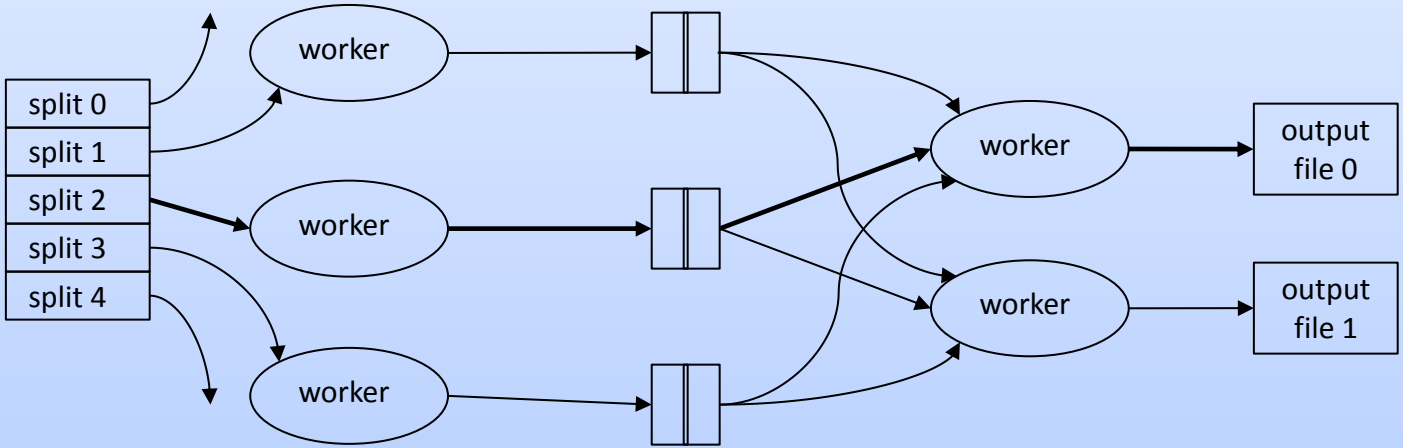
# Map-Reduce

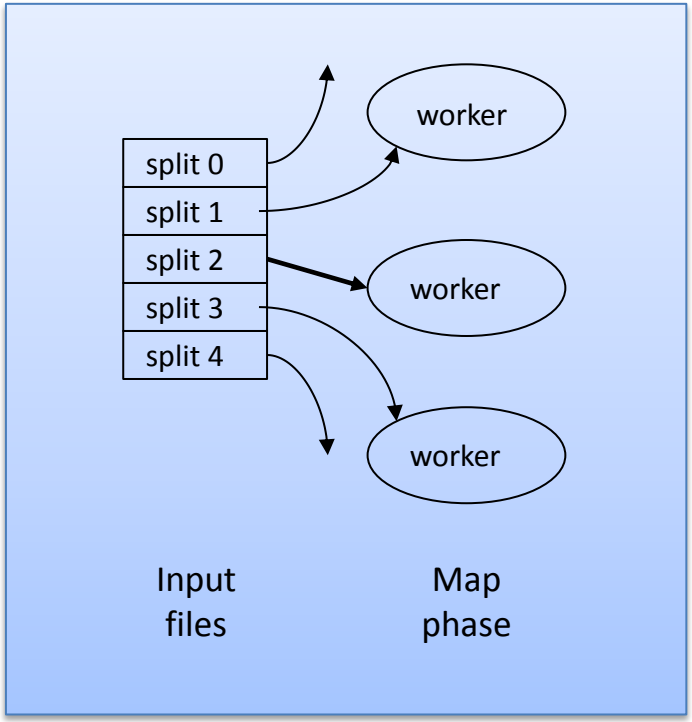
# в

# Hadoop

# Схема Map-Reduce

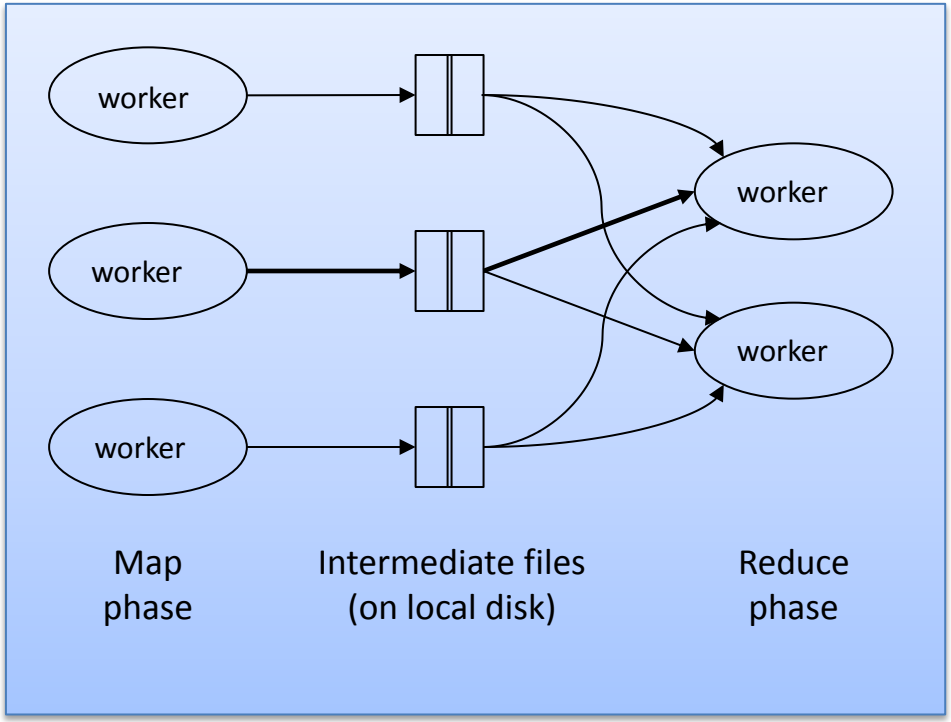


# Входные данные



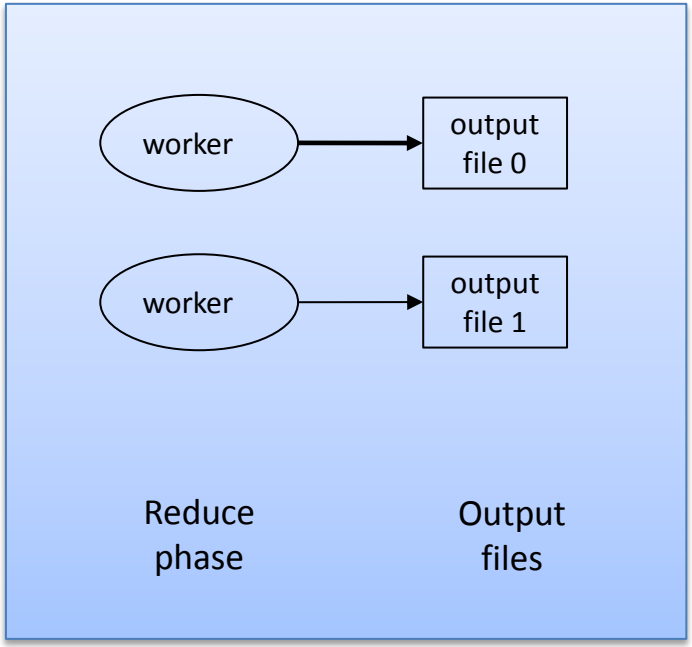

- Входные данные должны быть разделяемыми (splitable):
  - файл .gz не делется
  - Обычно split=block в hdfs
- Данные в каждом split должны быть независимыми
- Один worker обрабатывает один split (число worker = число split)
- Worker обычно запускается там, где лежит его split

