系统设计
MapReduce
（九章网站下载最新课件）

本节主讲人：北丐老师

今日课程大纲

- Map Reduce Problems
  - 多台机器并行处理数据
  - Count Word Frequency
  - Build Inverted Index

Map Reduce

Why Map Reduce?

Distributed System is built for fast computing

大数据职位面试敲门砖

学会MapReduce可以找大数据工作

Interviewer: Count the word frequency of a web page?

Google 面试真题

http://www.lintcode.com/en/problem/word-count/

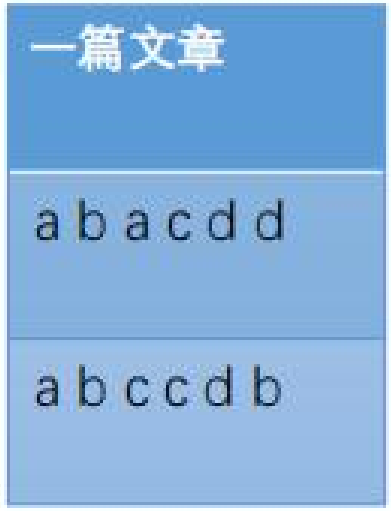
http://www.jiuzhang.com/solutions/word-count/

常见土方法一 For循环

方法一 For循环

伪代码

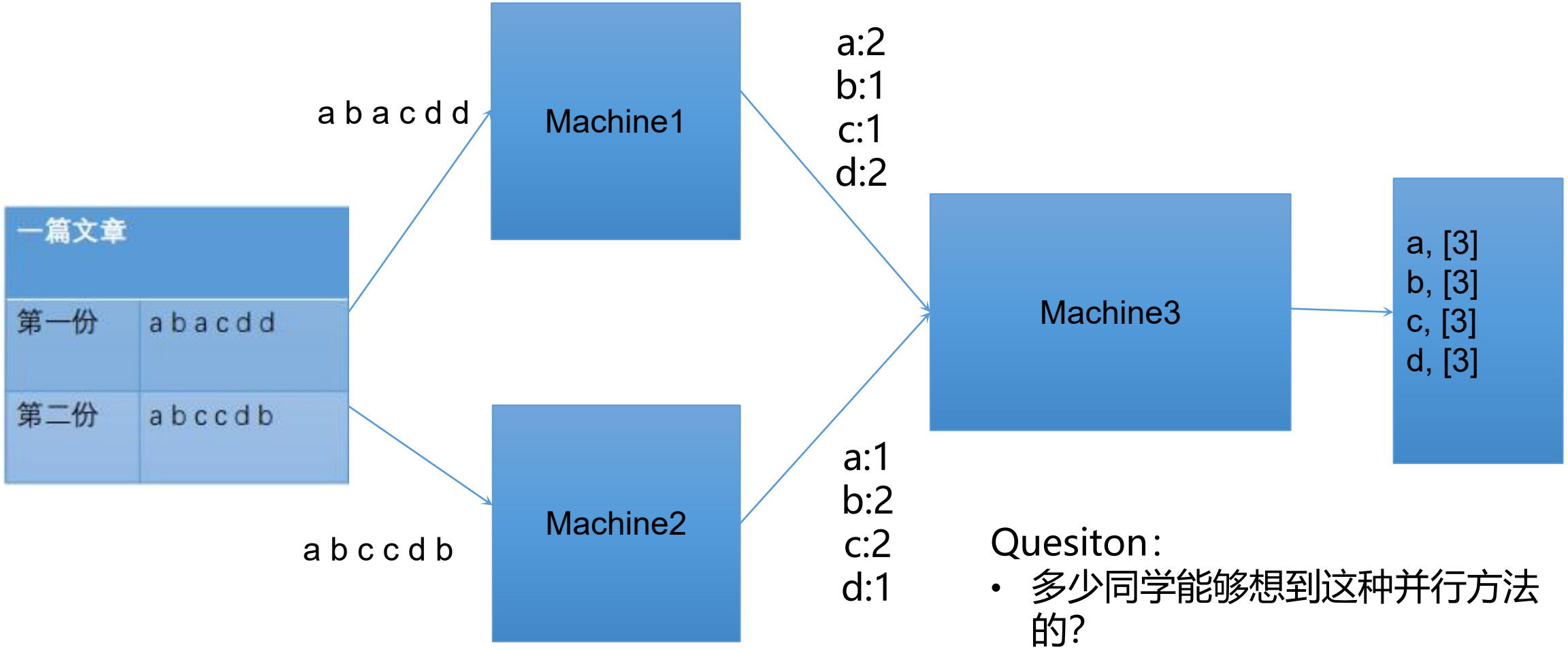
- HashMap<String,int> wordcount;
- for each word in webpage :
  - wordcount[word]++



