Large Scale Machine Learning (LSML)

О курсе

Темы

1. Hadoop & Apache Spark
2. Онлайн Обучение и линейные модели, vopal wabbit
3. Градиентный бустинг для больших данных
4. Рекомендательные системы, их работа поверх больших данных
5. Трюки с хэшированием данных
6. Работа биг даты на примере больших компаний
7. Работа нейросетей поверх больших данных, их оптимизации

Формула Оценки

$$О_{итоговая} = 0.5 * \frac{(О_{МДЗ1} + О_{МДЗ2} + О_{МДЗ3} + О_{МДЗ4})}{4} + 0.5 * О_{БДЗ}$$

LSML #1
Введение в Hadoop и Map Reduce

Задача Word Count

- У нас есть огромный текстовый файл
- Хотим посчитать частоты слов в тексте

Как посчитать?

1. Разделим на кусочки
2. Обработаем каждый на отдельном ядре
3. Сложим счетчики слов

Наивное решение в питоне:

dict = {} – наш словарь

```
for word in text:
        dict[word]++
```

Наивное решение в питоне:

```
dict = {} – наш словарь
for word in text:
        dict[word]++
```

Какие проблемы есть в данном решении?

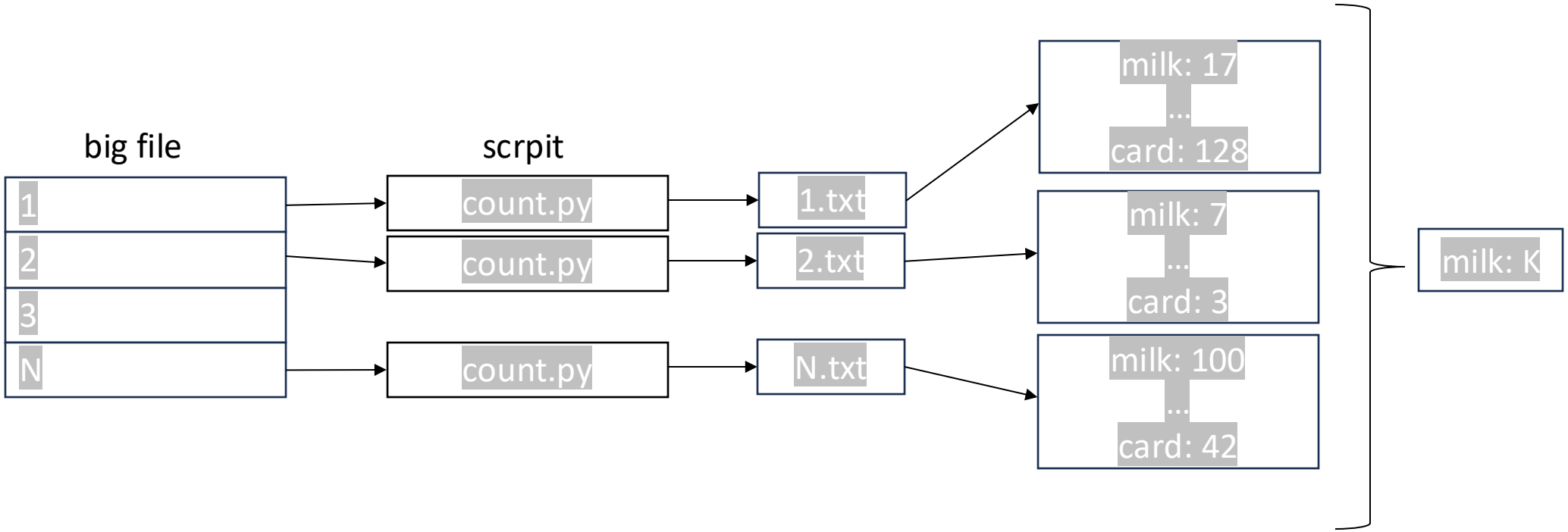
Наивное решение в питоне:

```
dict = {} – наш словарь
for word in text:
        dict[word]++
```

