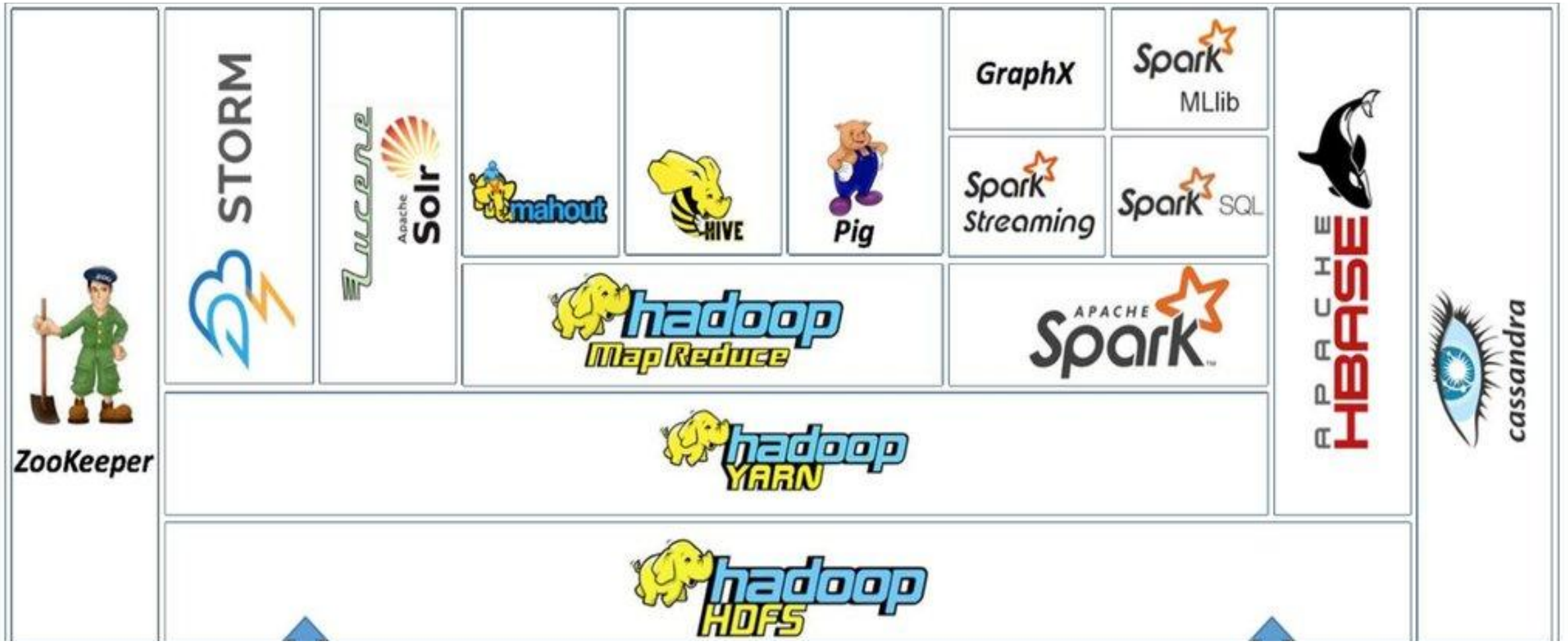
GeekBrains

MapReduce & YARN

Принцип работы распределенных вычислений, архитектура менеджера ресурсов

ЭКОСИСТЕМА HADOOP



HADOOP MAPREDUCE

APACHE HADOOP v1

HADOOP MAPREDUCE

- Названия **Map** и **Reduce** заимствованы из функциональных языков программирования

HADOOP MAPREDUCE

GeekBrains

- Названия **Map** и **Reduce** заимствованы из функциональных языков программирования
- Перемещение **кода к данным**, а не наоборот

HADOOP MAPREDUCE

GeekBrains

- Названия **Map** и **Reduce** заимствованы из функциональных языков программирования
- Перемещение **кода к данным**, а не наоборот
- Данные представляют собой пары вида **“ключ-значение”**

HADOOP MAPREDUCE

GeekBrains

- Названия **Map** и **Reduce** заимствованы из функциональных языков программирования
- Перемещение **кода к данным**, а не наоборот
- Данные представляют собой пары вида **“ключ-значение”**
- Состоит из JobTracker и TaskTracker

JOB TRACKER

Сервис запущен в единственном экземпляре на весь кластер. Строит план исполнения:

- определяет входные файлы, для запуска задач на соответствующих узлах
- перезапускает упавшие задачи

Job — план исполнения. Включает MapReduce конфигурацию, входные выходные пути, классы InputFormat, OutputFormat, Mapper, Reducer, Combiner, Partitioner.