- Question?
  - 多少同学能够想到这种方法?
  - 有什么优缺点?
    - 优点：简单　　缺点：只有一台机器——慢、内存大小受限
  - 如果你有多台机器呢?

常见土方法二 多台机器For循环

方法二 多台机器For循环

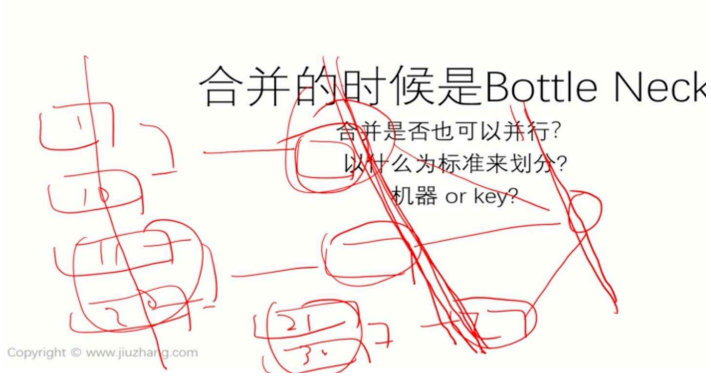


Quesiton：

- 多少同学能够想到这种并行方法的?
- 大家觉得这种方法有啥问题?