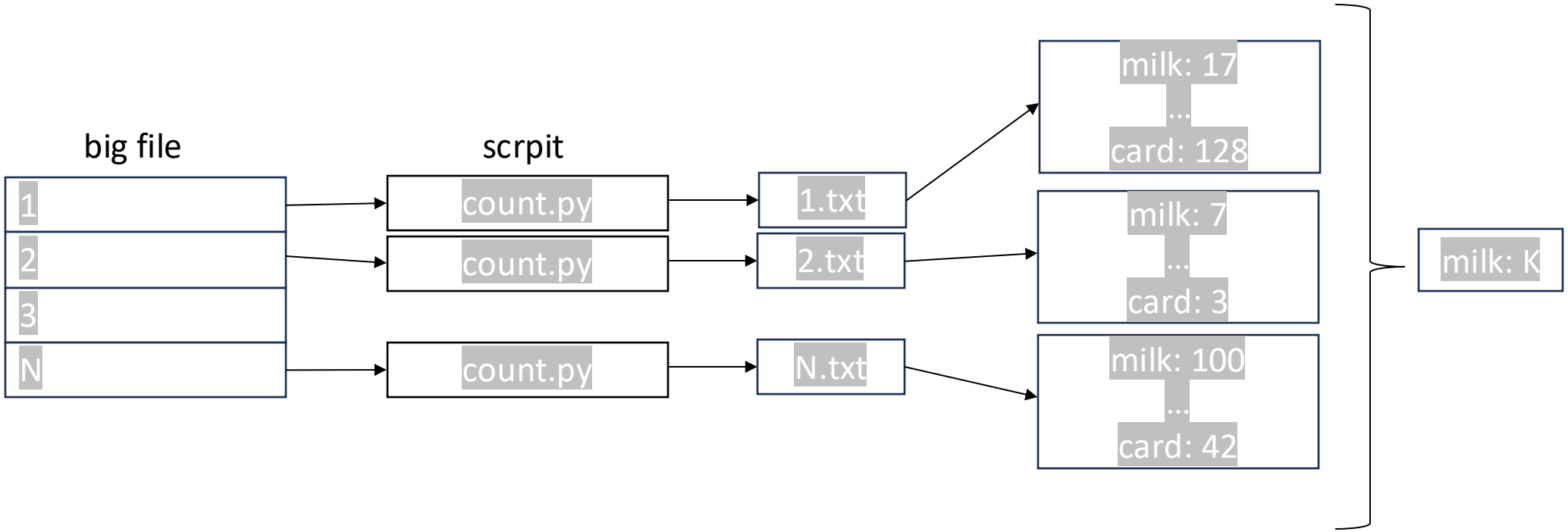
Какие проблемы есть в данном решении?
dict – переменная в RAM и может получиться большого размера по мере обработки файла

Как можно улучшить эту программу?

