Интеллектуальные информационные системы

MapReduce

Кафедра управления и интеллектуальных технологий НИУ «МЭИ»

2023 г.

MapReduce

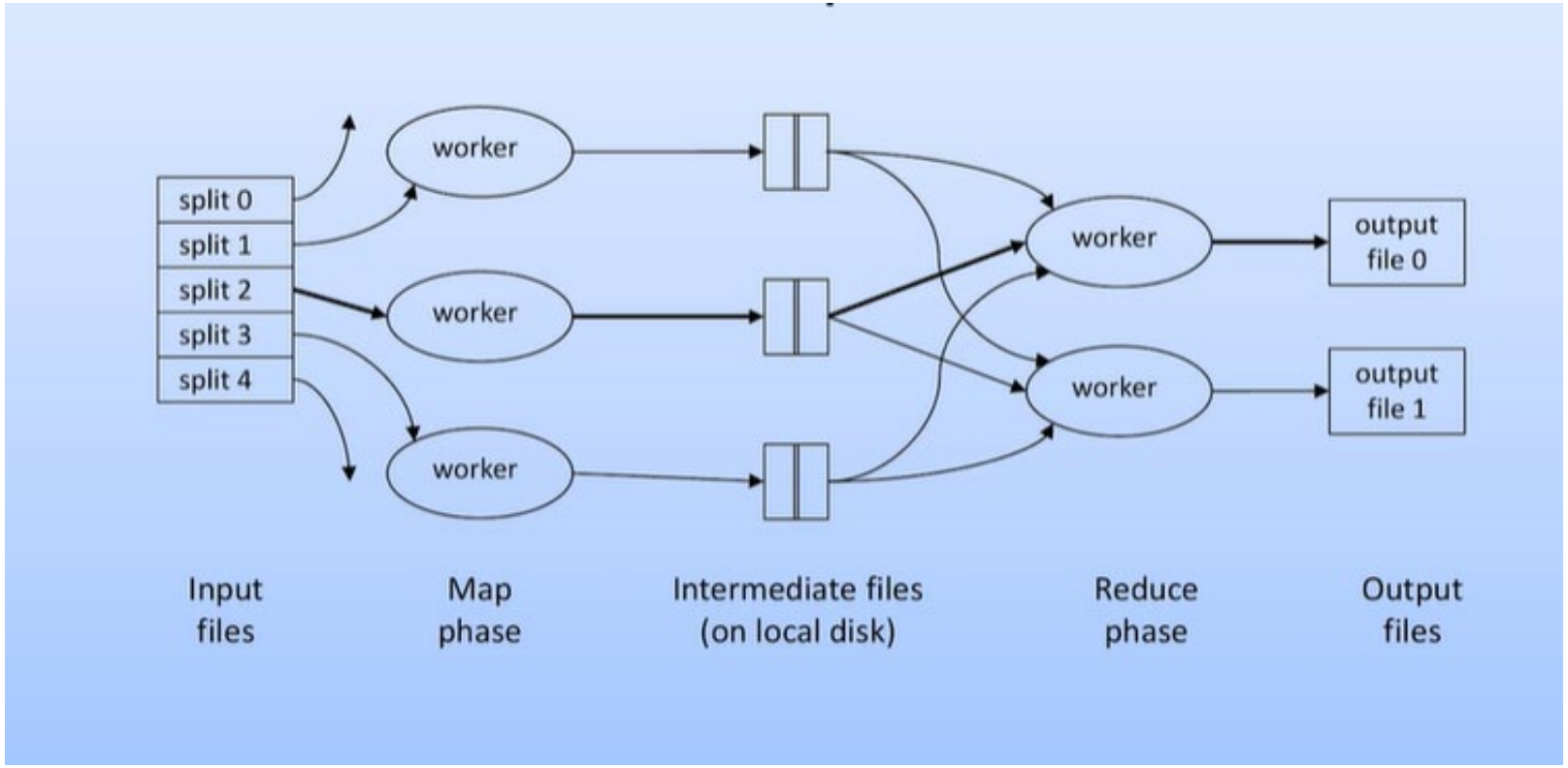
MapReduce – модель распределенных вычислений для обработки больших объемов данных.

Алгоритмы вычислений могут быть реализованы с помощью модели MapReduce, когда не хватает памяти для обработки на одном узле.

