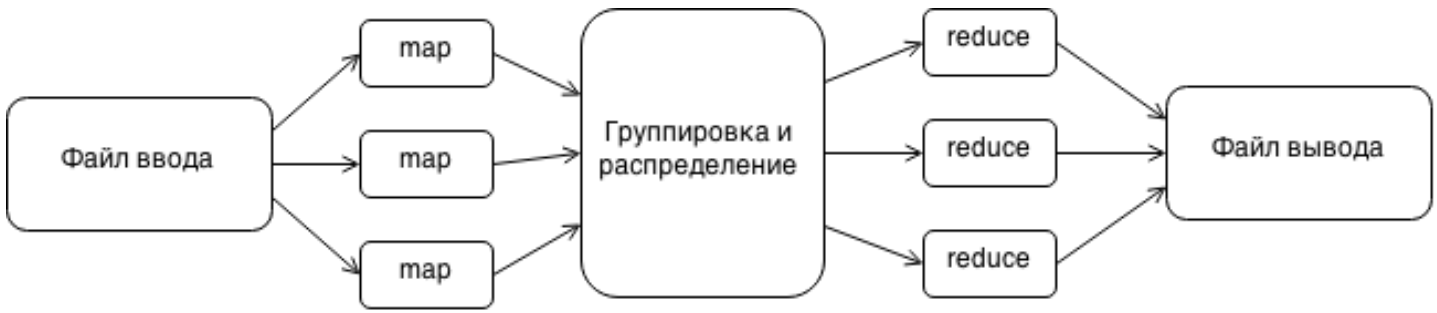
Зачем нужен map-reduce?

- Вычисления превосходят возможности одной машины.
- Данные не помещаются в памяти, необходимо обращаться к диску.
- Большое количество узлов в кластере вызывает множество отказов.
- Данные хранятся на множестве машин.
- Достаточно дорогая и сложная разработка низкоуровневных приложений для подобных систем.

Что такое map-reduce?

- Структура (key, value) - пара (ключ, значение).
- Программирование представляет собой определение двух функций:
  - $map : (key, value) \rightarrow [(key, value)]$
  - $reduce : (key, [value]) \rightarrow [(key, value)]$
- Между стадиями *map* и *reduce* происходит группировка и сортировка данных.

Что такое map-reduce?



Map-reduce на примере - Поиск общих друзей

- **Задача:** Есть граф пользователей некоторого ресурса, заданный в виде строчек: «пользователь - друг1 друг2 ...». Для каждой пары пользователей найти общих друзей.
- Разберём задачу на следующих входных данных:
  - A - B C D
  - B - A C
  - C - A B D
  - D - A C

Map-reduce на примере - Поиск общих друзей

- На стадии map преобразовываем пару (пользователь, друзья) в множество пар следующим образом:
  - (A, B C D) -> (A B, B C D), (A C, B C D), (A D, B C D)
  - (B, A C) -> (A B, A C), (B C, A C)
  - (C, A B D) -> (A C, A B D), (B C, A B D), (C D, A B D)
  - (D, A C) -> (A D, A C), (C D, A C)

Map-reduce на примере - Поиск общих друзей

- Сливаем результаты полученные на стадии map, получаем список пар:
  - (A B, [B C D, A C])
  - (A C, [B C D, A B D])
  - (A D, [B C D, A C])
  - (B C, [A B D, A C])
  - (C D, [A B D, A C])