Визуализация решения

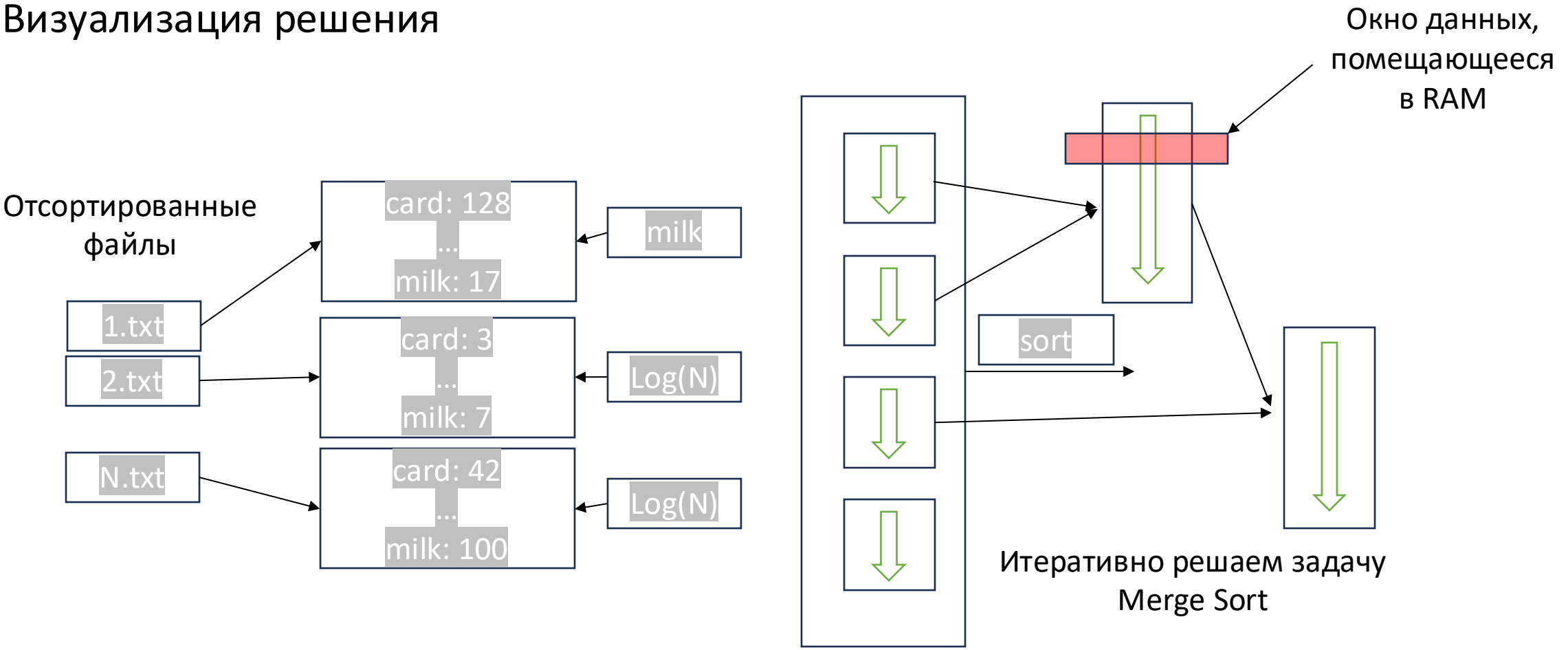


Визуализация решения

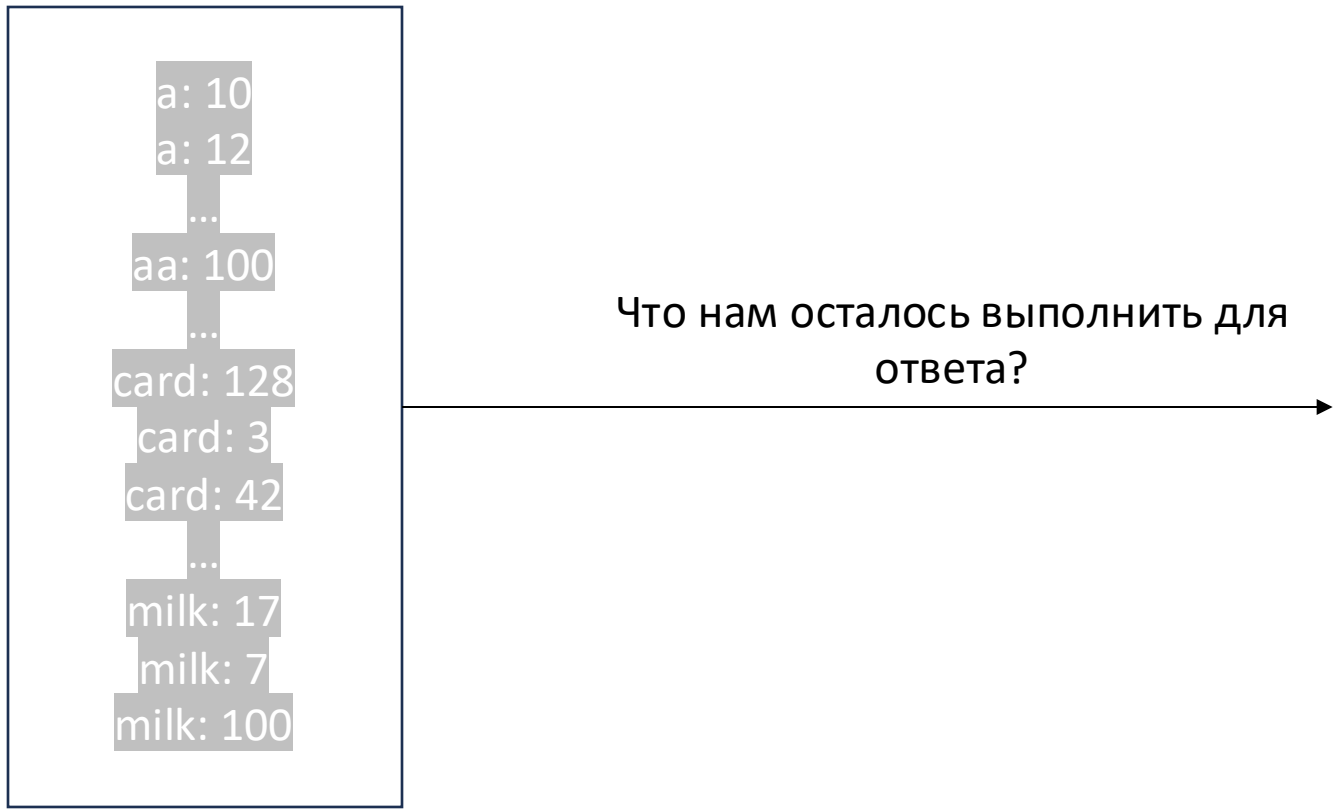


Хотим быстро искать ключи-слова в файлах N.txt

Визуализация решения

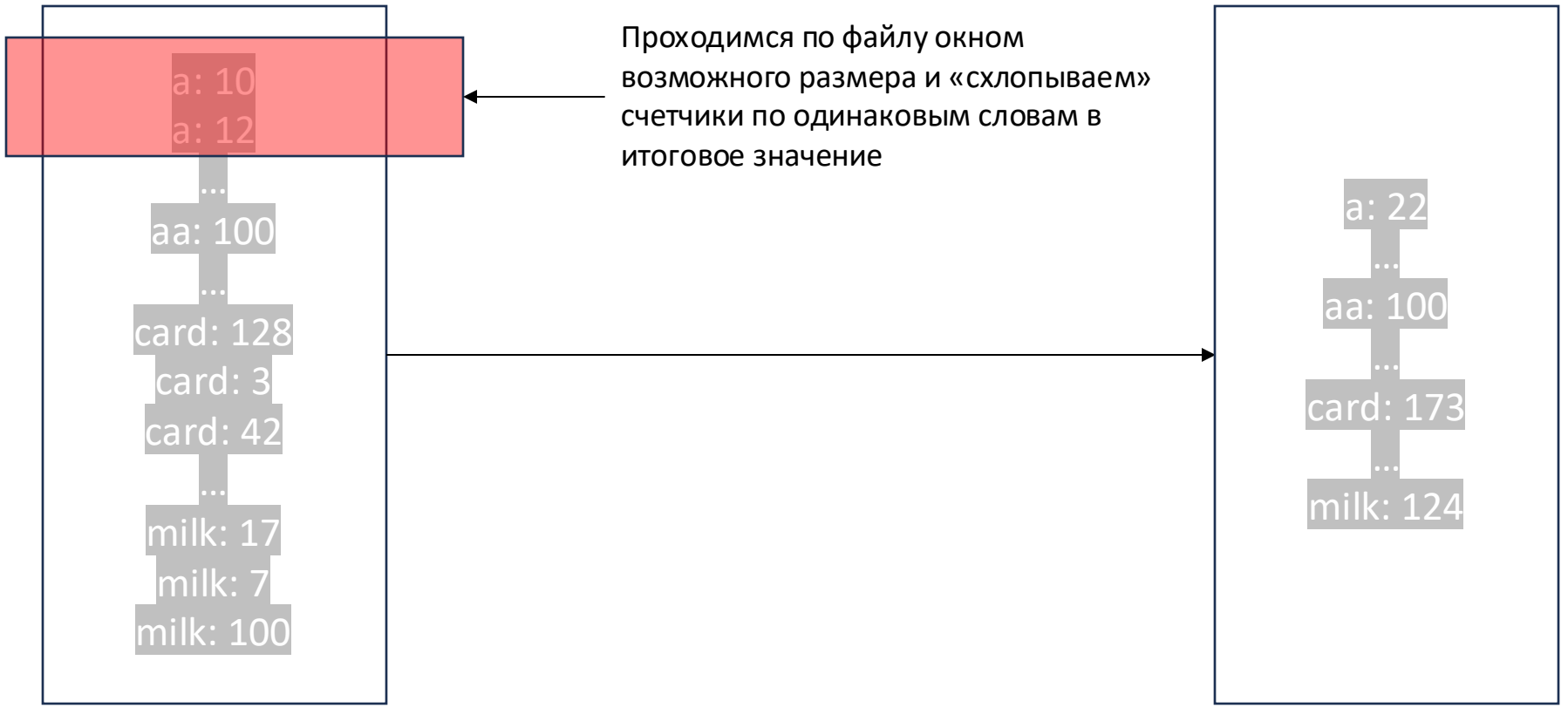


Конечный отсортированный файл по ключам



Что нам осталось выполнить для ответа?

Конечный отсортированный файл по ключам



Google Big Table / Google File System

The Google File System

Sanjay Ghemawat, Howard Gobioff, and Shun-Tak Leung
Google*

ABSTRACT

We have designed and implemented the Google File System, a scalable distributed file system for large distributed data-intensive applications. It provides fault tolerance while running on inexpensive commodity hardware, and it delivers high aggregate performance to a large number of clients.

While sharing many of the same goals as previous distributed file systems, our design has been driven by observations of our application workloads and technological environment, both