Map – обработка данных

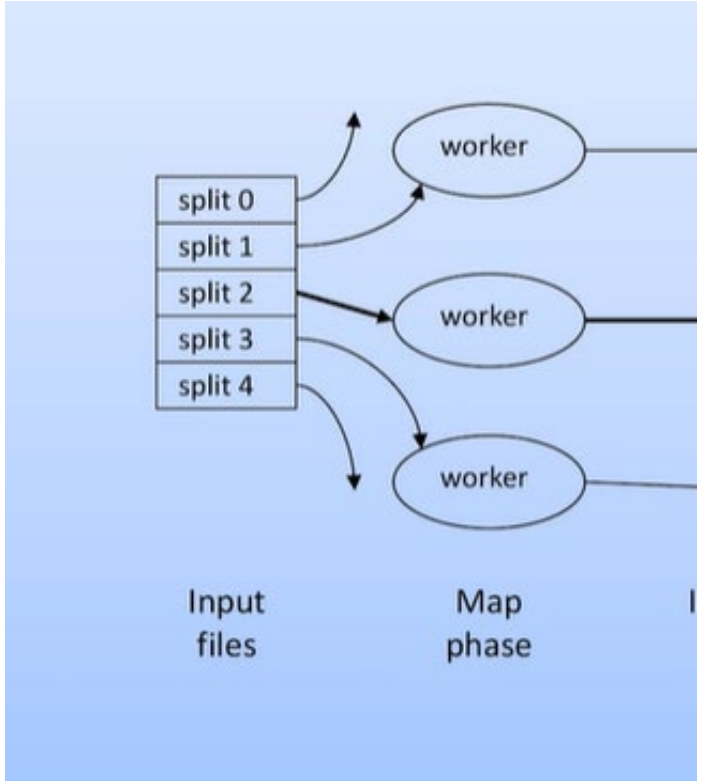
Reduce – свертка данных

MapReduce



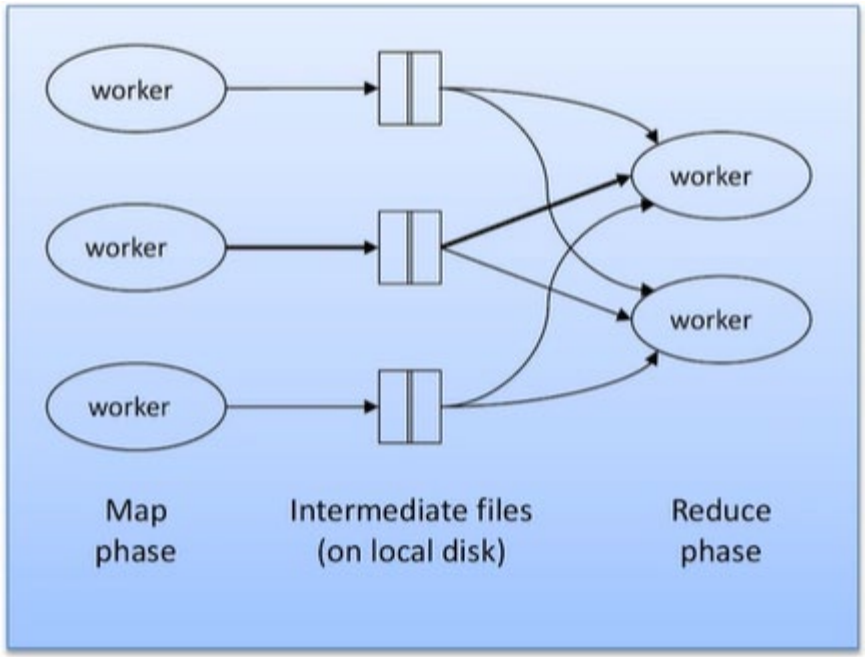
Worker – процесс. Процессы могут запущены параллельно и не общаются друг с другом
Каждый mapper пишет результат в файл в виде *key: value*