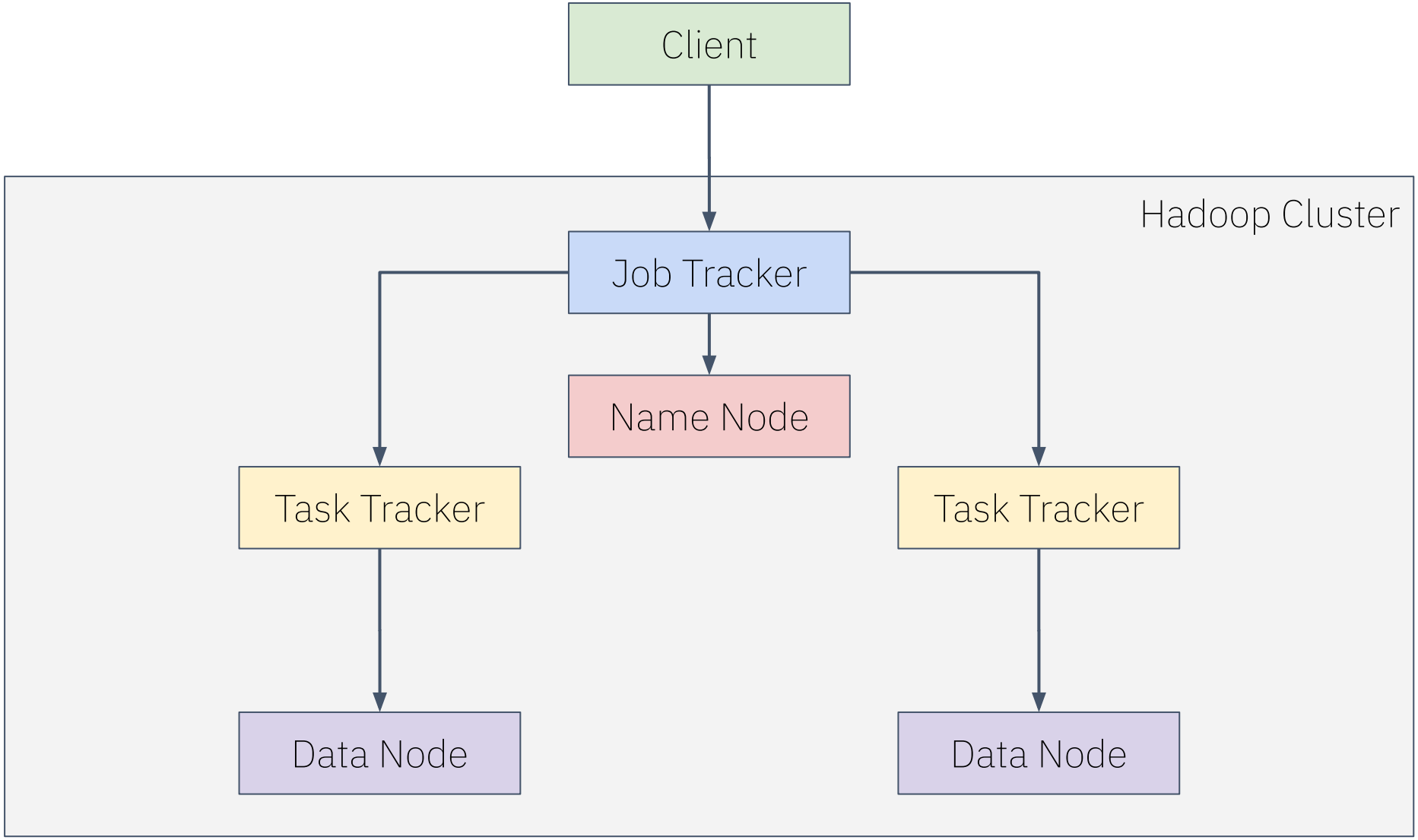
TASK TRACKER

Сервис запущен на каждой ноде кластера, где возможно исполнение расчетов

- запускает план исполнения, полученный от JobTracker
- отчитывается перед JobTracker о состоянии выполнения задач

Task — запущенный экземпляр исполняемого кода

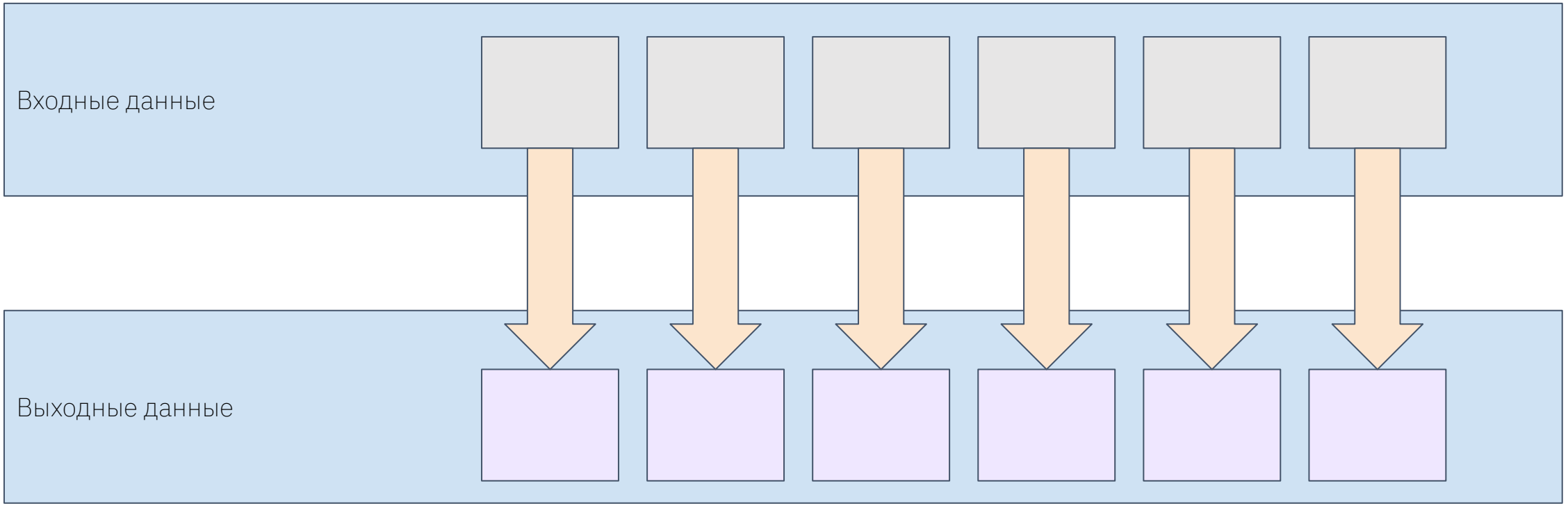
ЗАПУСК ПРИЛОЖЕНИЯ



MAP И REDUCE

MAP

Операция Map применяет к каждому блоку массива данных преобразующую функцию



ОСОБЕННОСТИ MAP

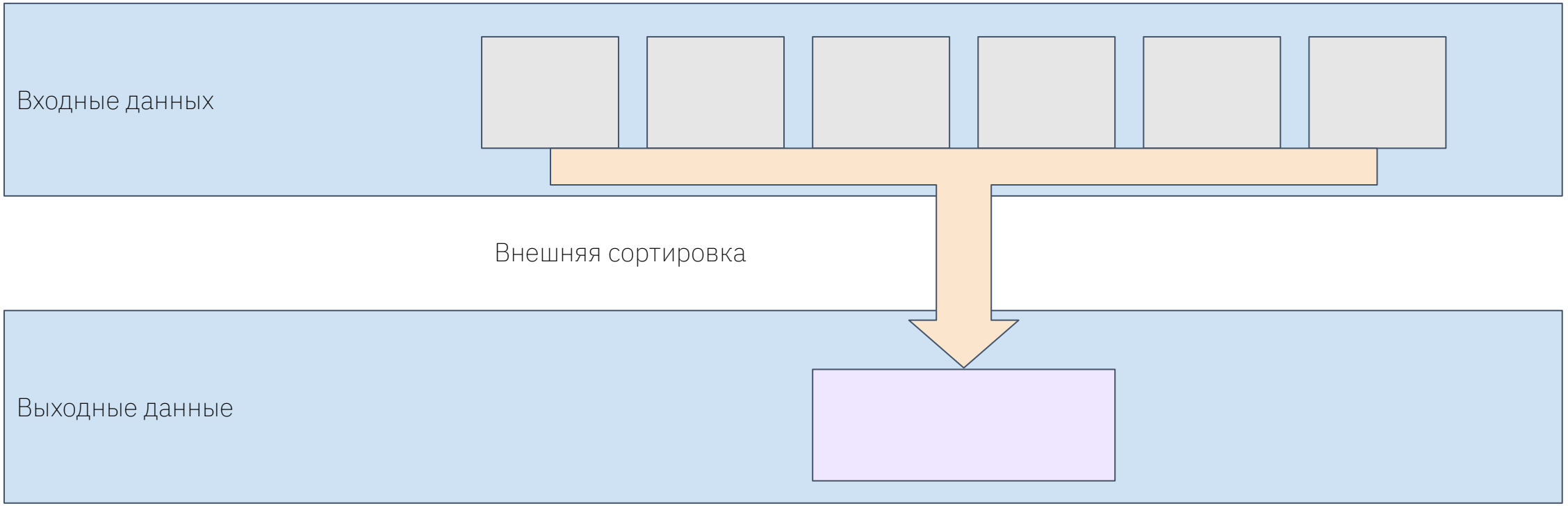
- Способен получать данные из различных источников. По умолчанию встроены реализация чтения из текстовых файлов (ключ от значения отделен табуляцией) и SequenceFile (ключ и значение хранятся в сериализованном виде). Можно написать собственные входные форматы данных. Например:
  - чтение из нескольких директорий
  - разбиение файла на части и чтение только своей части
  - чтение json
  - чтение из kafka