# Передача данных между Map и Reduce



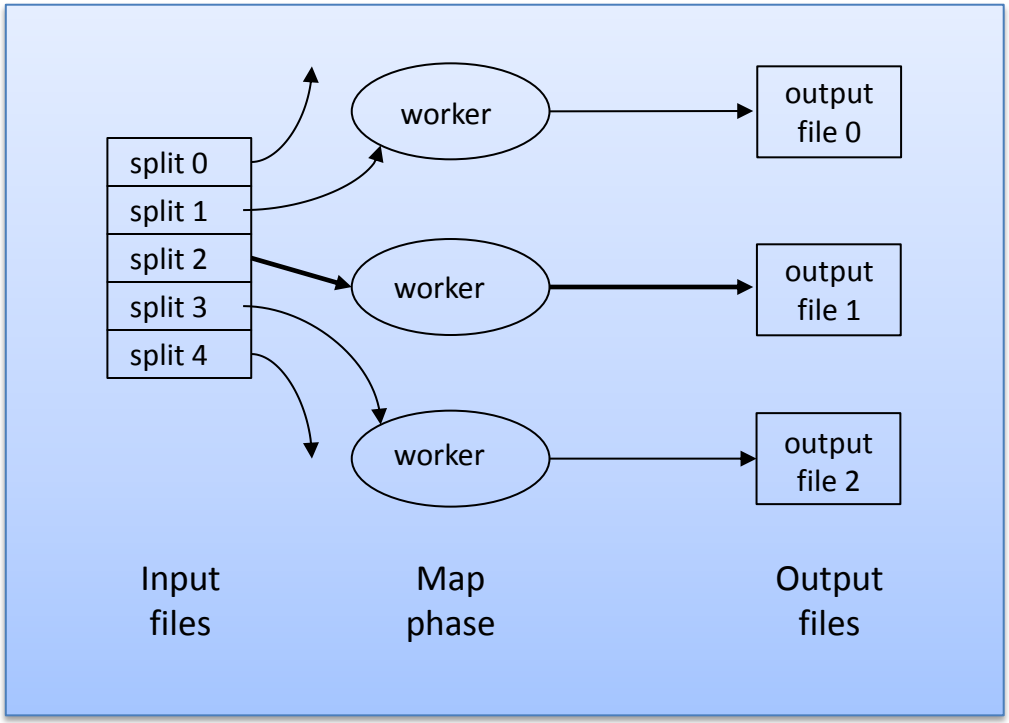

- Промежуточные данные пишутся на локальный диск
- Для каждого редьюсера маппер создает свой файл с данными
- Дынные – это пара (Key, Value)
- Данные с одним Key попадают на один Reducer
- Редьюсеры начинают работать после завершения всех мапперов