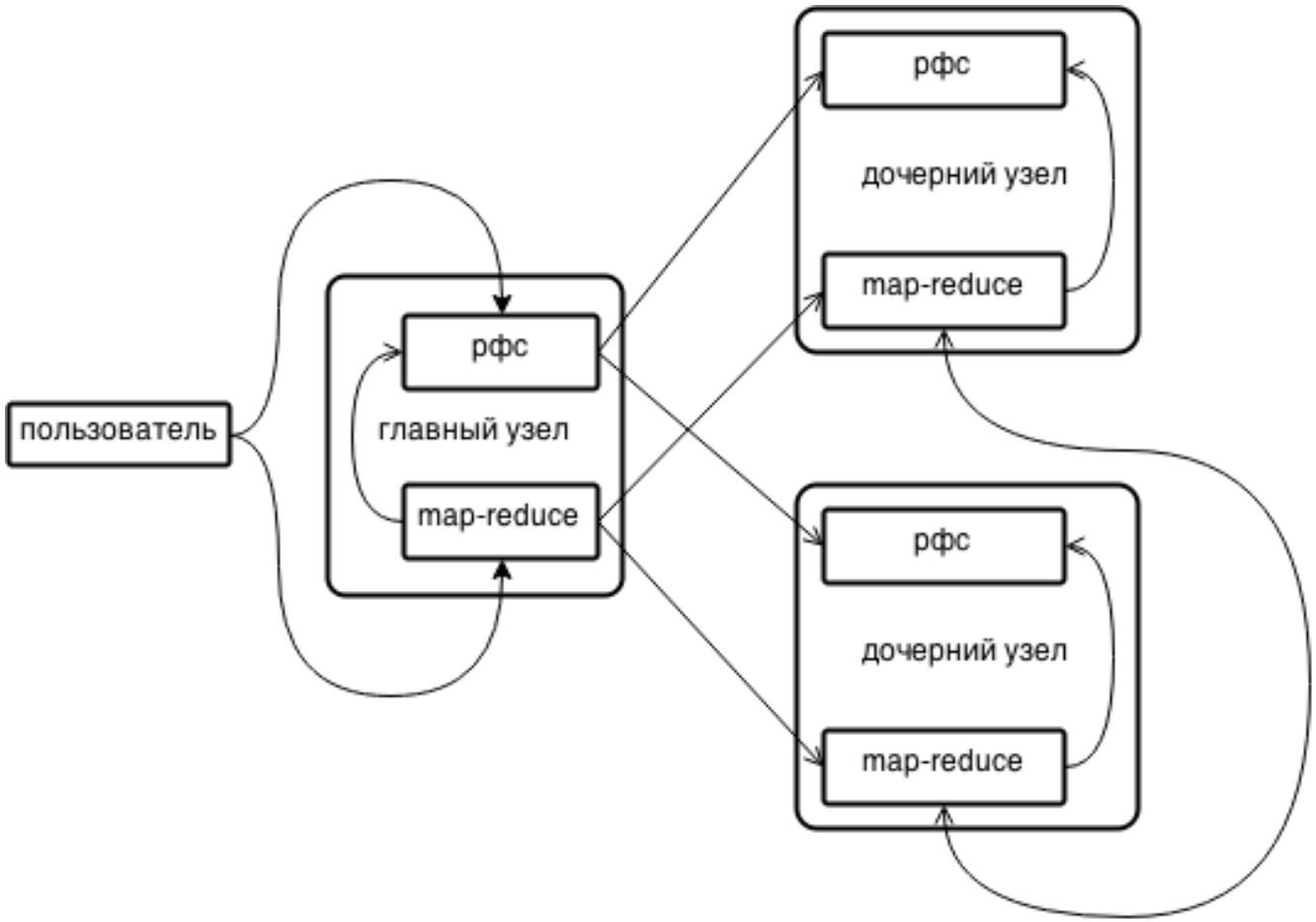
Map-reduce на примере - Поиск общих друзей

- На стадии reduce пересекаем друг с другом все элементы списка значений и получаем:
  - (A B, C)
  - (A C, B D)
  - (A D, C)
  - (B C, A)
  - (C D, A)