Map



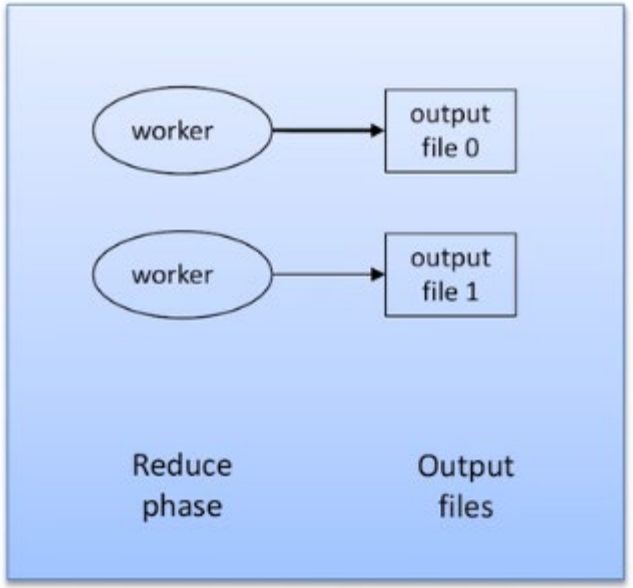
- Входной файл – разделяемый на сплиты.
- Данные в блоках не зависимы друг от друга и обрабатываются параллельно.
- Обычно – сплит = блок HDFS.
- Число Worker = количеству Split
- Обычно Worker там же, где лежит split (data locality)

Map –> Reduce



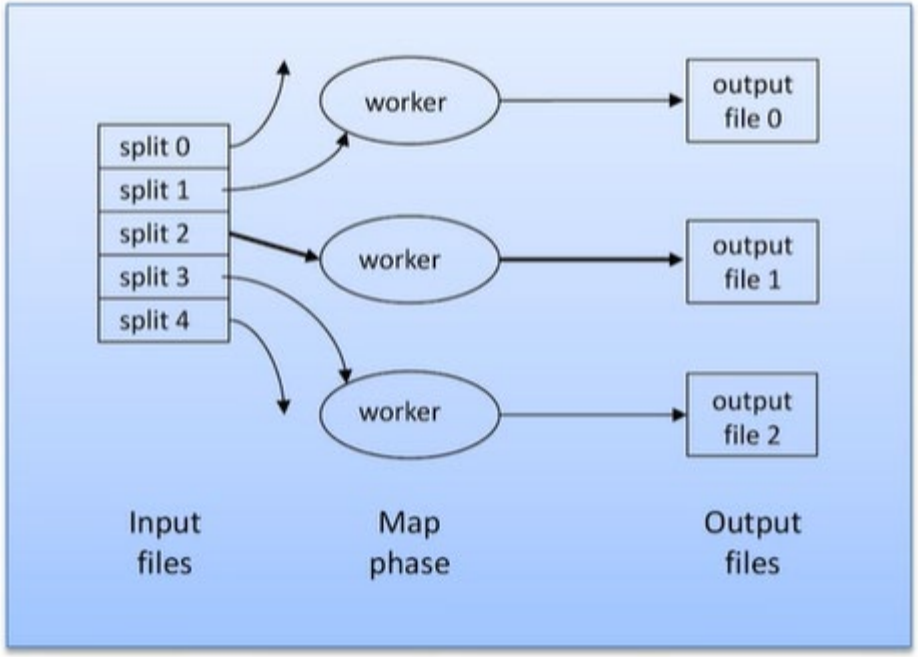
- Результат mapper – файл в локальной системе (не HDFS)
- Результат mapper – пара (key: value)
- Один ключ с разных mapper подаются на один reducer
- Копирование результатов mapper на reducer – shuffle.