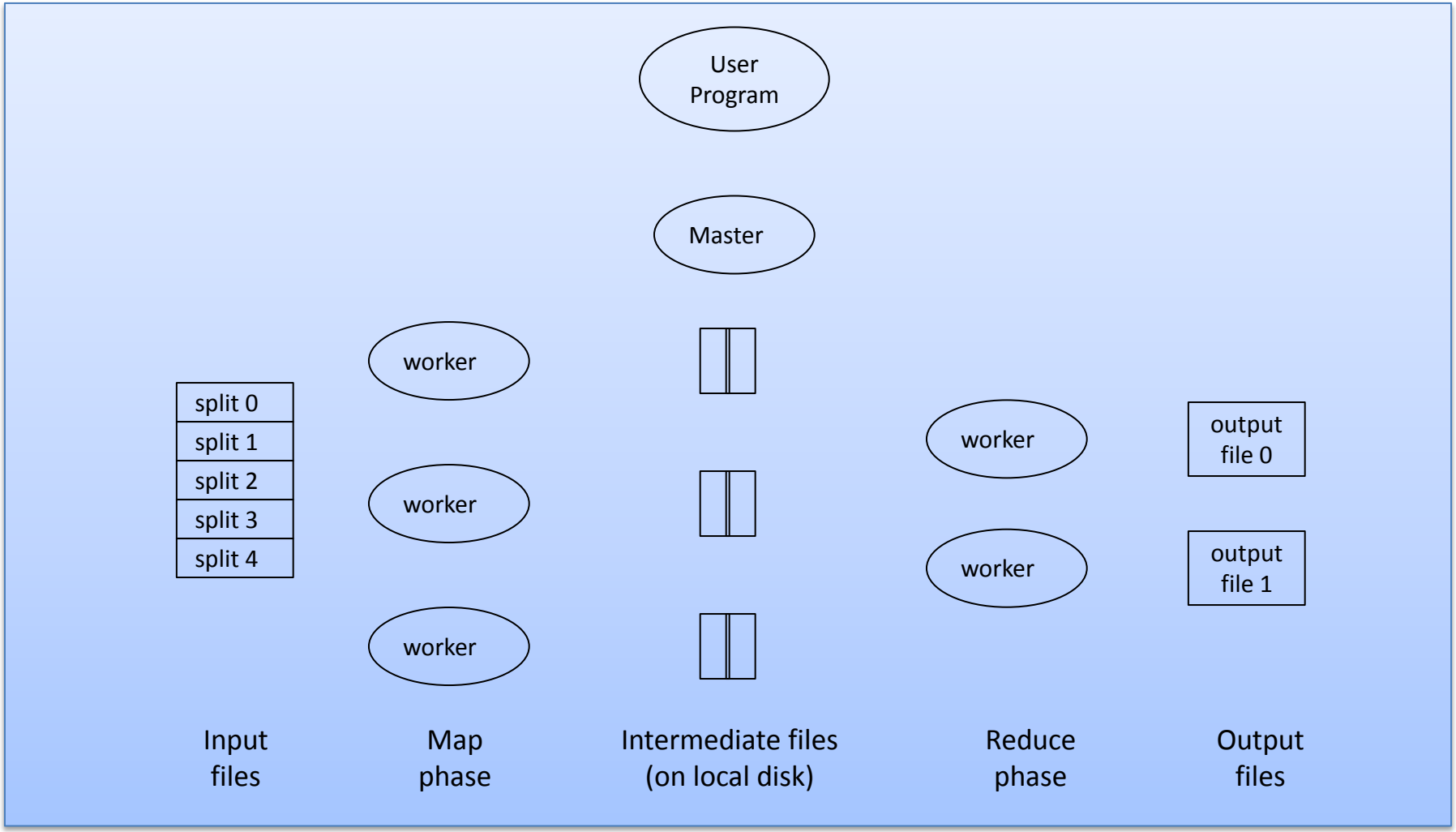
# Результат Map-Reduce задачи




- Каждый редьюсер пишет в один файл
- Число редьюсеров задает пользователь
- Данные сохраняются в hdfs
- Данные вида Key -> Value
- Формат данных определяется пользователем