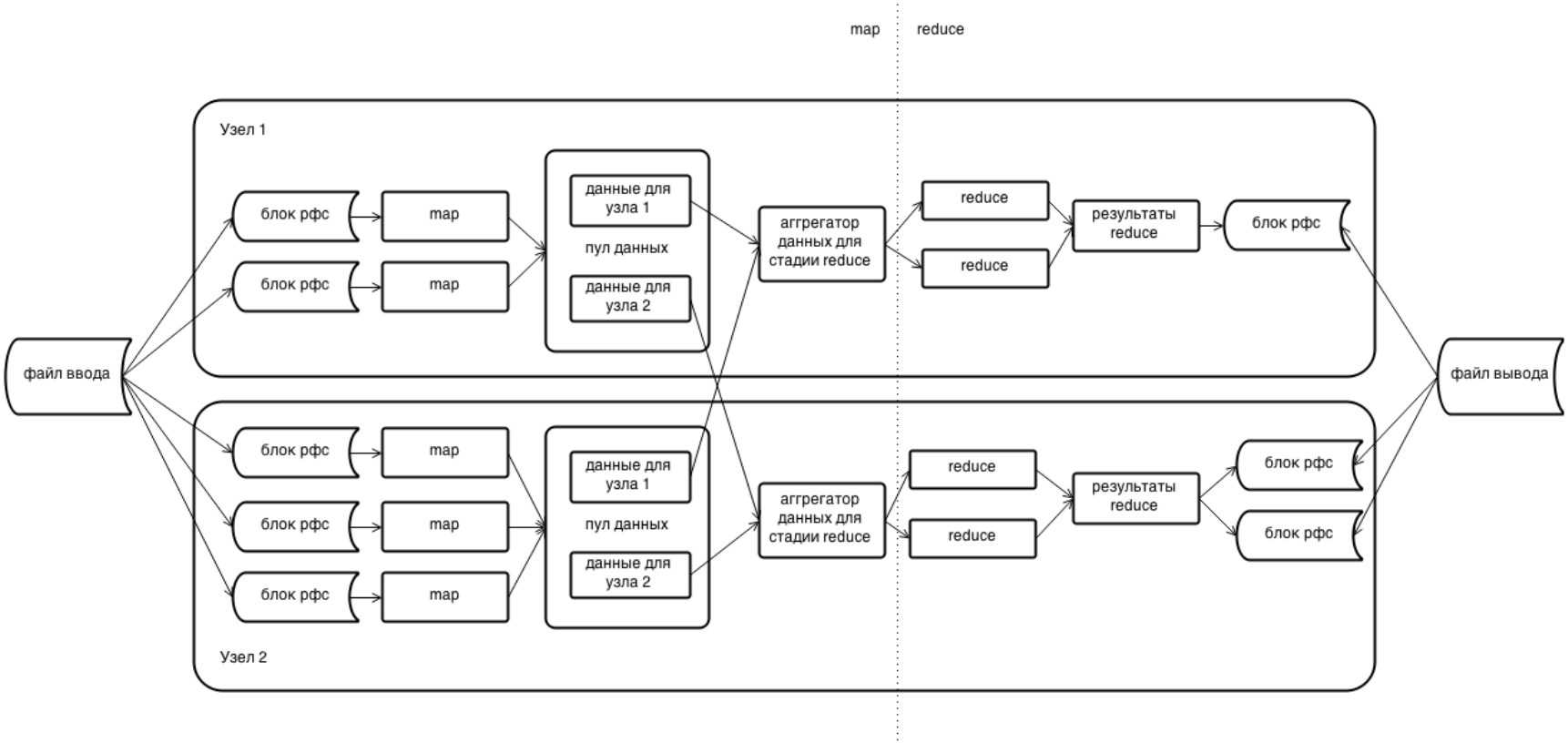
Map-reduce на примере - Поиск общих друзей

```
def map_func(string):
    [person, friends] = string.split('␣–␣')
    friends = friends.split()

    for friend in friends:
        yield ('%s␣%s:␣' % tuple(sorted([person, friend])), friends)

def reduce_func(key, values):
    result = ''
    if len(values) == 2:
        result = '␣'.join(set(values[0]).intersection(set(values[1])))
    return [(key, result)]
```