ОСОБЕННОСТИ MAP

- Способен получать данные из различных источников. По умолчанию встроены реализация чтения из текстовых файлов (ключ от значения отделен табуляцией) и SequenceFile (ключ и значение хранятся в сериализованном виде). Можно написать собственные входные форматы данных. Например:
  - чтение из нескольких директорий
  - разбиение файла на части и чтение только своей части
  - чтение json
  - чтение из kafka
- Количество маперов равно количеству входных сущностей (сплитов), согласно входному формату данных.

REDUCE

Операция Reduce получает на вход итератор со всеми входными данными, отсортированными по ключу, и применяет к ним заданную функцию



ОСОБЕННОСТИ REDUCE

- **Количество** редьюсеров задается программно в коде

ОСОБЕННОСТИ REDUCE

- **Количество** редьюсеров задается программно в коде
- **Партиционирование** данных по редьюсерам происходит согласно формуле:

key.hashcode() % reducers.size()

ОСОБЕННОСТИ REDUCE

- **Количество** редьюсеров задается программно в коде
- **Партиционирование** данных по редьюсерам происходит согласно формуле:

key.hashcode() % reducers.size()

- Формулу партиционирования можно изменить задав собственный Partitioner

ОСОБЕННОСТИ REDUCE

GeekBrains

- **Количество** редьюсеров задается программно в коде
- **Партиционирование** данных по редьюсерам происходит согласно формуле:

key.hashcode() % reducers.size()

- Формулу партиционирования можно изменить задав собственный Partitioner
- Перед попаданием на редьюсер данные **сортируются** по ключу

ОСОБЕННОСТИ REDUCE

GeekBrains

- **Количество** редьюсеров задается программно в коде
- **Партиционирование** данных по редьюсерам происходит согласно формуле:

key.hashcode() % reducers.size()

- Формулу партиционирования можно изменить задав собственный Partitioner
- Перед попаданием на редьюсер данные **сортируются** по ключу
- Редьюсер может быть также **комбинатором**: в этом случае он обработает результат маперов, запущенных на той же ноде кластера (локальный редьюс)

ОСОБЕННОСТИ REDUCE

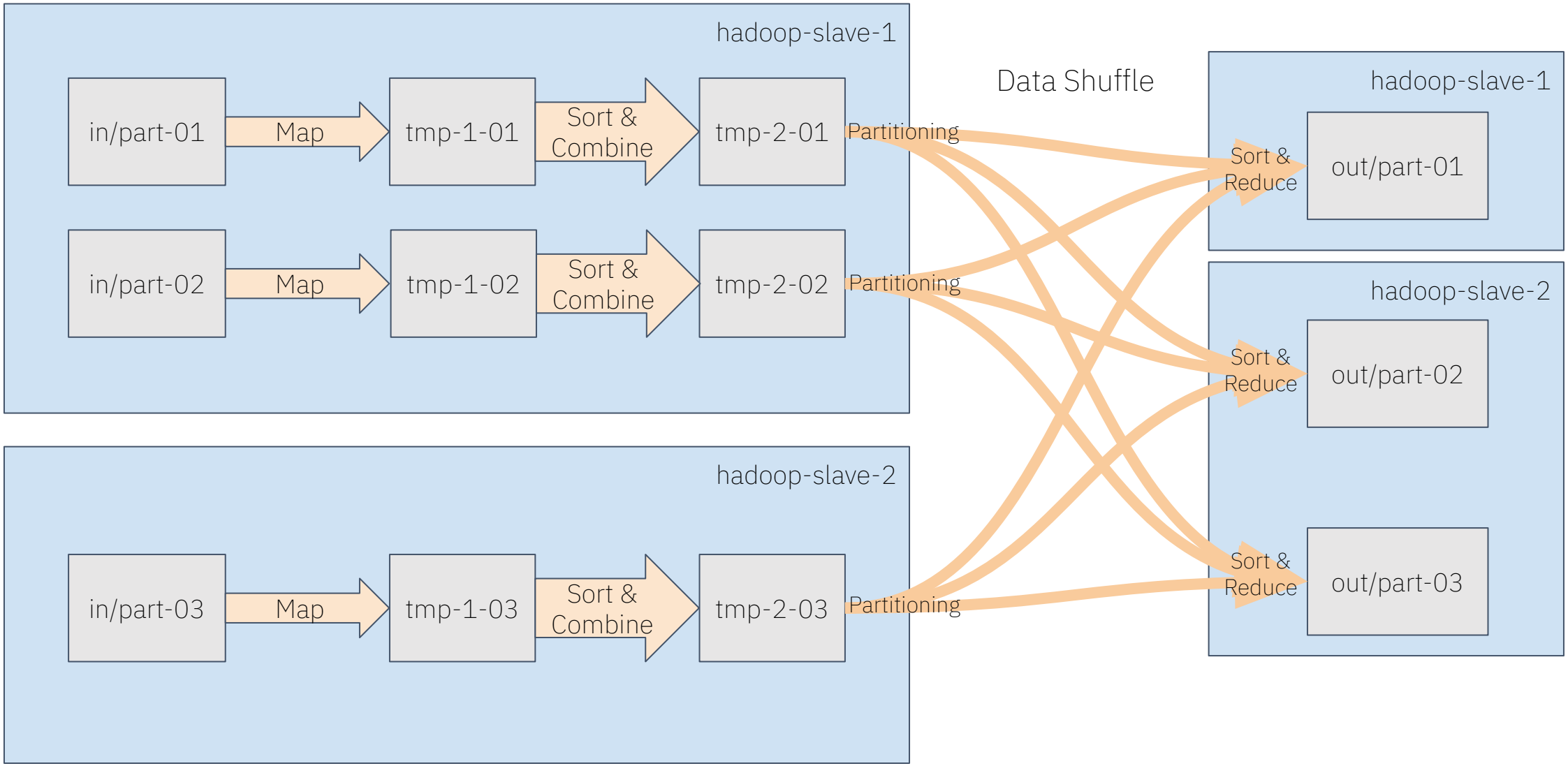
GeekBrains

- **Количество** редьюсеров задается программно в коде
- **Партиционирование** данных по редьюсерам происходит согласно формуле:

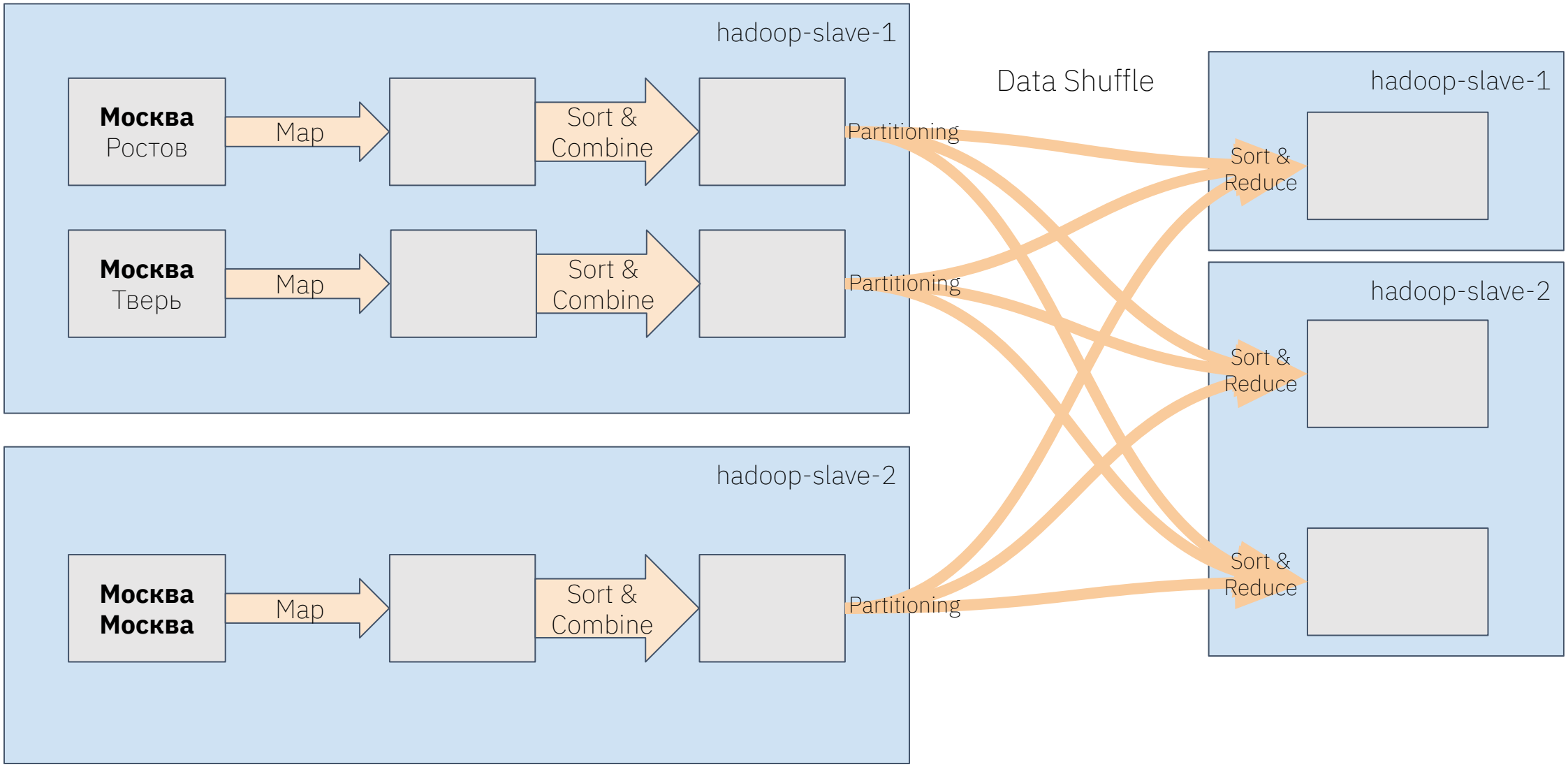
key.hashcode() % reducers.size()

- Формулу партиционирования можно изменить задав собственный Partitioner
- Перед попаданием на редьюсер данные **сортируются** по ключу
- Редьюсер может быть также **комбинатором**: в этом случае он обработает результат маперов, запущенных на той же ноде кластера (локальный редьюс)
- Количество выходных файлов равно количество редьюсеров. Если редьюсеров нет (Map only job), то количество выходных файлов будет равно количеству мапперов.