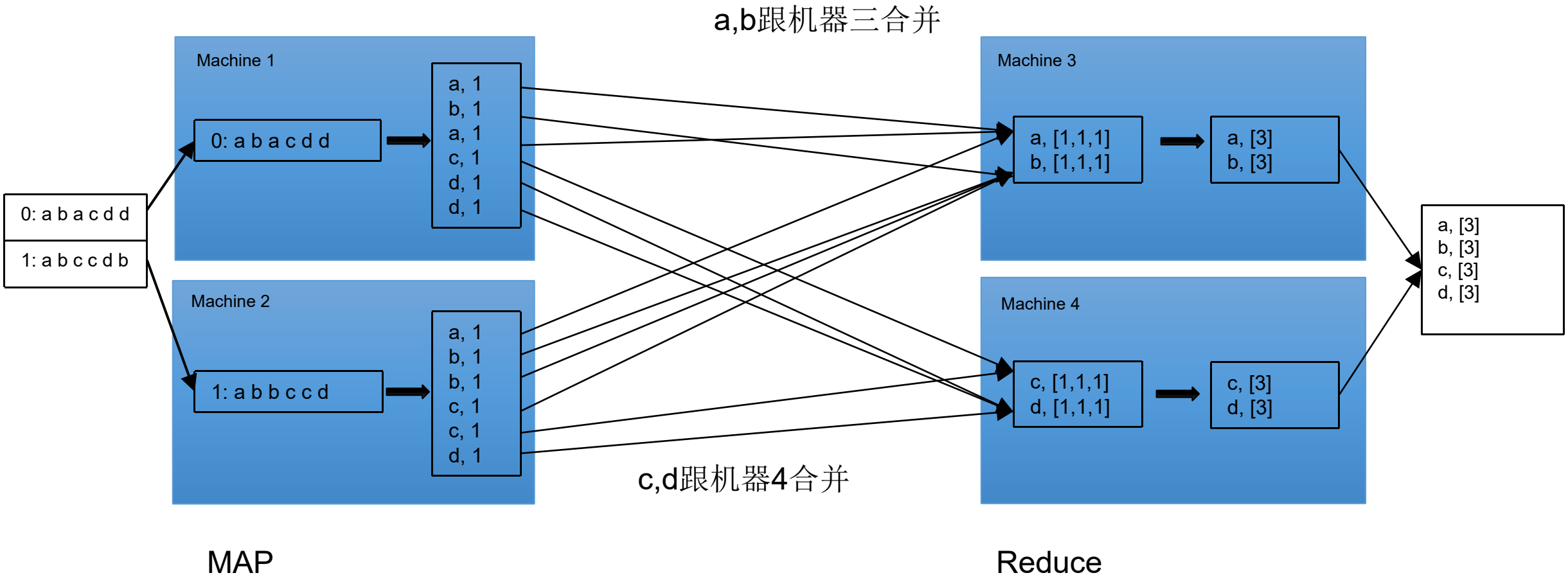
合并的时候是Bottle Neck

合并是否也可以并行?
以什么标准来划分?
机器 or key?



方法三 多台机器Map Reduce

方法三：Map Reduce



多台机器Map Reduce

Map

- 机器1，2 只负责把文章拆分为一个一个的单词

Reduce

- 机器3，4各负责一部分word的合并

Map Reduce

Map
把文章拆分单词的过程

Reduce
把单词次数合并在一起的过程

存在的问题

谁来负责把文章拆分为一小段一小段?

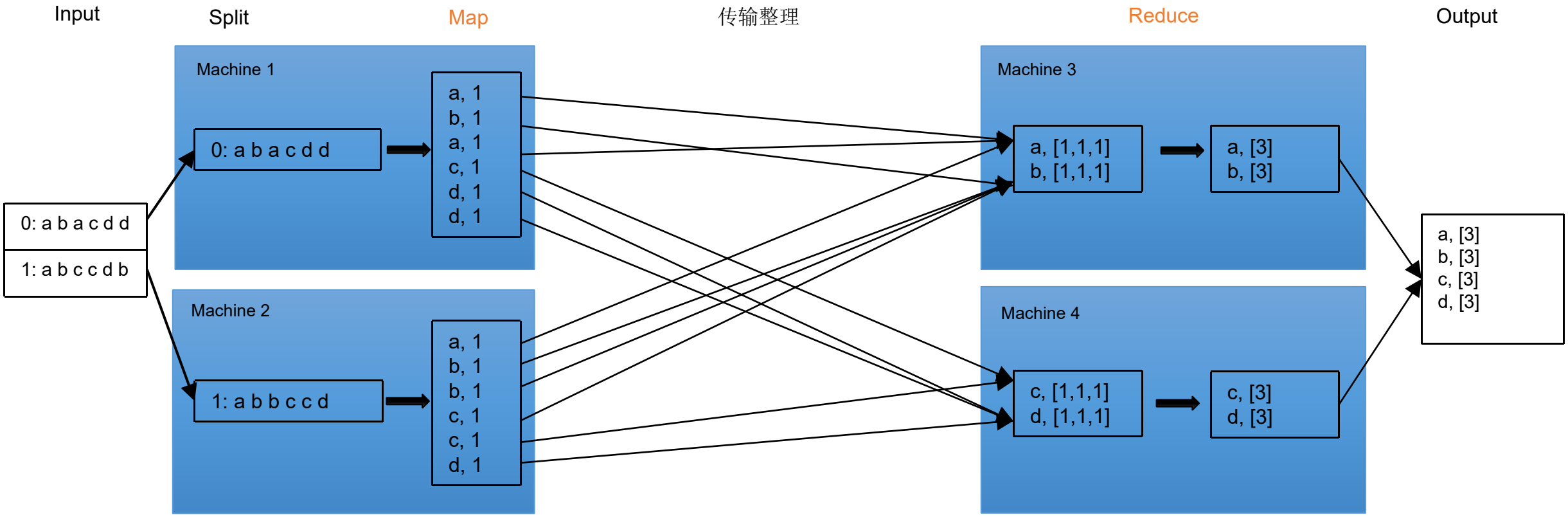
中间传输整理谁来负责？比如怎么知道把a放在机器3还是机器4?

依靠Map Reduce的框架实现

Map Reduce Steps

- Map Reduce 是一套实现分布式运算的框架
- Step1 Input
- Step2 Split
- Step3 Map
- Step4 传输整理
- Step5 Reduce
- Step6 Output

Word Count



Map为什么不做Aggregation?