current and anticipated, that reflect a marked departure from some earlier file system assumptions. This has led us to reexamine traditional choices and explore rad-

1. INTRODUCTION

We have designed and implemented the Google File System (GFS) to meet the rapidly growing demands of Google's data processing needs. GFS shares many of the same goals as previous distributed file systems such as performance, scalability, reliability, and availability. However, its design has been driven by key observations of our application workloads and technological environment, both current and anticipated, that reflect a marked departure from some earlier file system design assumptions. We have reexamined traditional choices and explored radically different points in the design space.

https://static.googleusercontent.com/media/research.google.com/en//archive/gfs-sosp2003.pdf

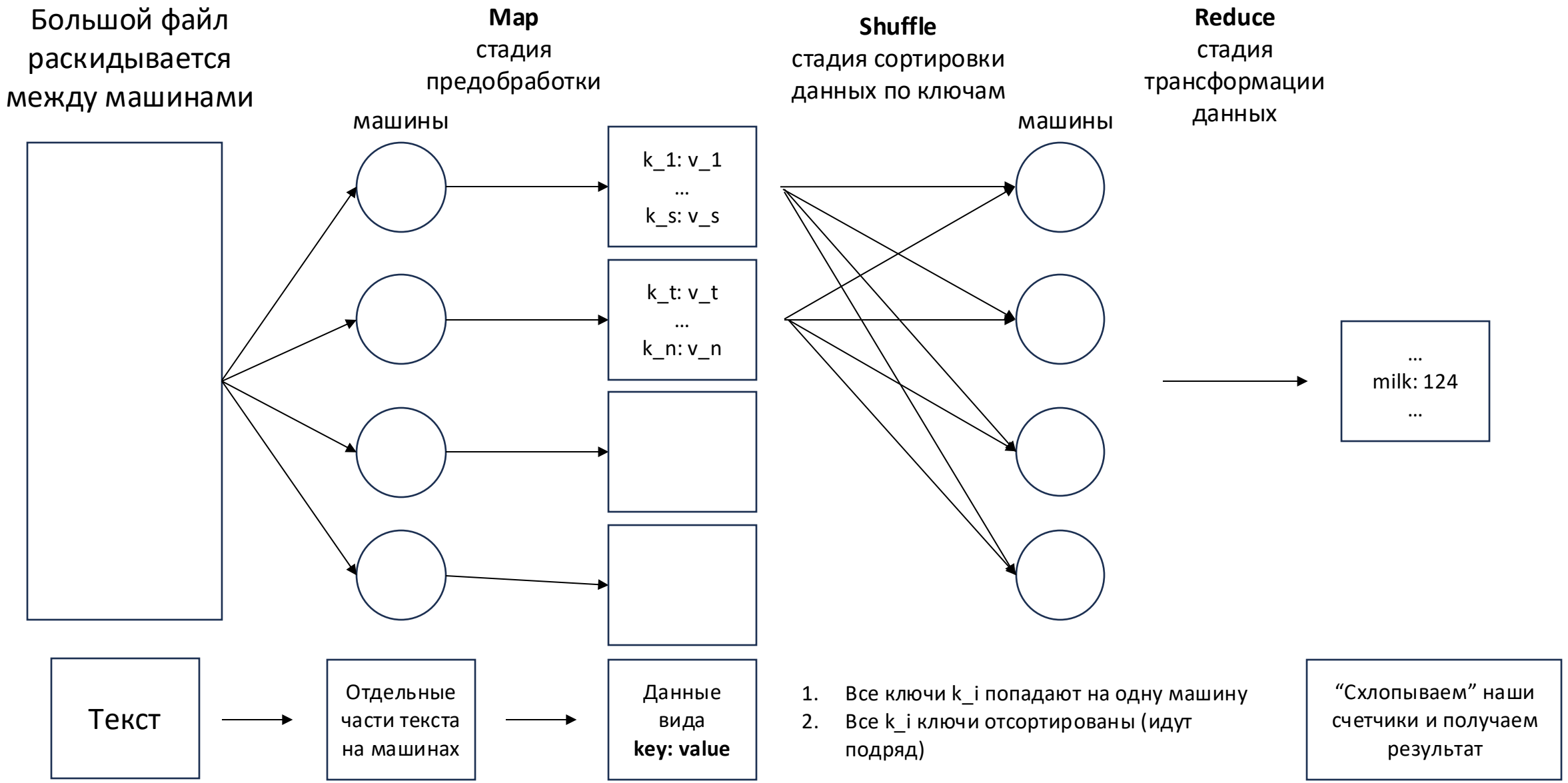
Полученный нами алгоритм обработки большого файла и есть Map Reduce.