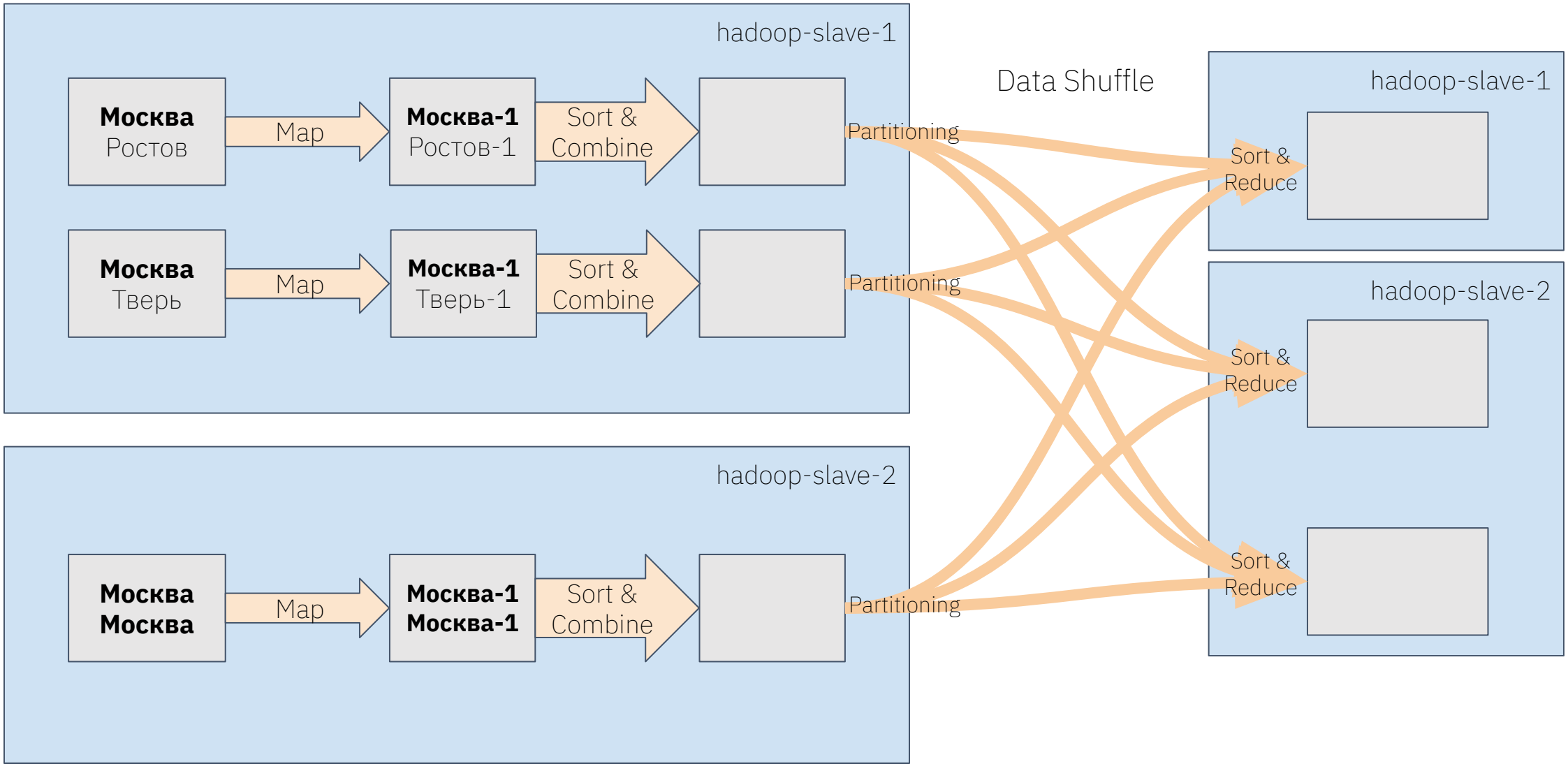
ПРИМЕР MAPREDUCE



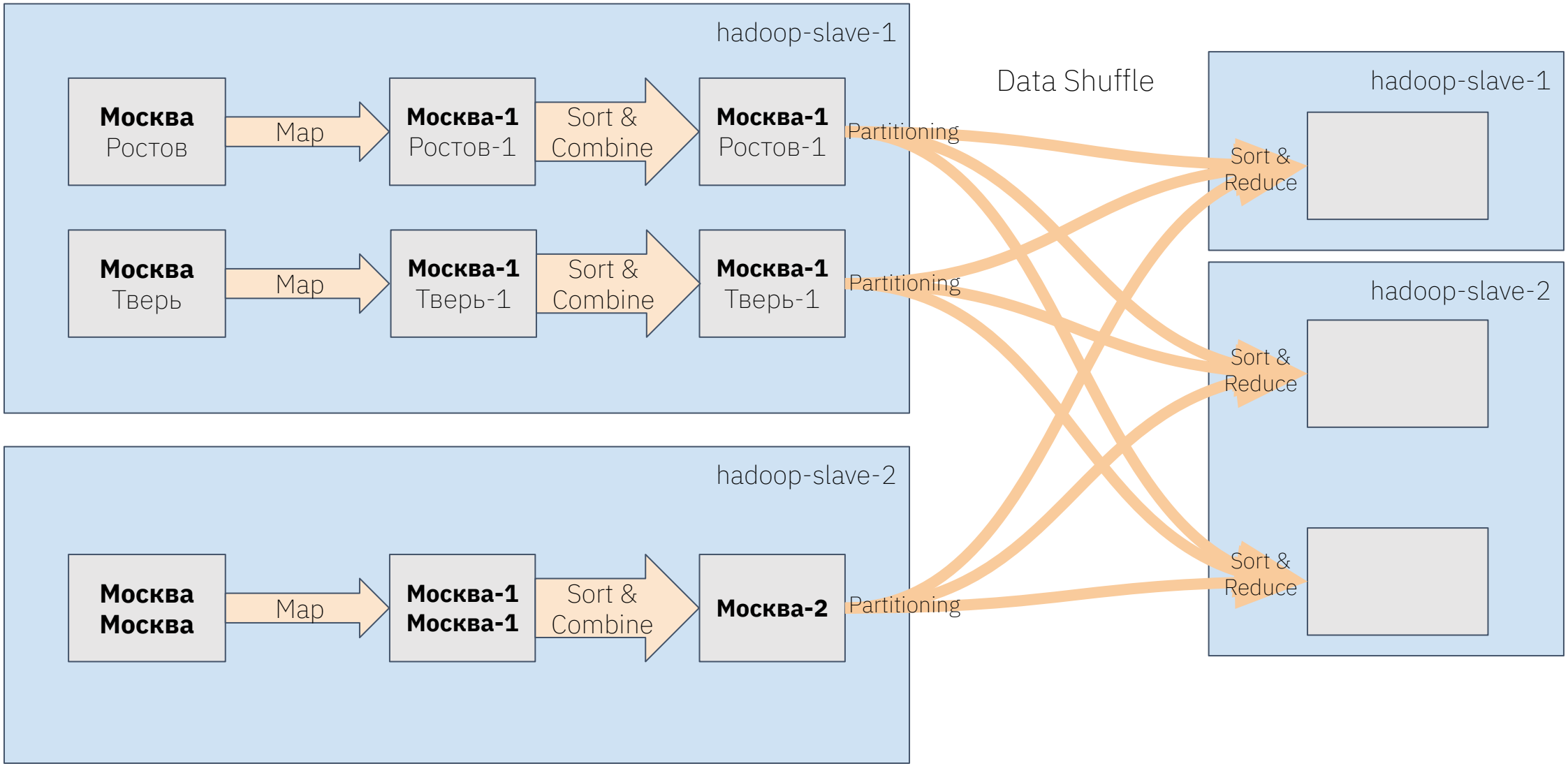
MAPREDUCE WORDCOUNT



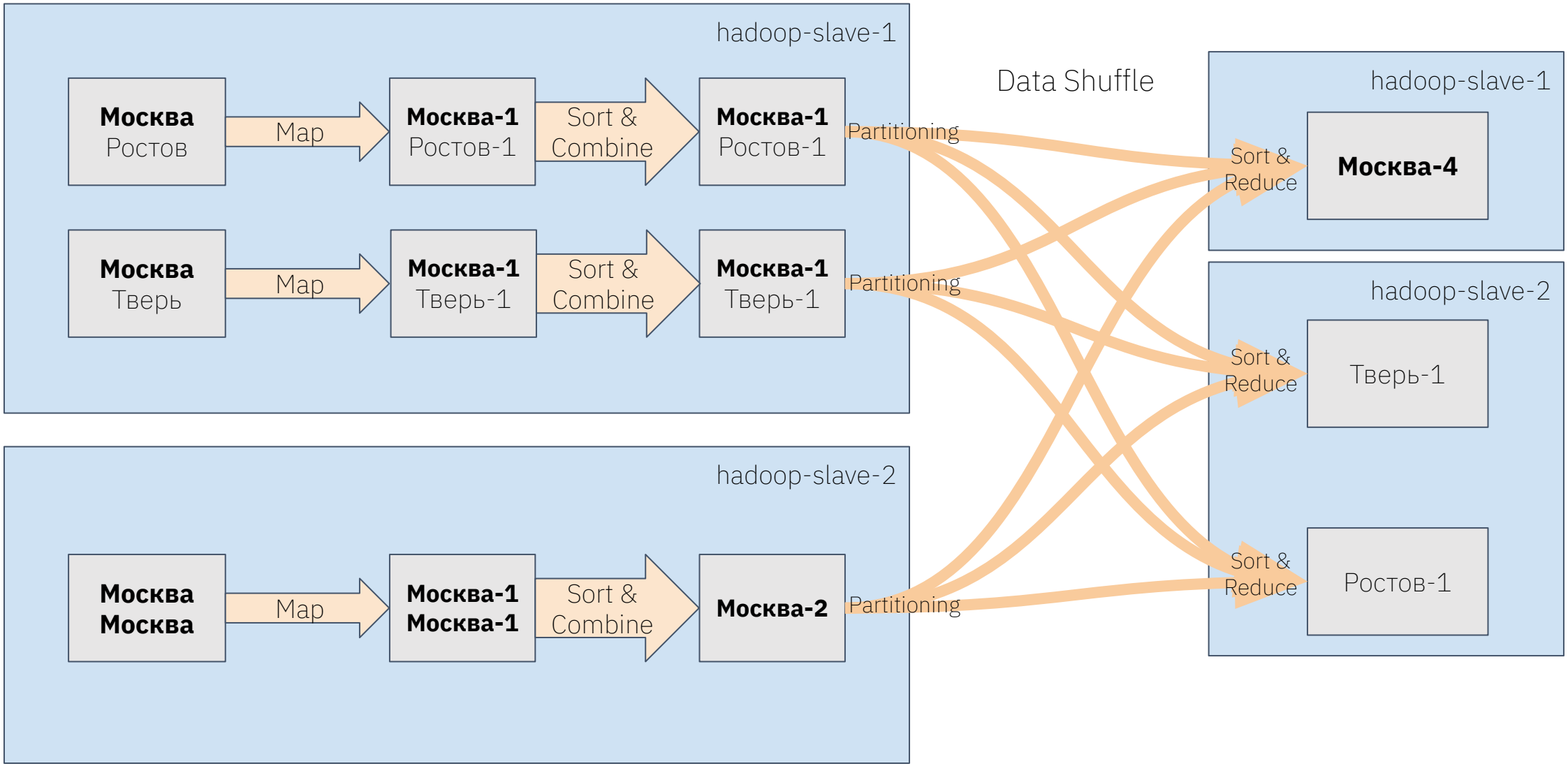
MAPREDUCE WORDCOUNT



MAPREDUCE WORDCOUNT

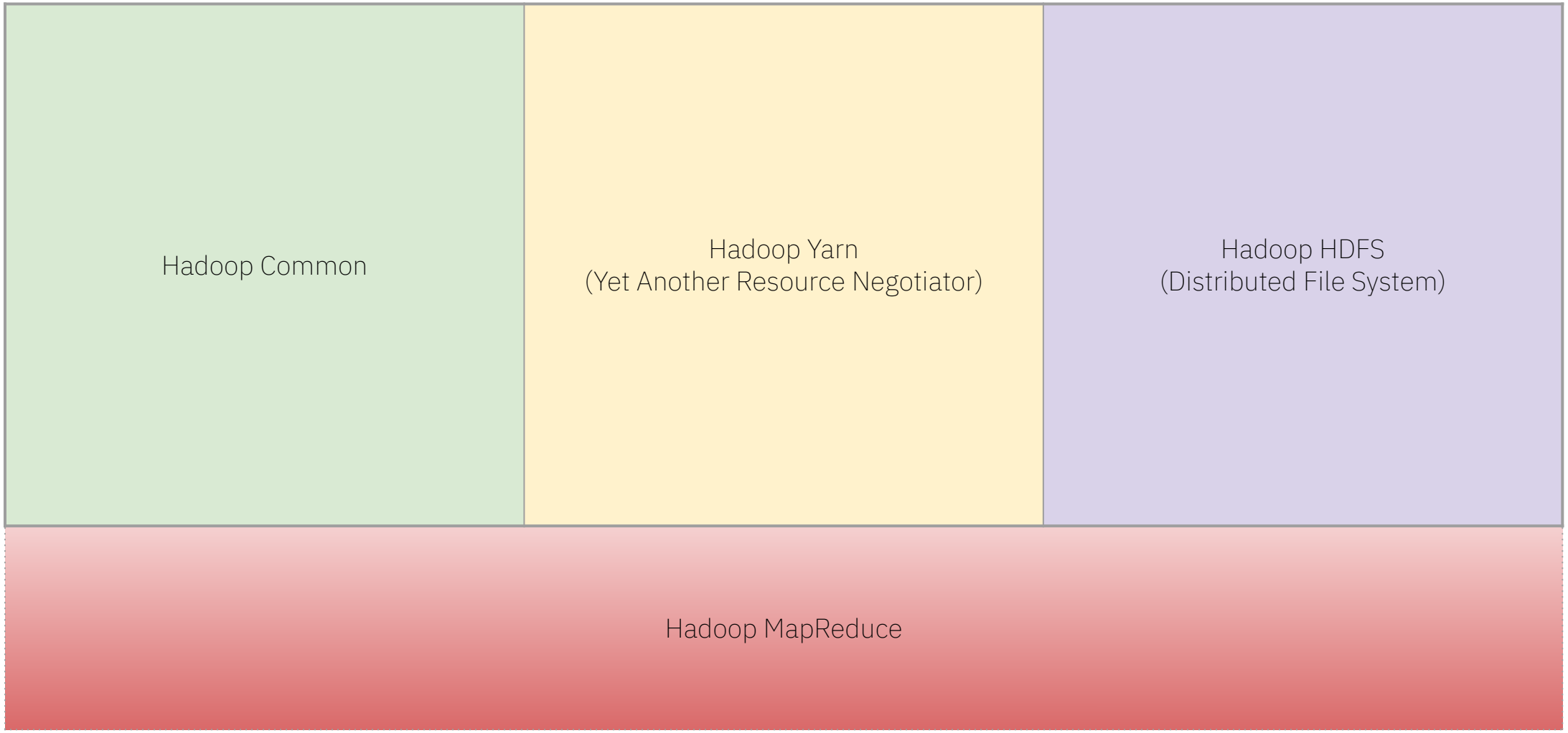


MAPREDUCE WORDCOUNT



HADOOP YARN

APACHE HADOOP v2+



HADOOP YARN

Отвечает за выделение ресурсов для запуска задач. Состоит из:

- Resource Manager (RM)
- Node Manager (NM)
- Container
- Application Master (AM)