我们要实现什么呢？

Map 函数 和 Reduce 函数

Map Reduce Steps

- Map Reduce 是一套实现分布式运算的框架

- Step1 Input
- Step2 Split
- Step3 Map 实现怎么把文章切分成单词
- Step4 传输整理
- Step5 Reduce 实现怎么把单词统一在一起
- Step6 Output

- 所以MapReduce帮我们把框架大部分实现好，我们只用实现Map Reduce解决逻辑计算的问题。

```
public class WordCount {

    public static class Map {
        public void map(String key, String value, OutputCollector<String,
            Integer> output) {
            // Write your code here
            // Output the results into output buffer.
            // Ps. output.collect(String key, int value);
            StringTokenizer tokenizer = new StringTokenizer(value);
            while (tokenizer.hasMoreTokens()) {
                String outputkey = tokenizer.nextToken();
                output.collect(outputkey, 1);
            }
        }
    }

    public static class Reduce {
        public void reduce(String key, Iterator<Integer> values,
                           OutputCollector<String, Integer> output) {
            // Write your code here
            // Output the results into output buffer.
            // Ps. output.collect(String key, int value);
            int sum = 0;
            while (values.hasNext()) {
                sum += values.next();
            }
            output.collect(key, sum);
        }
    }
```

Accepted

100%

100% 数据通过测试　　总耗时 505 ms

您的提交打败了 99.00% 的提交!

查看详情

用 Chrome 插件看看更好的参考答案

和你的朋友们分享你的AC

输入测试数据 (每行一个参数)

chunk1: "Google Bye GoodBye Hadopp lintcode"
chunk2: "lintcode Google code"
chunk3: "Bye Bye Google"

Map Reduce 函数接口是什么？

他们的输入和输出必须是Key Value 形式

Map 输入: key:文章存储地址, Value: 文章内容

Reduce 输入：key:map输出的key, value: map输出的value

Google面试真题实战

http://www.lintcode.com/en/problem/word-count/

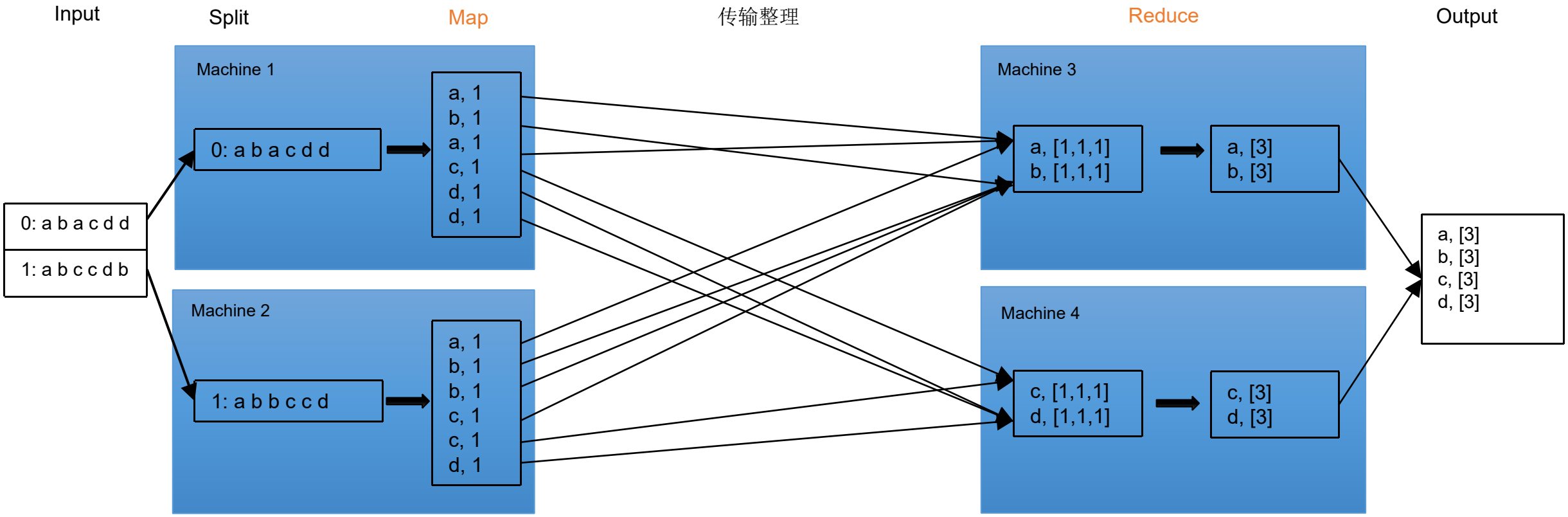
http://www.jiuzhang.com/solutions/word-count/

