Методы и системы обработки больших данных

11. Модель параллельной обработки больших данных MapReduce

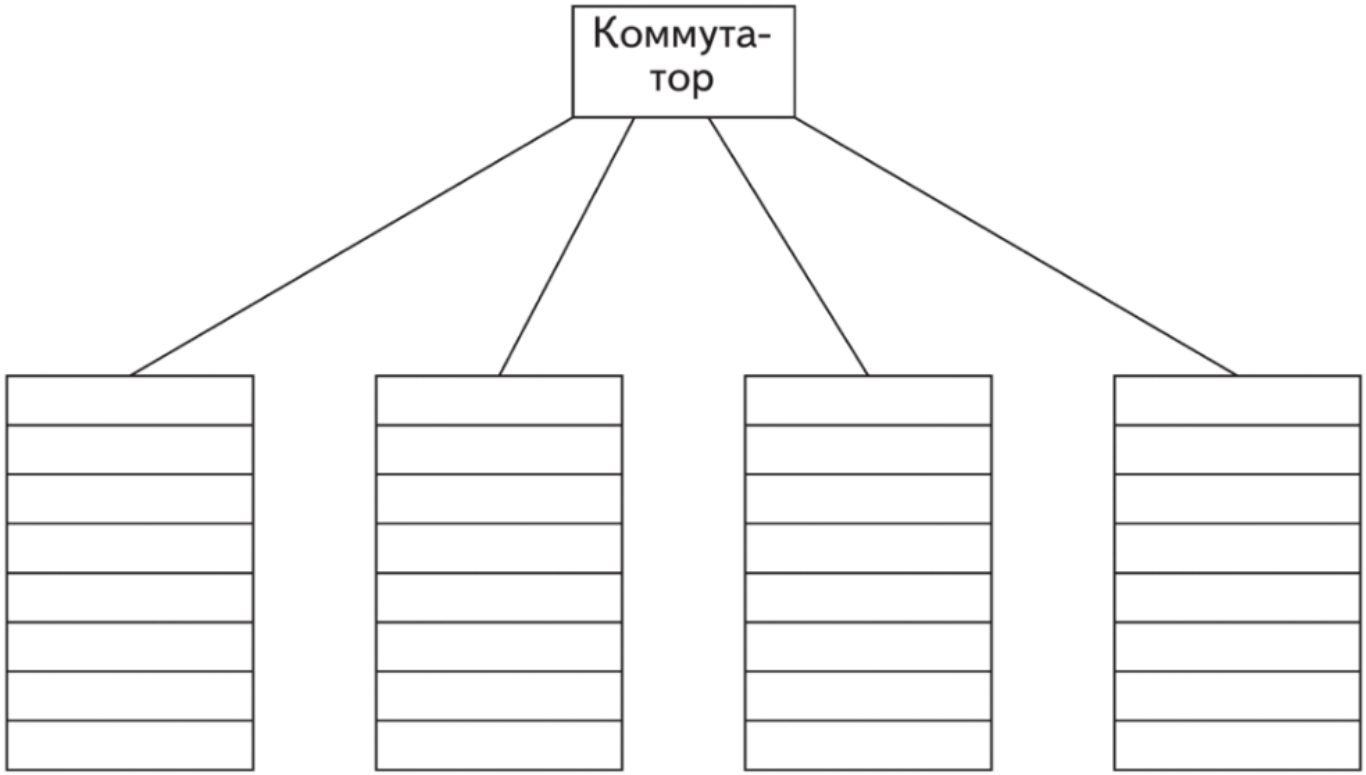
Стек обработки больших данных

MapReduce

Распределенная файловая система

Вычислительный кластер

Вычислительный кластер

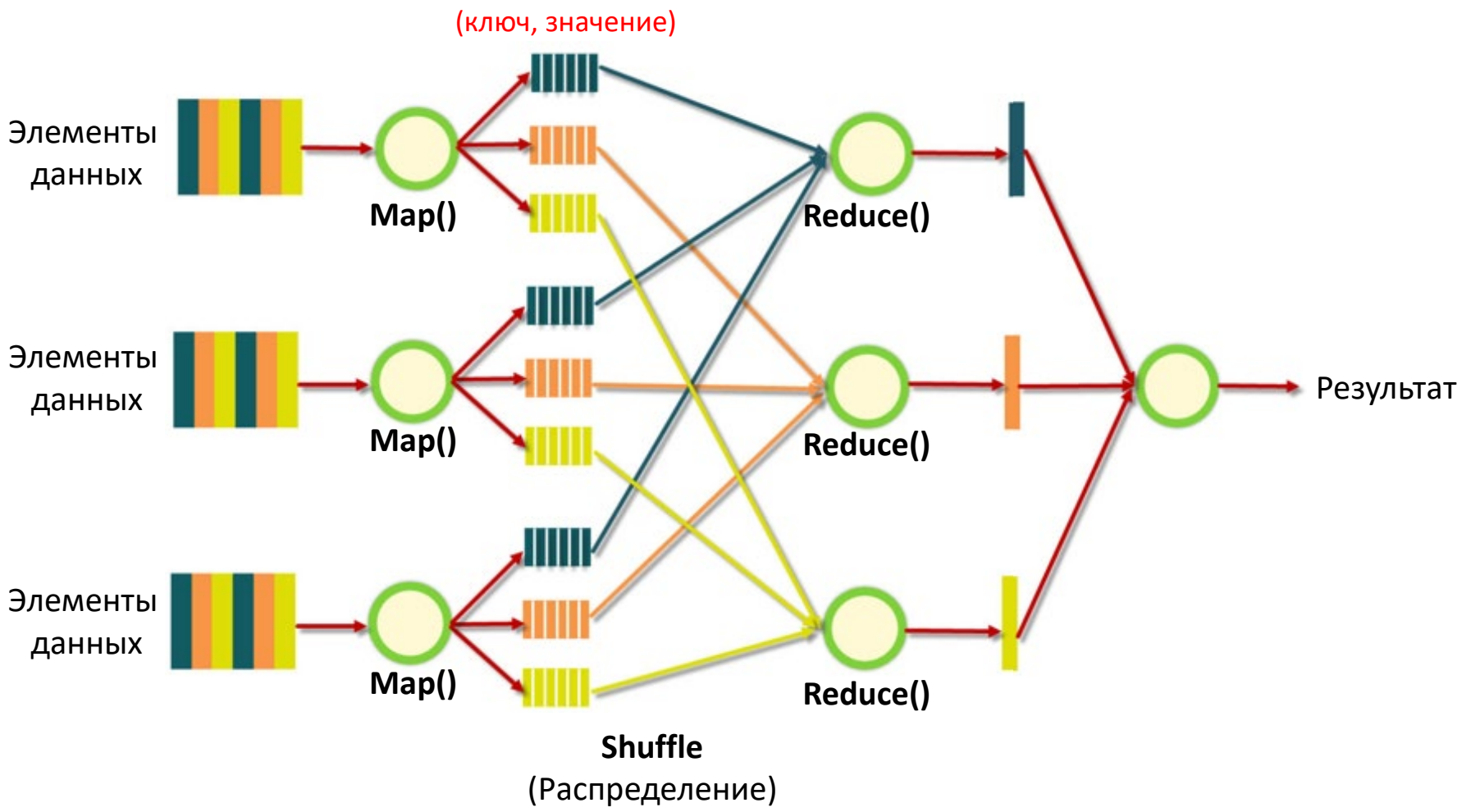


Стойки с вычислительными узлами

Распределенная файловая система

- Предназначение
  - Обработка файлов терабайтного размера (10^{12} байт)
  - Файлы редко обновляются
- Принципы работы
  - Файлы разбиваются на фрагменты размером 64 Мбайт (параметр системы)
  - Каждый фрагмент реплицируется на 3 узла в разных стойках (параметр системы)
  - Информация о всех фрагментах и репликах каждого файлах хранится в специальном файле «Каталог»
  - Каталог также реплицируется
- Реализации
  - GFS: Google File System (2000)
  - Colossus (Google, 2010)
  - HDFS: Hadoop Distributed File System (Apache Software Foundation, 2008)
  - CloudStore (Kosmix, 2007)