RESOURCE MANAGER

- Работает на отдельном сервере
- Знает какие ресурсы есть на каждой ноде в кластере
- Распределяет воркеры по кластеру для приложений
- Реализует логику планировщика запуска
- Был выделен из MapReduce в Hadoop 2.x в виде отдельной сущности
    - Job в терминологии YARN теперь называется Application
    - Task в терминологии YARN теперь называется Worker
    - Является обобщенной реализацией JobTracker

NODE MANAGER

- Работает на каждой ноде
- Следит за работой контейнеров и отправляет отчеты о их статусе в RM
- Был выделен из MapReduce в Hadoop 2.x и является обобщенной реализацией TaskTracker

CONTAINER

- Выделяется NM по команде RM инициированной AM с фиксированными ресурсами
- Является средой исполнения исполняемого кода

APPLICATION MASTER

- Запускается в контейнере
- Запрашивает создание дополнительных контейнеров у RM для воркеров
    - Воркер — любой исполняемый код, запускаемый распределенно
- Следит за исполнением задачи и передает результаты

WORKFLOW

1. **Запускаем** YARN приложение на edge или slave узле
2. **RM выделяет** ресурсы на кластере для AM и сообщает об этом соответствующей NM
3. **NM создает контейнер** в который загружается исполняемый код, после чего производится его запуск
4. **AM** может запросить у RM **дополнительные ресурсы**, а также указать список файлов для запуска воркеров с учетом data locality
5. **RM выделяет ресурсы** и сообщает об этом NM
6. **NM создают контейнеры** в которые загружается исполняемый код AM
7. **AM** на выделенных контейнерах для воркеров **запускает** нужные участки кода на исполнение

HADOOP >2.x

Благодаря новой обобщенной архитектуре позволяет запускать на кластере не только MapReduce приложения, но и другие. Это повлекло за собой появление новых исполнительных движков:

- Apache Tez
- Apache Spark
- Apache Impala
- Apache Samza
- и др.

РАБОТА YARN

Распределение ресурсов

FIFO

Capacity

Fair (DRF)

ПЛАНИРОВЩИКИ

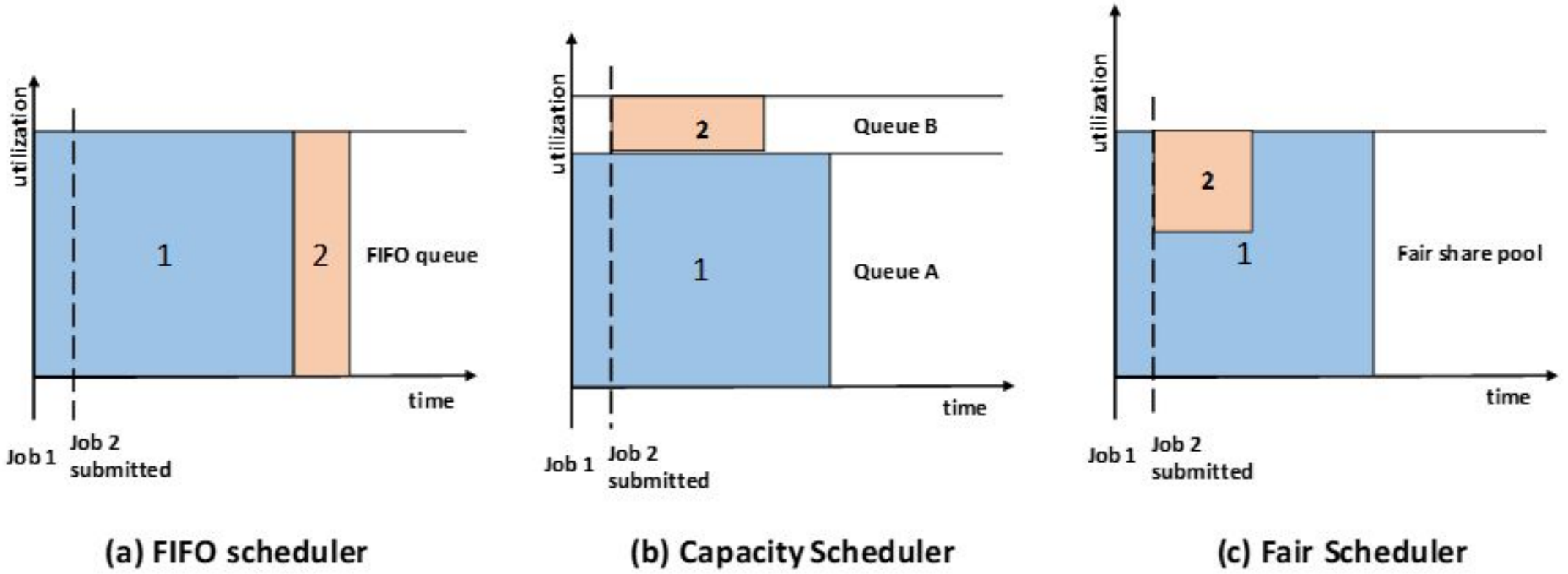


Figure 1: YARN Schedulers' cluster utilization vs. time

Особенности

Speculative execution

Preemption

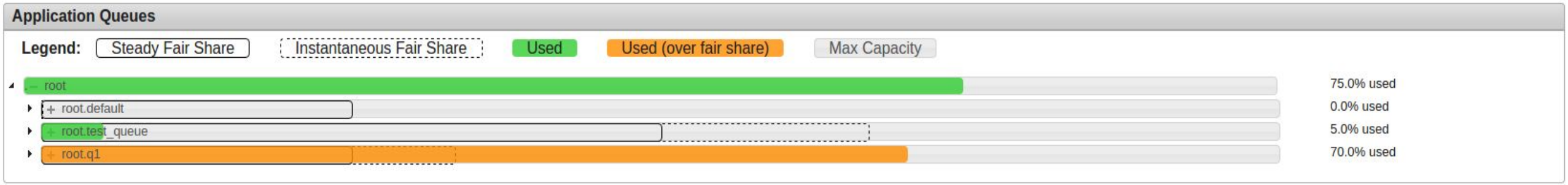
Cluster Metrics

Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running	Memory Used	Memory Total	Memory Reserved	VCores Used	VCores Total	VCores Reserved	Active Nodes	Decommissioned Nodes	Lost Nodes	Unhealthy Nodes	Rebooted Nodes
7	1	3	3	6	30 GB	40 GB	18 GB	15	40	9	5	0	5	0	0

User Metrics for hdfs

Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running	Containers Pending	Containers Reserved	Memory Used	Memory Pending	Memory Reserved	VCores Used	VCores Pending	VCores Reserved
0	0	0	0	0	0	0	0 B	0 B	0 B	0	0	0

Application Queues



Show 20 entries Search:

ID	User	Name	Application Type	Queue	Fair Share	StartTime	FinishTime	State	FinalStatus	Running Containers	Allocated CPU VCores	Allocated Memory MB	Progress	Tracking UI
application_1472739138736_0005	root	org.apache.spark.examples.SparkPi	SPARK	root.q1	6827	Thu Sep 1 16:20:15 +0200 2016	N/A	RUNNING	UNDEFINED	3	9	18432		ApplicationMaster
application_1472739138736_0006	root	org.apache.spark.examples.SparkPi	SPARK	root.q1	6827	Thu Sep 1 16:21:22 +0200 2016	N/A	RUNNING	UNDEFINED	2	5	10240		ApplicationMaster
application_1472739138736_0007	root	org.apache.spark.examples.SparkPi	SPARK	root.test_queue	13654	Thu Sep 1 16:25:12 +0200 2016	N/A	RUNNING	UNDEFINED	1	1	2048		ApplicationMaster
application_1472739138736_0008	root	org.apache.spark.examples.SparkPi	SPARK	root.test_queue	13654	Thu Sep 1 16:25:37 +0200 2016	N/A	ACCEPTED	UNDEFINED	0	0	0		UNASSIGNED

ПЕРЕРЫВ


10:00

PYTHON MAPREDUCE

ПОДНИМАЕМ ЛОКАЛЬНЫЙ КЛАСТЕР

```
docker start -i gbhdp
```

ПОМЕЩАЕМ ДАННЫЕ В HDFS

1. Скачаем датасет ppkm_sentiment, размеченный эмоциональной окраской отзывов индонезийской компании PPKM
2. Поместим его в контейнер:

```
# docker cp archive.zip gbhdp:/home/hduser
```

3. Распакуем:

```
$ unzip archive.zip -d ppkm
$ rm archive.zip
```

4. Копируем директорию в hdfs:

```
$ hdfs dfs -put ppkm /
```

5. Проверим, что файлы в hdfs:

```
$ hdfs dfs -ls /ppkm
```

MAPPER.PY

```
#!/usr/bin/env python
"""mapper.py"""

import sys

# получаем данные из входной строки (STDIN - standard input)
for line in sys.stdin:
    # удаляем пробелы в начале и конце строки
    line = line.strip()
    # делим строку по пробельным символам на слова
    words = line.split()
    # выводим данные в STDOUT (standard output) для продюсера
    for word in words:
        # данные представляют собой пару ключ-значение, разделенные табуляцией,
        # где ключ - слово, а значение всегда 1
         print('%s\t%s' % (word, 1))
```

REDUCER.PY

GeekBrains

```
#!/usr/bin/env python
"""reducer.py"""

from operator import itemgetter
import sys

current_word = None
current_count = 0
word = None
```

REDUCER.PY

```
# получаем строку со стандартного ввода STDIN
for line in sys.stdin:
    # удаляем пробелы в начале и конце строки
    line = line.strip()

    # парсим данные, полученные из mapper.py
    word, count = line.split('\t', 1)

     # приводим тип count к int
    try:
        count = int(count)
    except ValueError:
        # count не является числом, поэтому
        # игнорируем строку
        continue

    # условие написано с учетом того, что Hadoop всегда сортирует
    # данные по ключу перед передачей на вход редьюсера
    if current_word == word:
        current_count += count
    else:
        if current_word:
            # пишем результат в STDOUT
            print('%s\t%s' % (current_word, current_count))
        current_count = count
        current_word = word
```

REDUCER.PY


```
# не забываем вывести последнее слово, так как предыдущий вывод
# работал только при смене слова
if current_word == word:
    print '%s\t%s' % (current_word, current_count)
```

НЕОБХОДИМЫЕ ПРАВА

Даем скриптам mapper.py и reducer.py права на исполнение:

```
chmod +x mapper.py
chmod +x reducer.py
```

Проверяем:

```
ls -la mapper.py reducer.py
```

ЛОКАЛЬНЫЙ ТЕСТ

Тест mapper.py:

```
$ echo "foo foo quux labs foo bar quux" | /home/hduser/mapper.py
foo     1
foo     1
quux    1
labs    1
foo     1
bar     1
quux    1
```

Тест reducer.py:

```
$ echo "foo foo quux labs foo bar quux" | /home/hduser/mapper.py | sort -k1,1 | /home/hduser/reducer.py
bar     1
foo     3
labs    1
quux    2
```

ЗАПУСК НА КЛАСТЕРЕ

```
$ hadoop jar ./hadoop/share/hadoop/tools/lib/hadoop-streaming-2.10.1.jar \
-D mapred.reduce.tasks=2 \
-file /home/hduser/mapper.py  -mapper /home/hduser/mapper.py \
-file /home/hduser/reducer.py -reducer /home/hduser/reducer.py \
-input /ppkm                  -output /ppkm.out
```

РЕЗУЛЬТАТ НА КЛАСТЕРЕ

```
hduser@gbhdp:~$ hdfs dfs -ls /ppkm.out
Found 2 items
-rw-r--r--   1 hduser supergroup          0 2021-08-08 21:30 /ppkm.out/_SUCCESS
-rw-r--r--   1 hduser supergroup      26251 2021-08-08 21:30 /ppkm.out/part-00000
hduser@gbhdp:~$ hdfs dfs -cat /ppkm.out/*
!       1
""      2
""Kalian        1
""Pilihan       1
""peraturan     1
#Adaptasikebiasaambaru  1
#Anggaran"      1
#AniesBaswedan  1
#AyoJogoBlitar  1
#BambangIsmadi  1
#Berita 3
#BeritaJakarta  2
#BeritaTerkini  3
#Beritaonline   1
#BersatuMelawanCovid19  2
#Bulungan       1
```

ОСТАНОВКА ЛОКАЛЬНОГО КЛАСТЕРА

```
exit
```

ПРАКТИЧЕСКОЕ ЗАДАНИЕ

ПРАКТИЧЕСКОЕ ЗАДАНИЕ

1. Может ли стадия Reduce начаться до завершения стадии Map? Почему?
2. Приведите пример Map only и Reduce задачи.
3. Разверните кластер hadoop, соберите WordCount приложение, запустите на датасете ppkm_sentiment *и выведите 10 самых редких слов**
4. Измените маппер в WordCount так, чтобы он удалял знаки препинания и приводил все слова к единому регистру
5. **У вас есть два датасета с одинаковыми ключами. Вам нужно их объединить, суммировав значения с одинаковыми ключами. Как это сделать в MapReduce?*
6. **На кластере лежит датасет, в котором ключами является id сотрудника и дата, а значением размер выплаты. Руководитель поставил задачу рассчитать среднюю сумму выплат по каждому сотруднику за последний месяц. В маппере вы отфильтровали старые записи и отдали ключ-значение вида: id-money. А в редьюсере суммировали все входящие числа и поделили результат на их количество. Но вам в голову пришла идея оптимизировать расчет, поставив этот же редьюсер и в качестве комбинатора, тем самым уменьшив шафл данных. Можете ли вы так сделать? Если да, то где можно было допустить ошибку? Если нет, то что должно быть на выходе комбинатора?*

Спасибо!

Каждый день вы становитесь лучше :)