Он состоит из нескольких стадий с простыми действиями внутри.

Как итог – через этот паттерн работы с данными можно обрабатывать более сложные запросы к данным:

- Время нахождения пользователя на сайте
- Кол-во посещенных пользователем сайтов определенного класса
- Подсчет стат значений на большом числе данных (мат. ожидание, дисперсия), если нужные формулы возможно перевести в итеративный формат

Map Reduce


Большой файл раскидывается между машинами
Map
стадия предобработки
машины
k_1: v_1
…
k_s: v_s
k_t: v_t
…
k_n: v_n
Shuffle
стадия сортировки данных по ключам
машины
Reduce
стадия трансформации данных
…
milk: 124
…
Текст
Отдельные части текста на машинах
Данные вида
key: value
1. Все ключи k_i попадают на одну машину
2. Все k_i ключи отсортированы (идут подряд)
“Схлопываем” наши счетчики и получаем результат

Заметки по использованию

Map – предобработка, реализованная фами

Shuffle – процесс, на который пользователь может влиять только через опр набор условий, влияющих на сортировку. Реализована не юзером

Reduce – трансформация, реализованная вами

Как итог - эту схему работы с данными удобно масштабировать. Большой объем данных сможем обрабатывать за адекватное время

Процесс обработки данных можно превратить в несколько последовательных M-R задач, где данные перетекают из одной задачи в последующие

Парадигма Map-Reduce на примере Word Count

Шаг Map:

(K1, V1) → List(K2, V2)
(#строки, “Deer Bear”) → [(“Deer”, 1), (“Bear”, 1)]

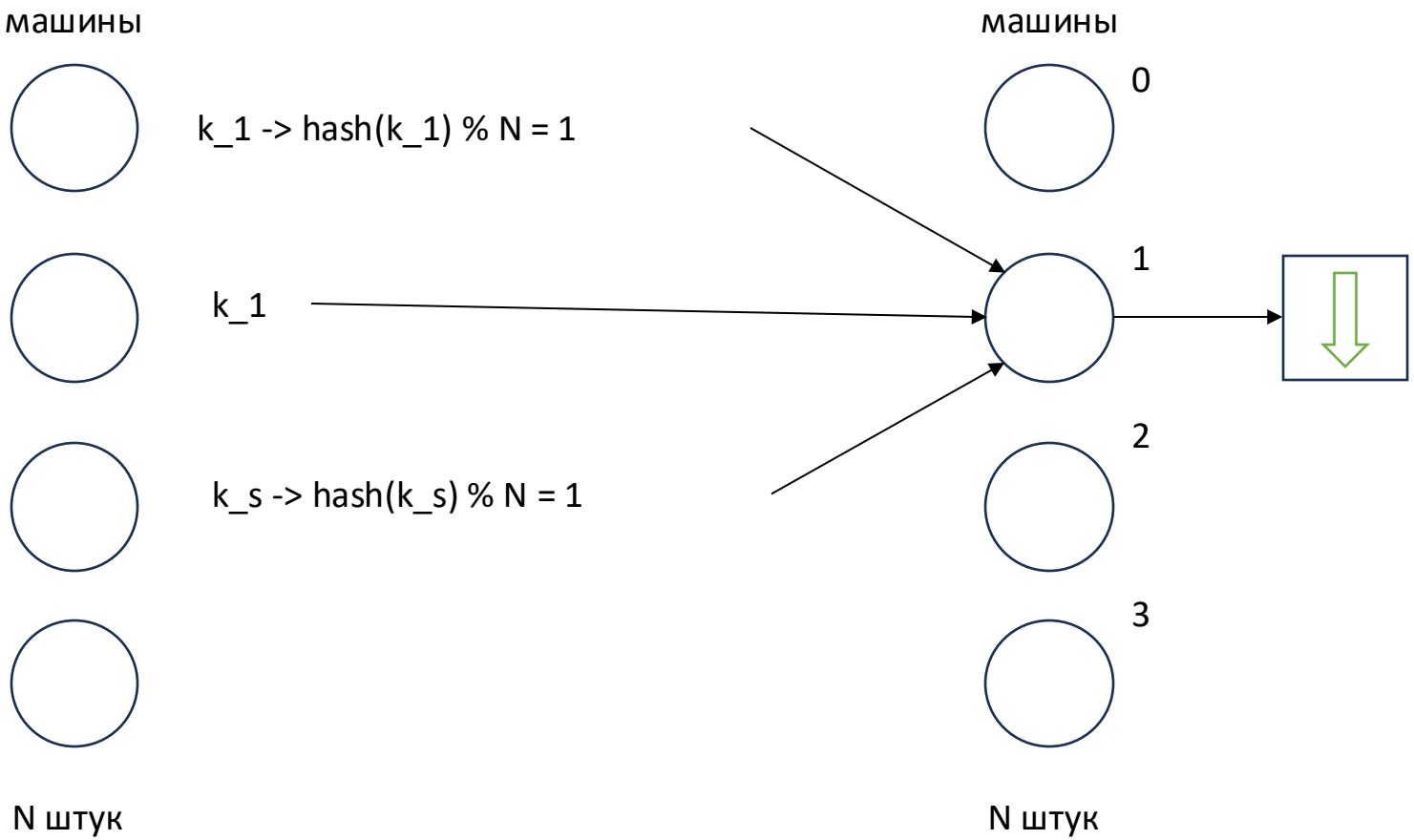
Шаг Shuffle (или Sort):