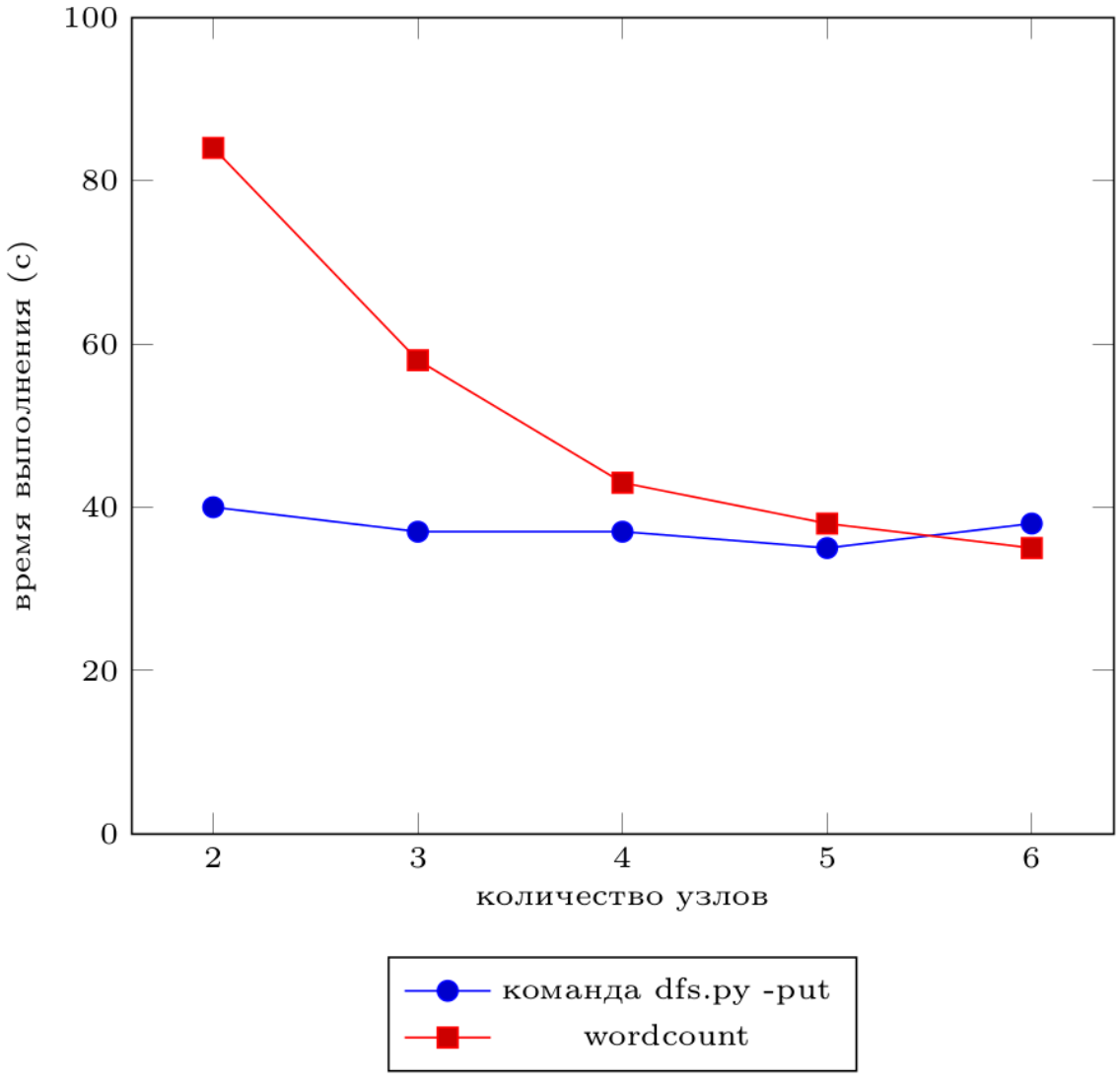
Архитектура распределённого map-reduce



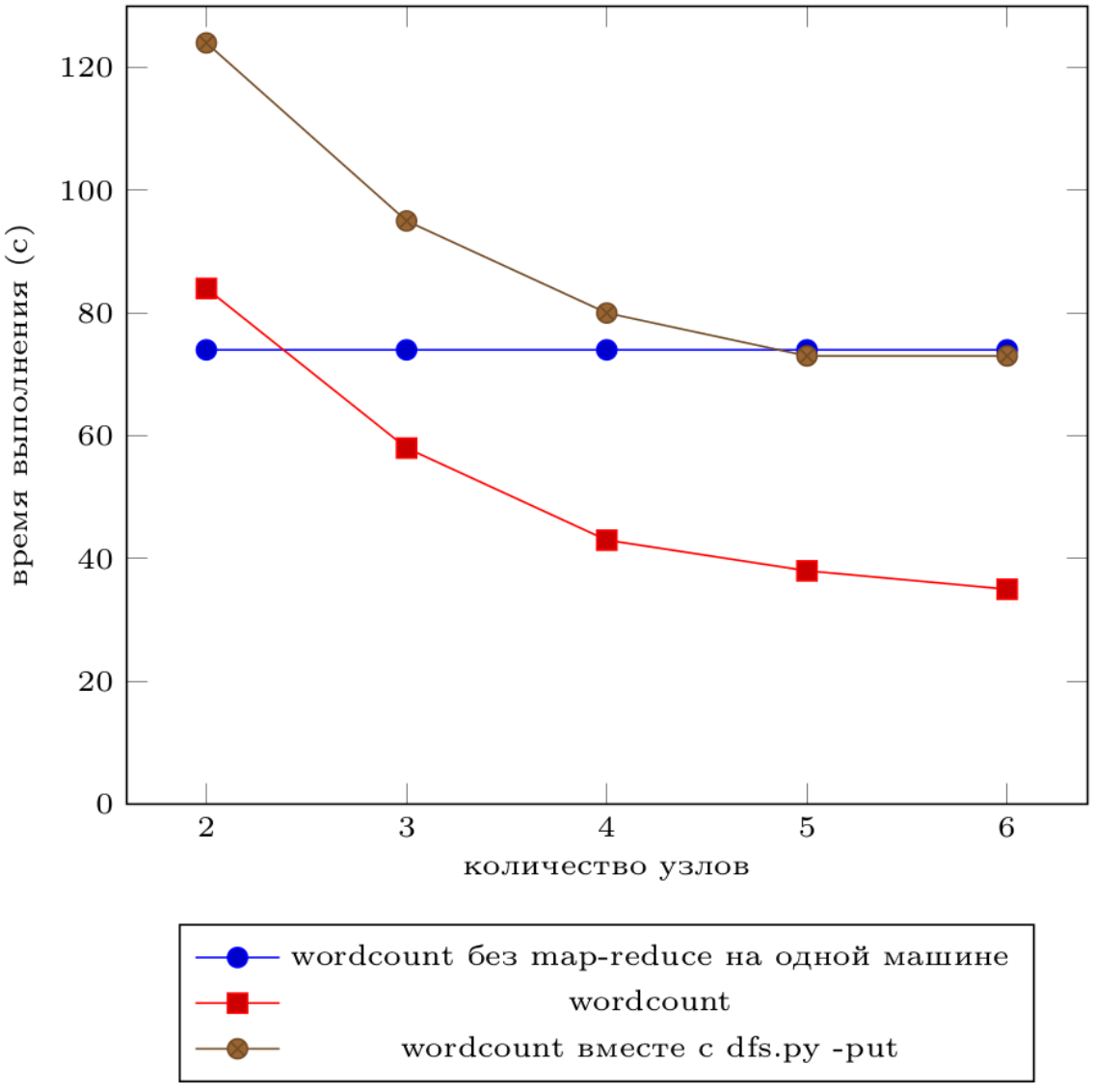
Архитектура распределённого map-reduce



Результаты тестирования



Результаты тестирования



Результаты тестирования

- Выполнение задачи класса информационный поиск:
  - время загрузки данных в РФС — 4мин. 30с.
  - время выполнения задачи с помощью фреймворка map-reduce — 1мин. 54с.
  - время выполнения задачи на одной машине без использования фреймворка — 6мин. 39с.

Выводы

- С увеличением количества узлов производительность выполнения задач с помощью фреймворка растёт и превосходит производительность выполнения на одной машине.
- Время загрузки данных в РФС не зависит от количества узлов.
- Наиболее подходящей стратегий использования фреймворка является загрузка данных в РФС и их последующее использование в качестве входных данных для нескольких задач.
- Map-reduce удобная абстракция для построения решения части задач класса информационный поиск.
- Использование фреймфорка map-reduce для решения задач класса информационный поиск целесообразно.

Курсовой проект
Фреймворк и файловая система для распределённой обработки больших данных в рамках концепции map-reduce

Выборнов А.И.

МГТУ им. Н. Э. Баумана

art-vybor@ya.com

26 декабря 2014 г.