Результаты



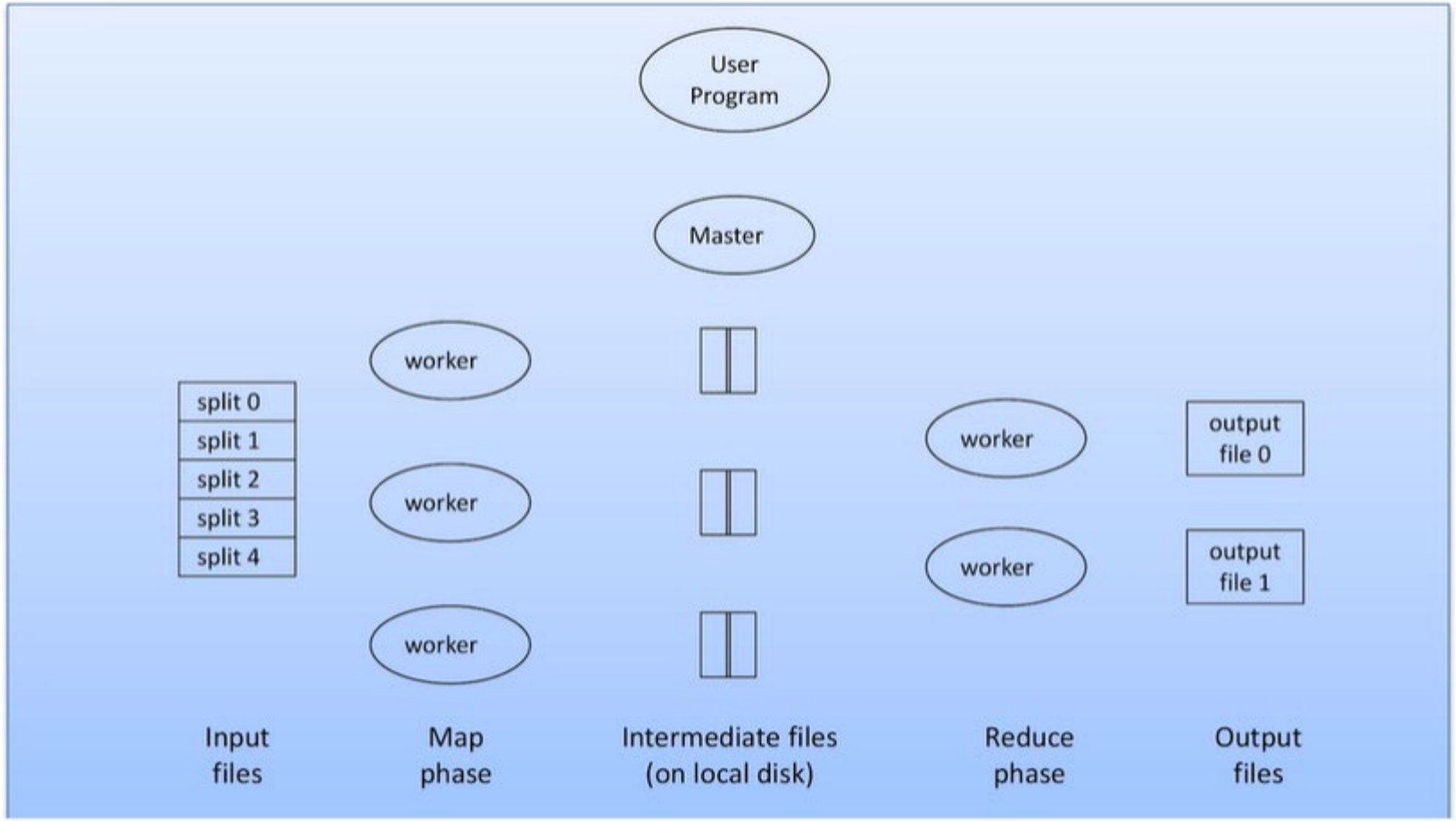
- Каждый reducer пишет в один файл в HDFS
- Результат в виде (key: value)

MapReduce без Reduce



- Результат записывается в HDFS.
- Обычно, далее следует MapReduce задача с тривиальным Map – просто считывание данных

MapReduce


User Program
Master
worker
worker
worker
split 0
split 1
split 2
split 3
split 4
worker
worker
output file 0
output file 1
Input files
Map phase
Intermediate files (on local disk)
Reduce phase
Output files