Shuffle делит данные по **hash(key) % N** на **N** частей
Sort сортирует данные по **key** и делит на **N** частей (по границам **key**)

Шаг Reduce:

(K1, (V1, V2, …)) → List(K3, V3)
(“Bear”, (1, 1)) → [(“Bear”, 2)]

Про shuffle и hash



1. Все ключи k_i попадают на одну машину
2. Все k_i ключи отсортированы (идут подряд)

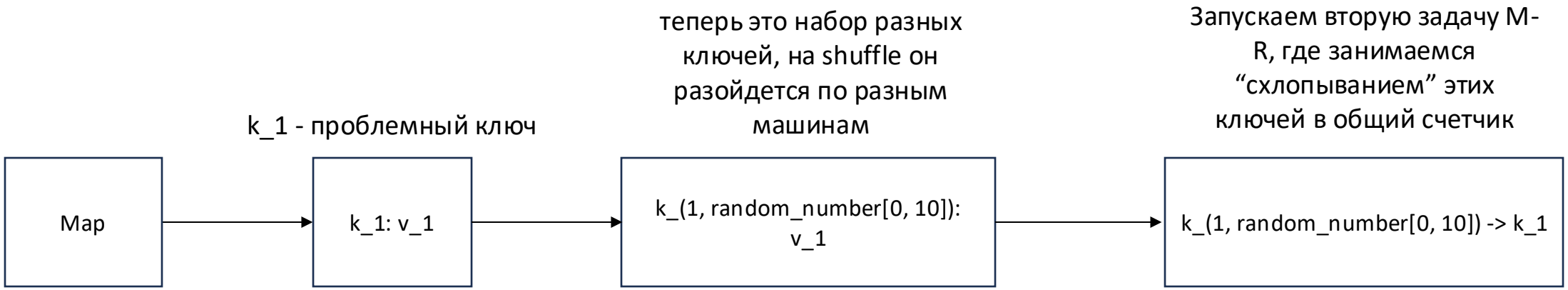
Про shuffle и hash. Проблемы, которые возникают на этом этапе

Что делать, если ключ k_i собрался со всех машин на одну после сортировки, и теперь под его хранение не хватает места?

Про shuffle и hash. Проблемы, которые возникают на этом этапе

Что делать, если ключ k_i собрался со всех машин на одну после сортировки, и теперь под его хранение не хватает места?

Ответ - “усложнить” ключ и решать задачу через 2 таски M-R



Hadoop и его составляющие

Hadoop - проект Apache для распределенных вычислений

Hadoop YARN - планировщик задач и система для менеджмента ресурсов кластера
Hadoop Distributed File System (HDFS) - распределенная файловая система
Hadoop Mapreduce - система для вычислений в парадигме M-R, движок обработки запросов к данным

HDFS: распределенная файловая система

Способ хранения данных на кластере, имеет следующие важные свойства:

1. Произвольная вместимость - возможность подключения новых машин в систему

HDFS: распределенная файловая система

Способ хранения данных на кластере, имеет следующие важные свойства:

1. Произвольная вместимость
2. Устойчивость к сбоям

Для пункта 2 разберем задачу:

p = 0.1% - вер-ть, что одна машина выйдет из строя в течение дня (данные на ней будут потеряны)

? - какова вер-ть, что хотя бы 1 из 1000 машин выйдет из строя?