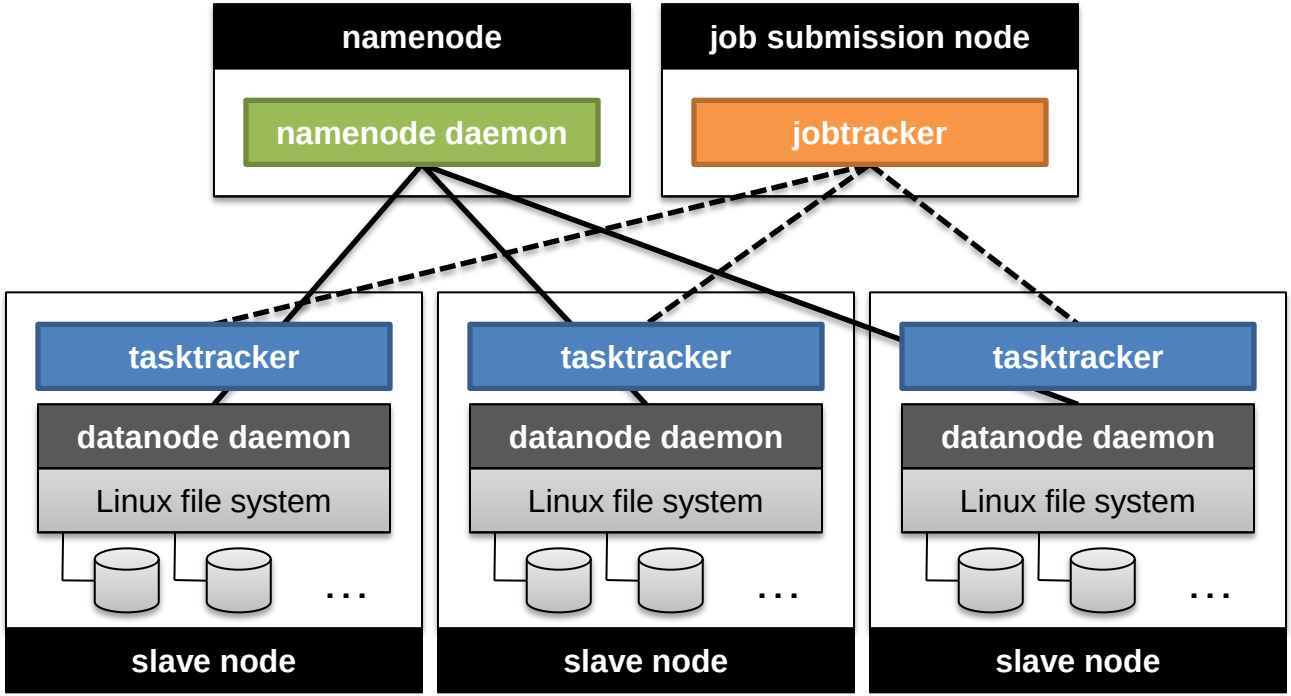
# Map-Reduce без Reduce




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

- Управление запуском задач
- Управление “*data distribution*”
- Управление синхронизацией этапов MapReduce
- Обработка ошибок и отказов
- Все работает поверх HDFS


namenode
job submission node
namenode daemon
jobtracker
tasktracker
tasktracker
tasktracker
datanode daemon
datanode daemon
datanode daemon
Linux file system
Linux file system
Linux file system
...
...
...
slave node
slave node
slave node