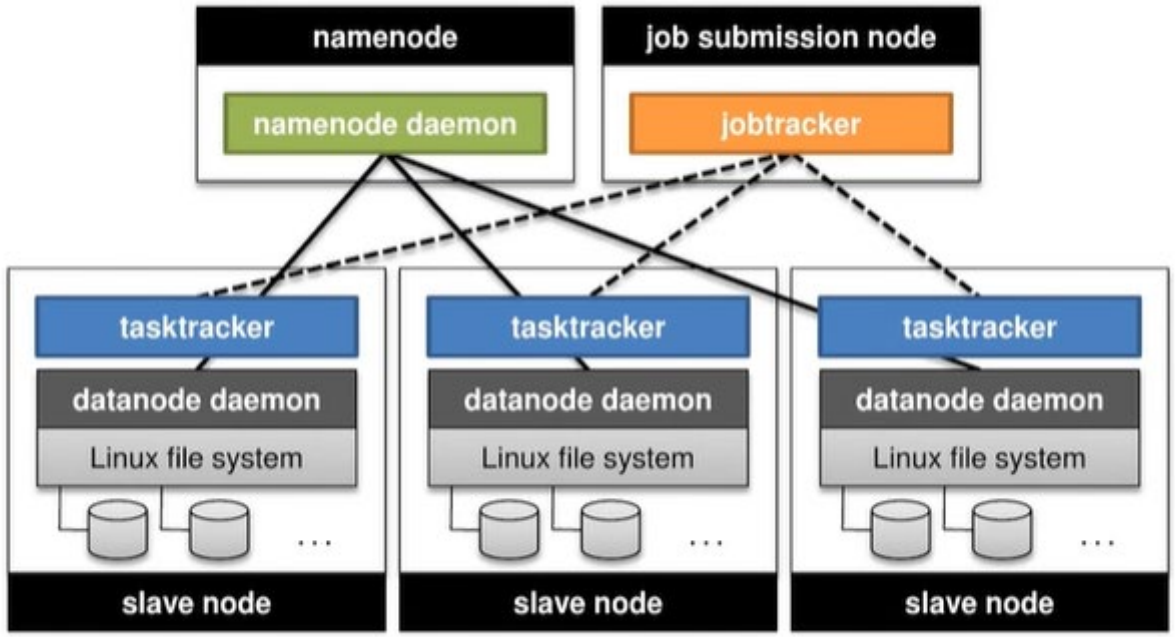
MapReduce в Hadoop

Основной функционал уже заложен в Hadoop

- Управление запуском задач
- Проверка выполнения задач
- Синхронизация выполнения задач
- Работа поверх HDFS

Разработчику остается реализовать логику работы

MapReduce в Hadoop



JobTracker:

- Отвечает за запуск задачи
- Управляет запуском TaskTracker
- Проверка выполнения работы
- Перезапуск оборвавшихся задач

TaskTracker:

- Управляет работой worker на данном сервере
- Статистика о выполнении задачи

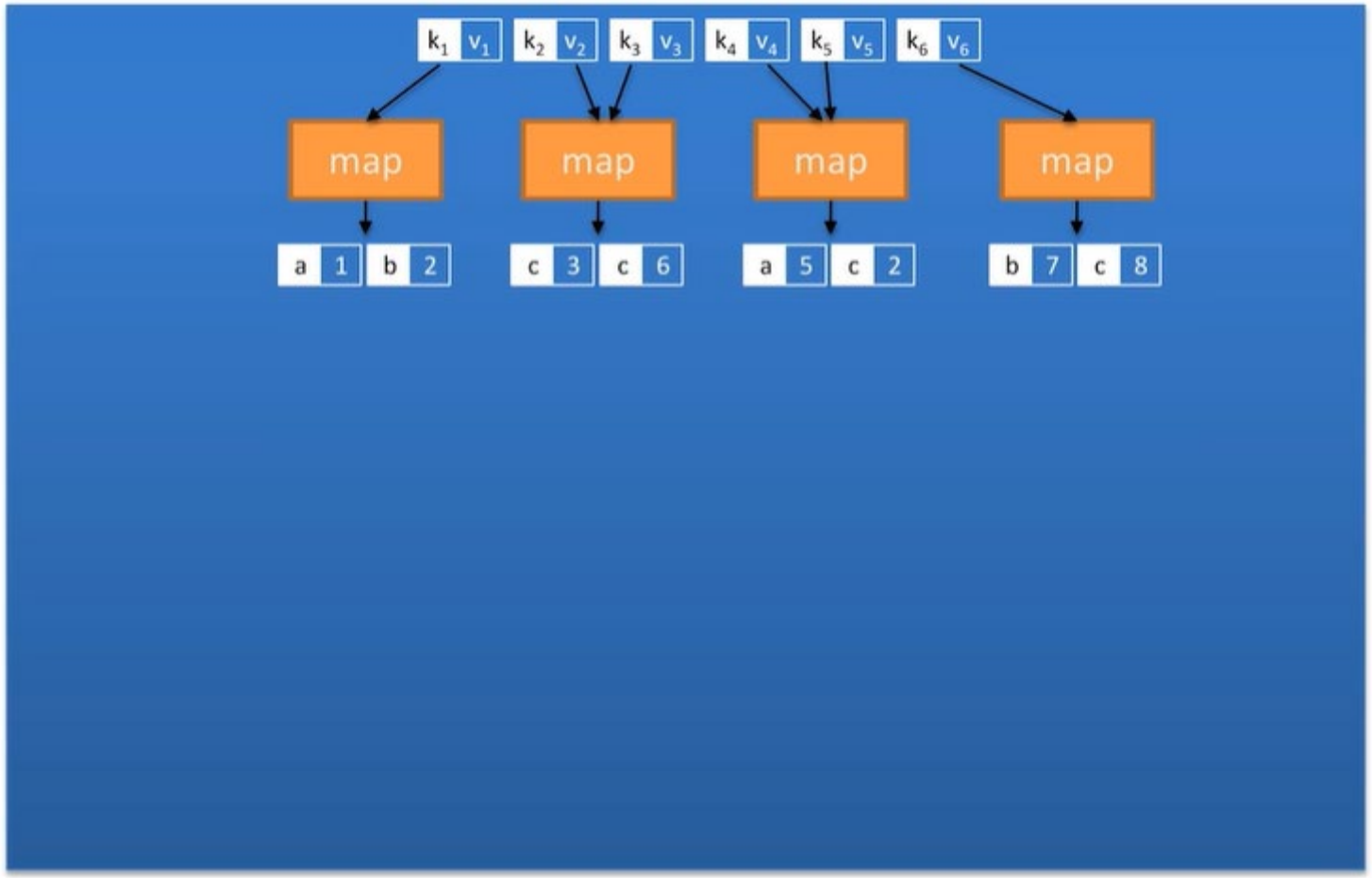
Система слотов

Для каждого TaskTracker определяется число слотов – блоков доступных ресурсов (CPU + RAM)
1 задача (task) = 1 слот
M мапперов + R редьюсеров = N слотов