HDFS: распределенная файловая система

Способ хранения данных на кластере, имеет следующие важные свойства:

1. Произвольная вместимость
2. Устойчивость к сбоям

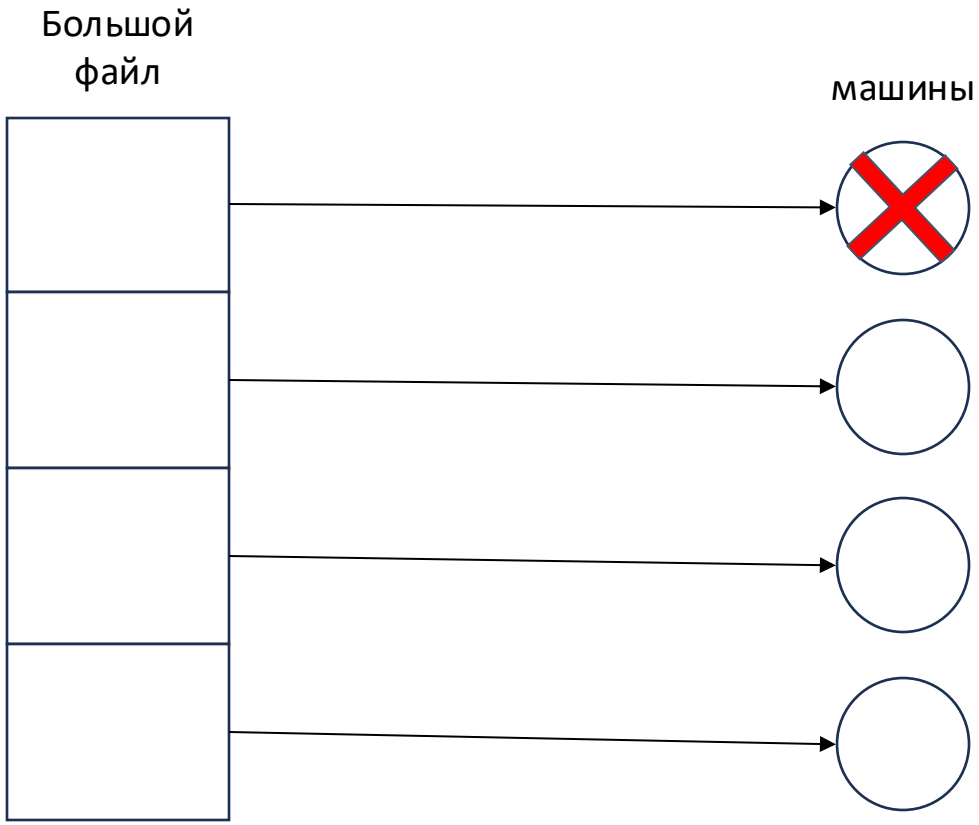
Для пункта 2 разберем задачу:
p = 0.1% - вер-ть, что одна машина выйдет из строя в течение дня (данные на ней будут потеряны)
? - какова вер-ть, что хотя бы 1 из 1000 машин выйдет из строя?

$1 - (1 - p)^{1000} \approx 0.63$

HDFS: распределенная файловая система

Способ хранения данных на кластере, имеет следующие важные свойства:

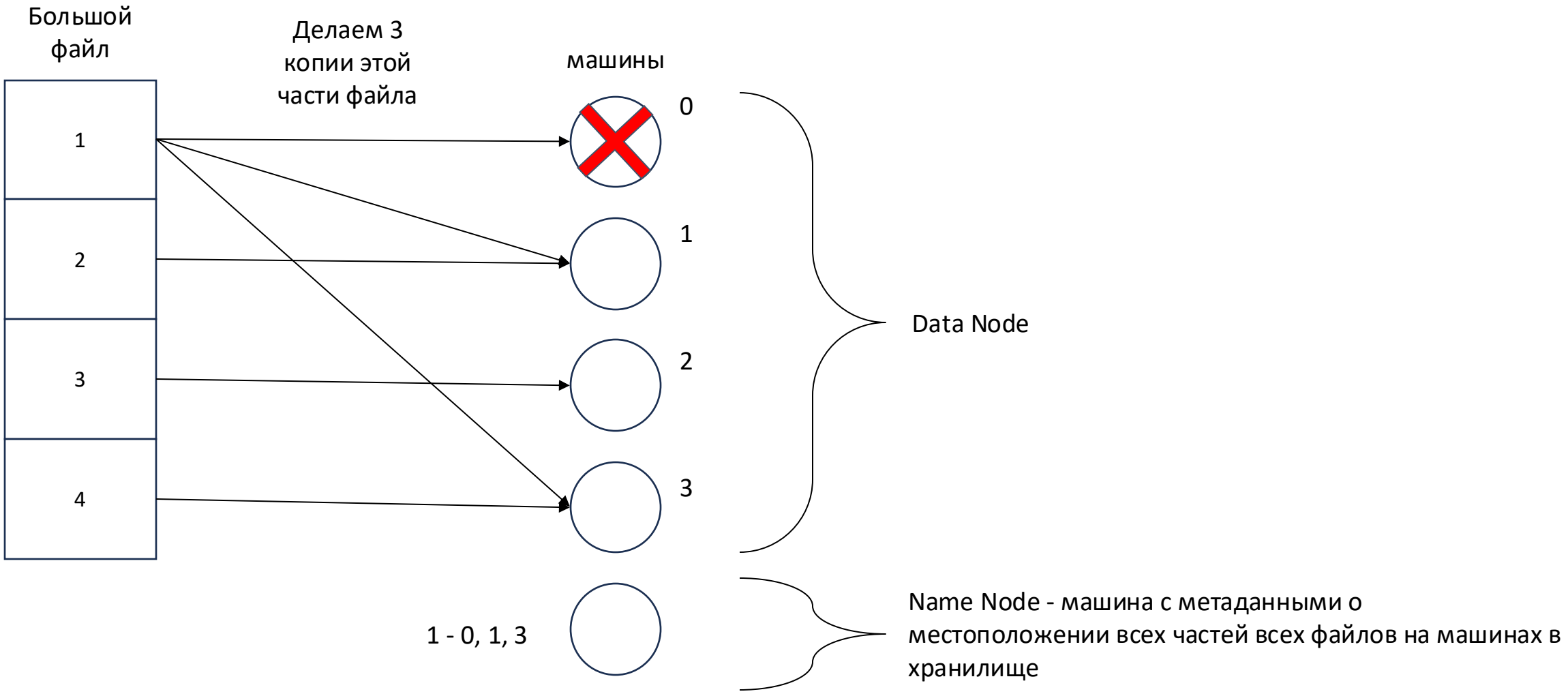
1. Произвольная вместимость
2. Устойчивость к сбоям - **репликация** частей файла