MapReduce


(ключ, значение)
Элементы данных
Элементы данных
Элементы данных
Map()
Map()
Map()
Reduce()
Reduce()
Reduce()
Результат
Shuffle
(Распределение)

Проблемно-зависимая функция $\boldsymbol{Map}()$

- R – множество элементов данных (реляционное отношение, множество документов, набор изображений и др.)
- K – множество ключей
- V – множество значений
- $K \times V$ – множество пар <ключ, значение> вида (k, v)
- $[K \times V]$ – множество списков различной длины вида $[(k_1, v_1), \dots, (k_l, v_l)]$, включающее пустой список []
- $Map: R \to [K \times V]$

$$Map(r) = [(k_1, v_1), \dots, (k_l, v_l)]$$

Проблемно-независимая процедура Shuffle

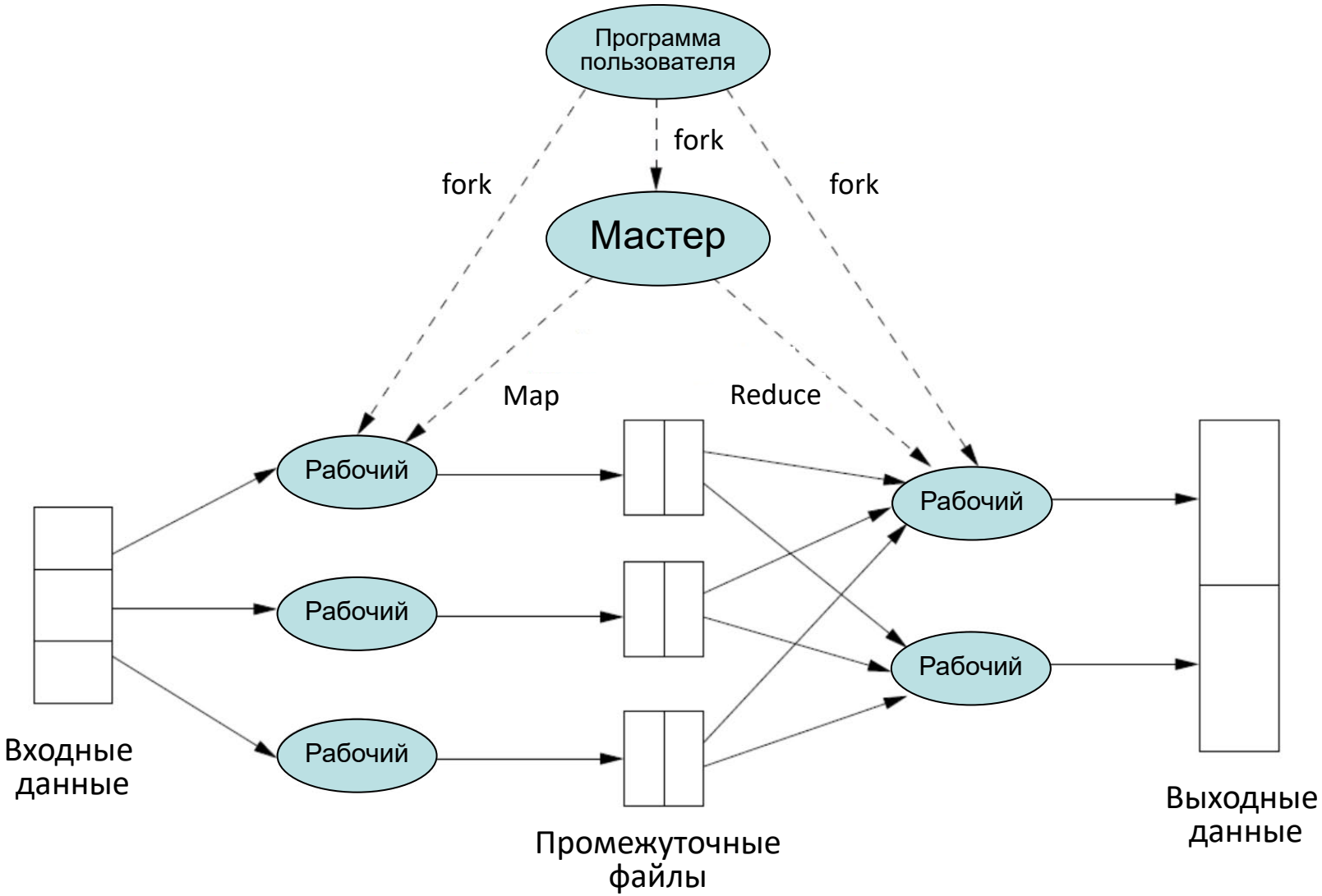
- Все пары $(k, v_1), \dots, (k, v_m)$ с одинаковым значением ключа k преобразуются в одну пару $(k, [v_1, \dots, v_m])$
- Каждая пара $(k, [v_1, \dots, v_m])$ пересылается для обработки на узел $h(k)$,
  где $h: K \to \{0, \dots, p-1\}$ является проблемно-независимой хеш-функцией, вычисляющей номер узла Reduce (может задаваться пользователем)