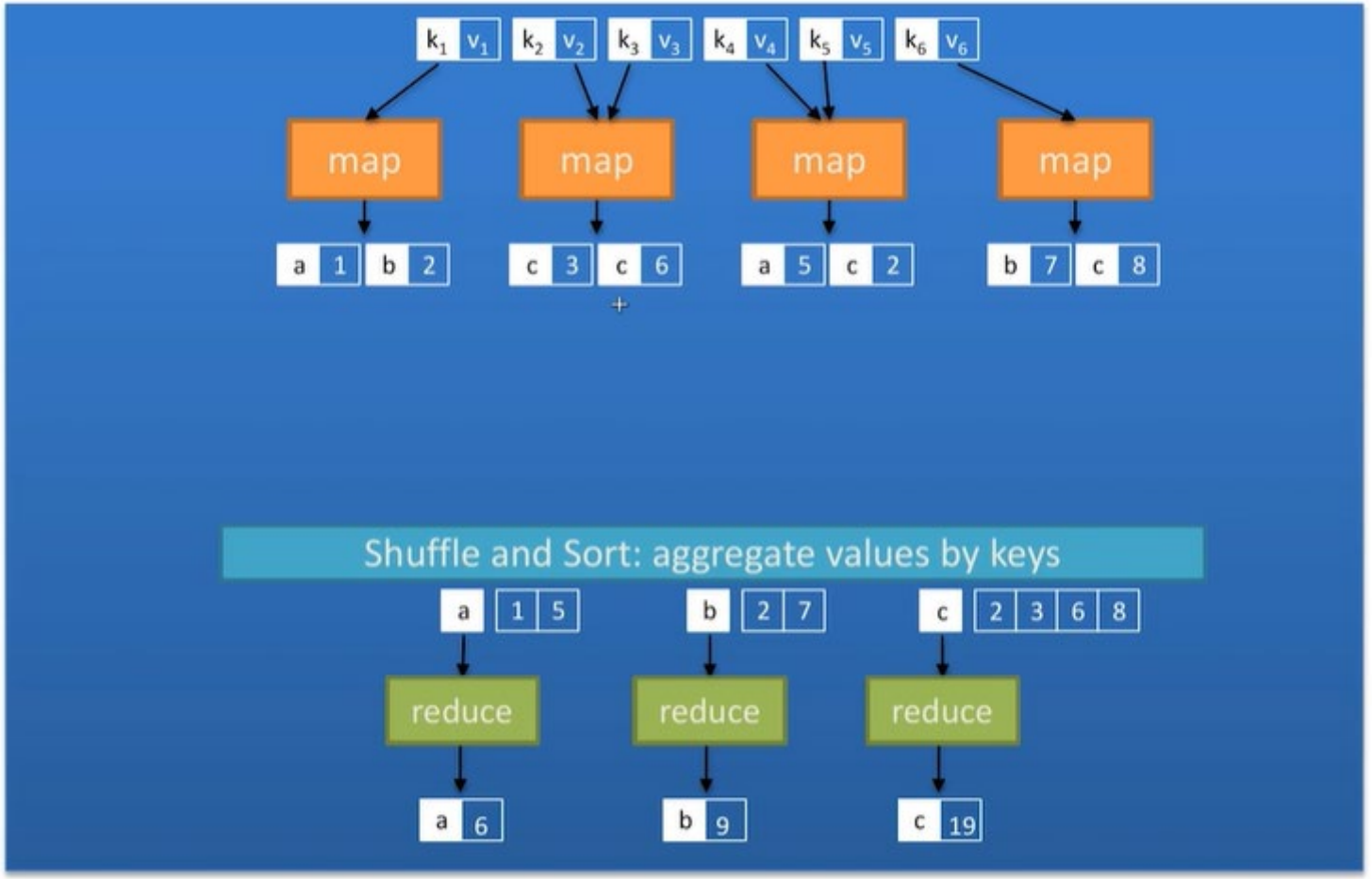
Заранее задается количество слотов на кластере для mapper и reducer.

В кластере **102** машины (2 из них для NameNode и JobTracker) со следующими характеристиками: **24** ядра, **64 Гб** памяти. Какое максимальное число **mapper**-процессов может быть запущено в таком кластере, если **1** слот занимает **1** ядро и **3 Гб** памяти, а mapper-слоты и reducer-слоты разбиты в соотношении **5:2**?