# JobTracker

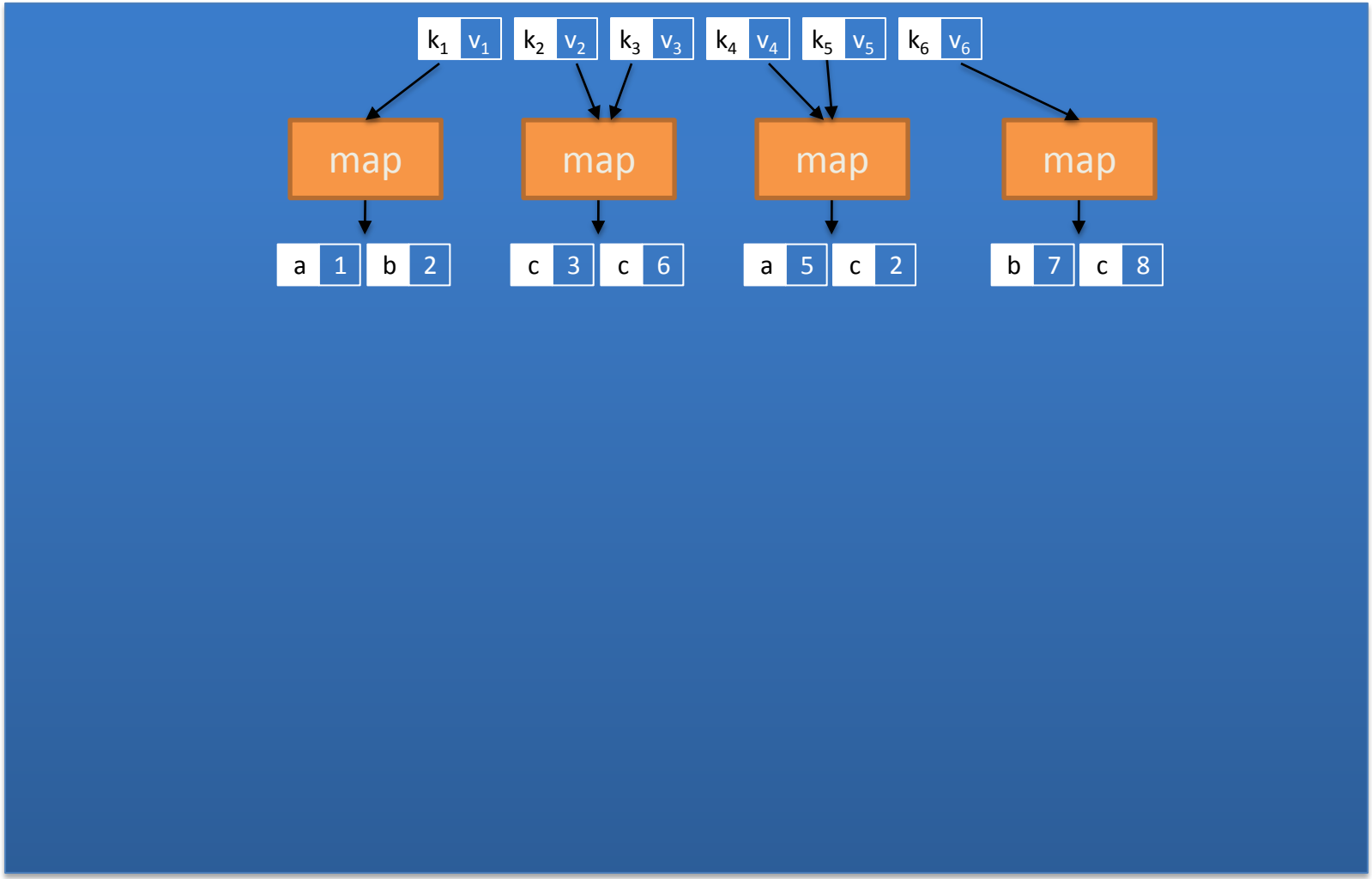
- Управляет запуском тасков и определяет, на каком ***TaskTracker*** будет запущен worker
- Управляет процессом работы MapReduce задач (*jobs*)
- Мониторит прогресс выполнения задач
- Перезапускает зафейленные или медленные таски