Проблемно-зависимая функция $\boldsymbol{Reduce}()$

$$Reduce(k, [v_1, \dots, v_m]) = (k, w)$$

$$w = v_1 \oplus \dots \oplus v_m$$

Схема работы MapReduce



Пример с подсчетом слов

- Имеется коллекция документов $R = \{Doc_1.txt, \dots, Doc_m.txt\}$.
  Необходимо подсчитать для каждого слова суммарное количество вхождений во всех документах коллекции (без учета словоформ)
- Множество ключей: $K = \{\text{слово}_1, \dots, \text{слово}_n\}$
- Множество значений: $V = \mathbb{N}$ (натуральные числа)
- $Map(Doc_1.txt) =$
  $\{(\text{иметь}, 1), (\text{коллекция}, 1), (\text{документ}, 1), \dots, (\text{коллекция}, 1), (\text{документ}, 1), \dots\}$
- *Shuffle*:
  $\{(\text{иметь}, [1,1,1]), (\text{коллекция}, [1,1,1,1,1]), (\text{документ}, [1,1]), \dots\}$
- $Reduce(\text{коллекция}, [1,1,1,1,1]) = (\text{коллекция}, 5)$

```
def map(doc):
        for word in doc:
                yield word, 1
```

```
def reduce(word, values):
        yield word, sum(values)
```

Программный стек Hadoop

- ***Hadoop*** – наиболее популярная свободная реализация MapReduce
  - ***Hadoop Common***: набор инфраструктурных программных библиотек и утилит
  - ***HDFS***: распределенная файловая система
  - ***YARN***: система для планирования заданий и управления кластером
  - ***Hadoop MapReduce***: платформа программирования и выполнения распределенных MapReduce-вычислений

Экосистема Hadoop

- ***PIG***: реализация реляционной алгебры на базе Hadoop
- ***Hive***: реализация ограниченной версии SQL на базе Hadoop
- ***Spark***: программная платформа с открытым исходным кодом для реализации распределённой обработки неструктурированных и слабоструктурированных данных, входящий в экосистему проектов Hadoop. Использует специализированные примитивы для рекуррентной обработки в оперативной памяти, благодаря чему позволяет получать значительный выигрыш в скорости работы для некоторых классов задач
  - Предоставляет программные интерфейсы для языков Java, Scala, Python, R
  - Spark SQL: позволяет выполнять SQL-запросы над данными
  - Spark Streaming: надстройка для обработки потоковых данных
  - Spark Mllib: набор библиотек машинного обучения
  - GraphX: распределенная обработка графов

Конец лекции 11