HDFS: распределенная файловая система

Способ хранения данных на кластере, имеет следующие важные свойства:

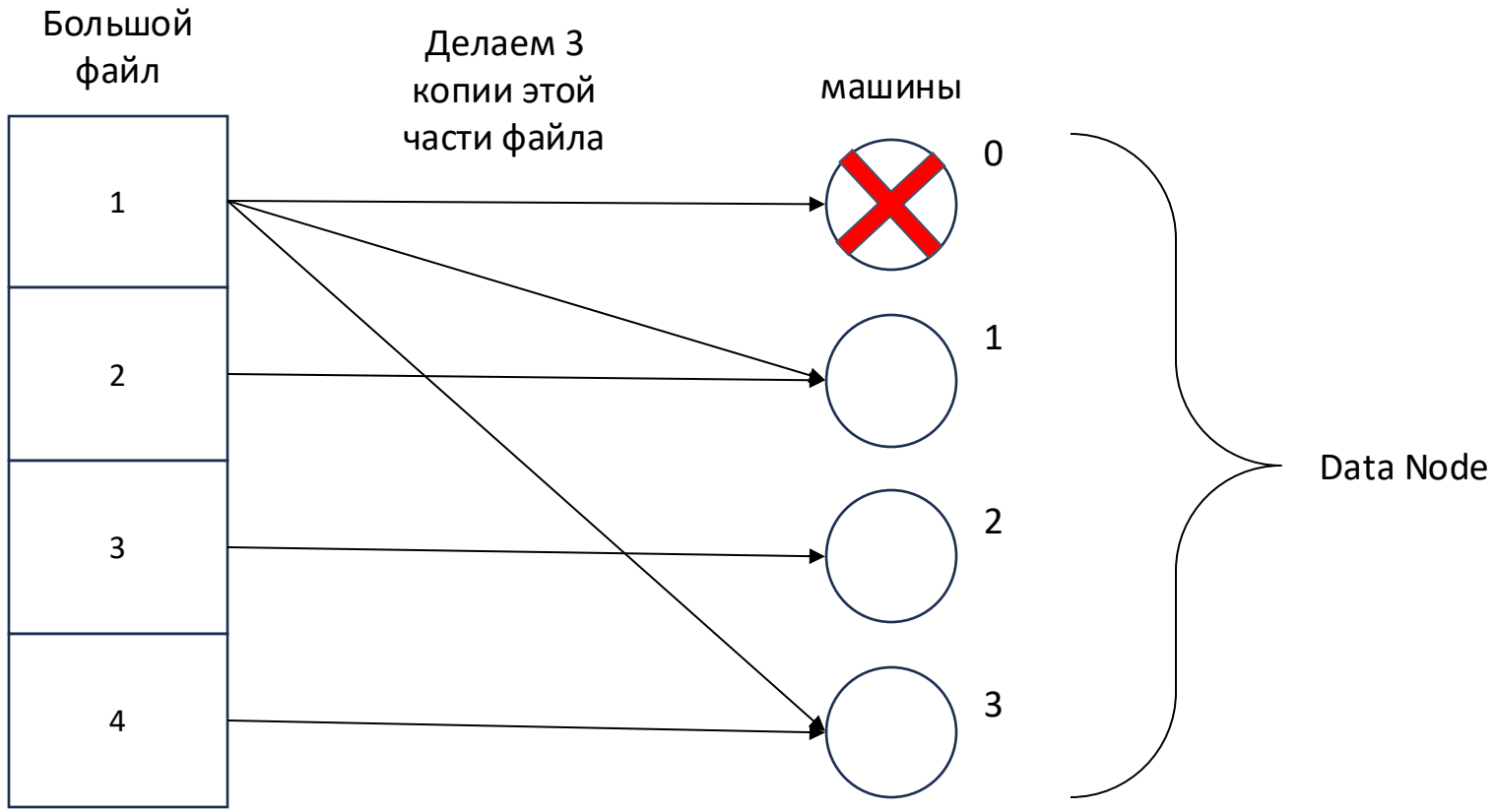
1. Произвольная вместимость
2. Устойчивость к сбоям - **репликация** частей файла



HDFS: распределенная файловая система

Способ хранения данных на кластере, имеет следующие важные свойства:

1. Произвольная вместимость
2. Устойчивость к сбоям - **репликация** частей файла



Файлы в хранилище бьются на блоки, всегда одинакового размера обычно это 128/256 Мб

1 - 0, 1, 3

Name Node - машина с метаданными о местоположении всех частей всех файлов на машинах в хранилище

Основная проблема работы с MapReduce - много чтений и записи на диски, не используется RAM