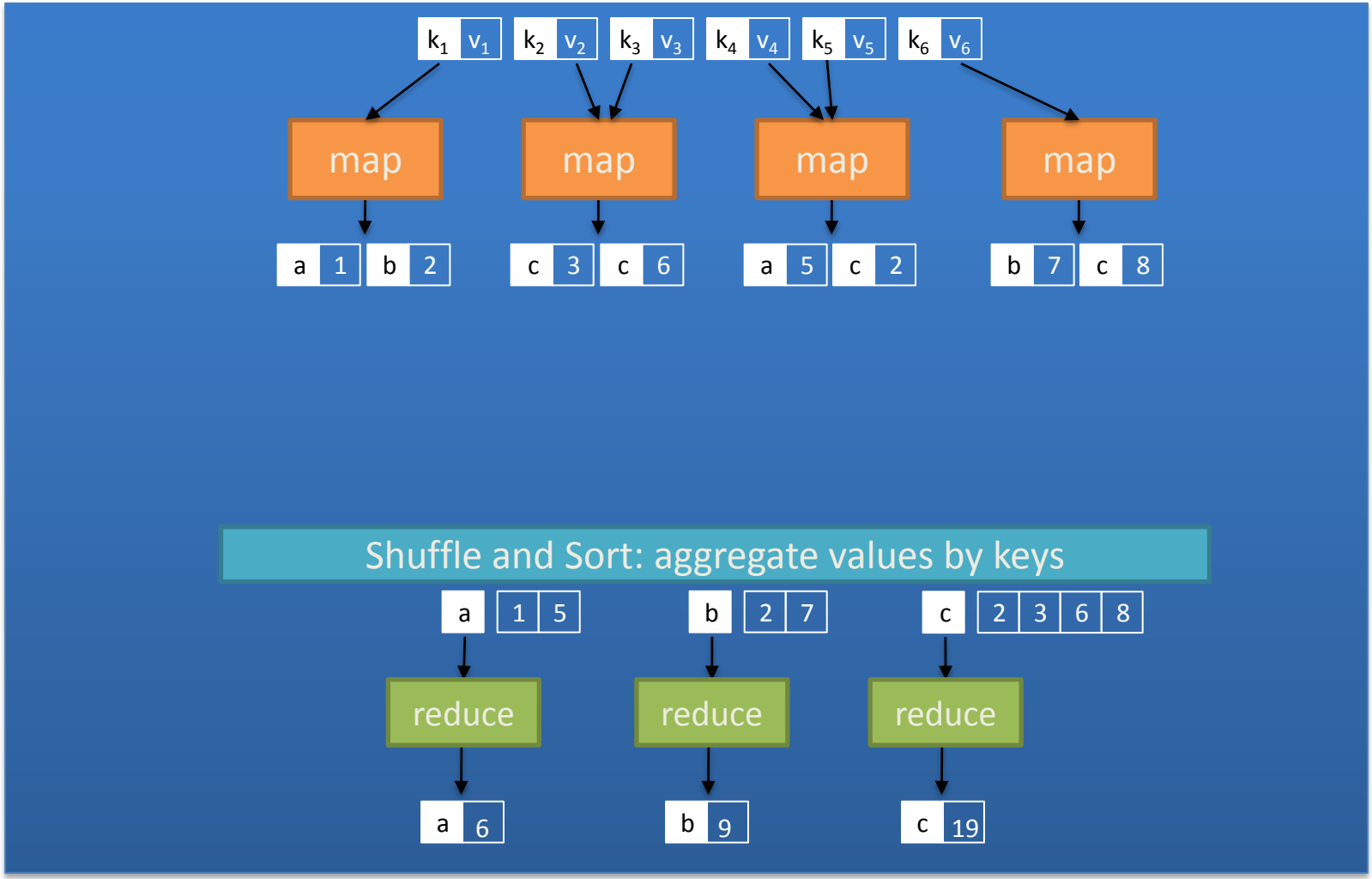
# TaskTracker

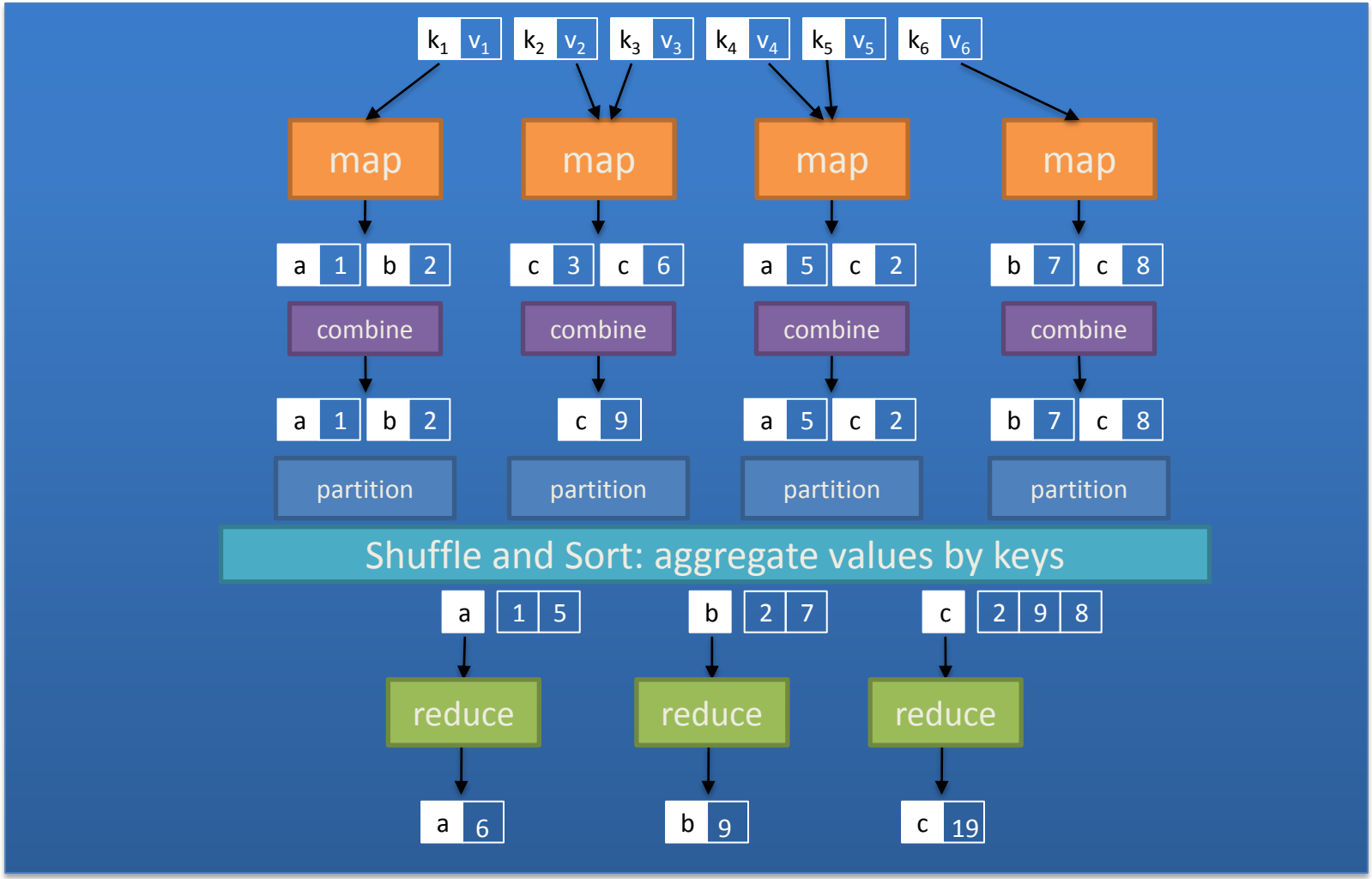
- Отвечает за работу всех worker на одном сервере
- Получает от **JobTracker** информацию о том, какой worker на каких данных нужно запустить
- Посылает в **JobTracker** статистику о прогрессе выполнения задачи (counters)
- Сообщает в **JobTracker** об удачном завершении или падении worker

# Система слотов

- Для каждого TaskTracker определяется число слотов (slots)
- Таск запускается на одном слоте
- M мапперов + R редьюсеров = N слотов
- Для каждого слота определяется кол-во потребляемой ОЗУ


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


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


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

# Основные функции

Программист определяет:

- **map** (k1, v1) → list(k2, v2)
- **reduce** (k2, list(v2*) )→ list(k3, v3)

# Опциональные функции

**partition** (k2, v2, |reducers|) → № of reducer

- распределяет ключи по редьюсерам
- Часто просто хеш от key: *hash(k2) mod n*

# Опциональные функции

**combine** (k2, v2) → list(k2, v2’)

- Мини-reducers которые выполняются после завершения фазы map
- Используется в качестве оптимизации для снижения  сетевого трафика на reduce
- (!) Не должен менять тип ключа и значения