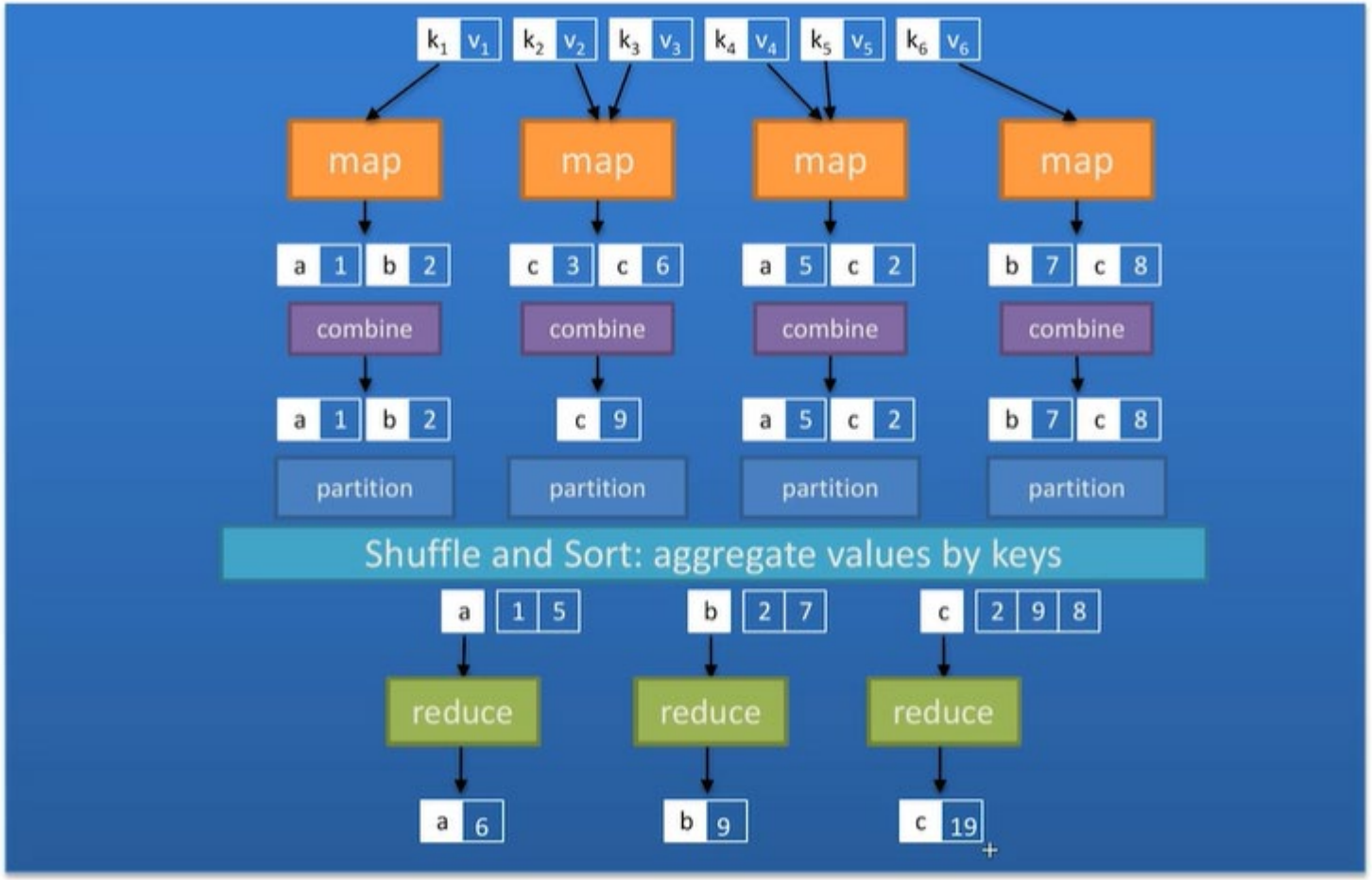
Word Count


k1 v1
k2 v2
k3 v3
k4 v4
k5 v5
k6 v6
map
map
map
map
a 1
b 2
c 3
c 6
a 5
c 2
b 7
c 8

Word Count


k1 v1
k2 v2
k3 v3
k4 v4
k5 v5
k6 v6
map
map
map
map
a 1 b 2
c 3 c 6
a 5 c 2
b 7 c 8
Shuffle and Sort: aggregate values by keys
a 1 5
b 2 7
c 2 3 6 8
reduce
reduce
reduce
a 6
b 9
c 19

Word Count


k1 v1
k2 v2
k3 v3
k4 v4
k5 v5
k6 v6
map
map
map
map
a 1 b 2
c 3 c 6
a 5 c 2
b 7 c 8
combine
combine
combine
combine
a 1 b 2
c 9
a 5 c 2
b 7 c 8
partition
partition
partition
partition
Shuffle and Sort: aggregate values by keys
a 1 5
b 2 7
c 2 9 8
reduce
reduce
reduce
a 6
b 9
c 19

Функции MR

Основные:

Map (k1,v1) -> list(k2, v2)

Reduce (k2, list(v2*)) -> list(k3, v3)

Опциональные:

Partition (k2, v2, |reducers|) -> номер reducer'а

- распределяет заданные ключи по заданным reducer'ам
- по умолчанию – hash(k2) mod n

Combine (k2, v2) -> list (k2, v2*)

- мини-редьюсер, которые выполняются после каждого mapper'а
- оптимизация сетевого трафика
- Не должен менять тип ключа и значения, т.к. фреймворк не гарантирует выполнение combine()

MapReduce Word Count

Подсчитать количество повторов каждого слова в текстовом файле

Map – считать строку, разбить на слова, вывести в формате (key, value): word tab 1

Reduce – считать строку, проверить что слово word не изменилось, просуммировать count. Иначе – вывести (word, count) и начать подсчет для нового слова

```
Map(String docid, String text):
    for each word w in text:
        Emit(w, 1);

Reduce(String term, Iterator<Int> values):
    int sum = 0;
    for each v in values:
        sum += v;
    Emit(term, sum);
```