Map Reduce Steps

- Map Reduce 是一套实现分布式运算的框架
- Step1 Input 设定好输入文件
- Step2 Split 系统帮我们把文件尽量平分到每个机器
- Step3 Map 实现代码
- Step4 传输整理 系统帮我们整理
- Step5 Reduce 实现代码
- Step6 Output 设定输出文件

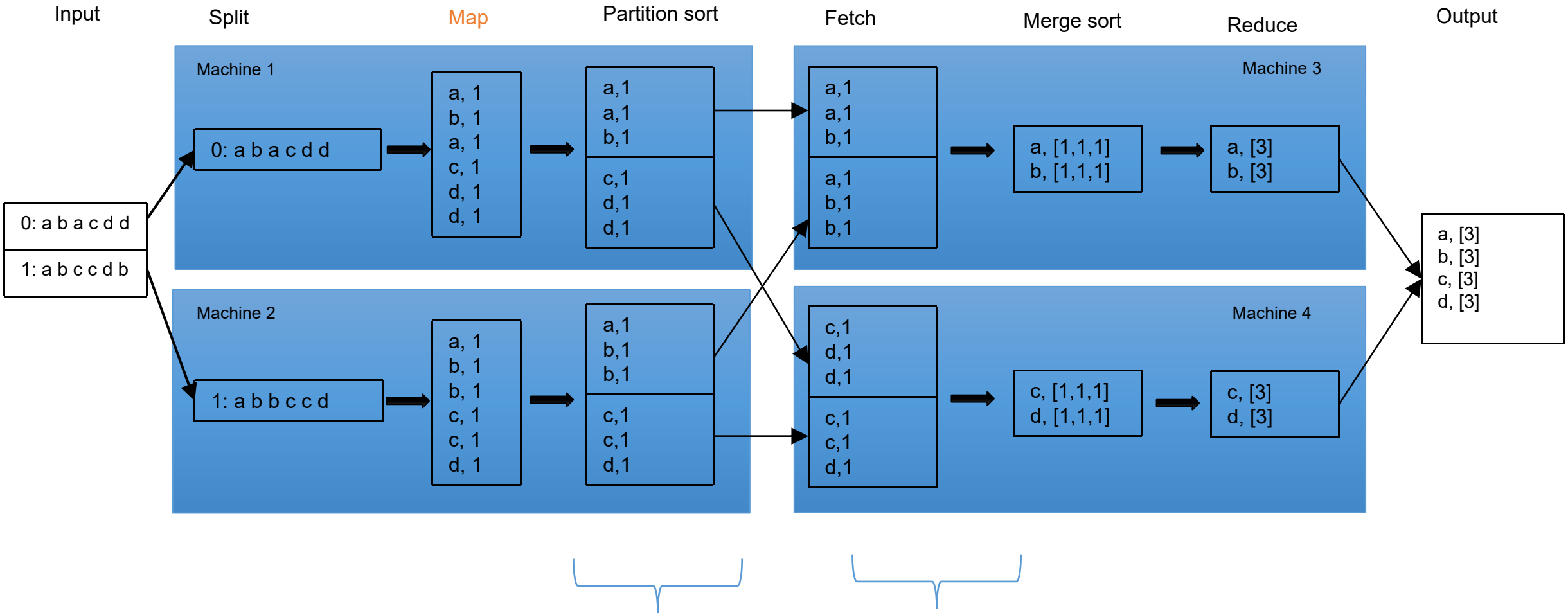
“传输整理”详细操作

Word Count



要你设计这一步你会怎么设计?

“传输整理”详细操作



Map Reduce Steps

- Map Reduce 是一套实现分布式运算的框架

- Step1 Input
- Step2 Split
- Step3 Map 实现怎么把文章切分成单词
- Step4 Partition sort
- Step5 Fetch + Merge Sort
- Step6 Reduce 实现怎么把单词统一在一起
- Step7 Output

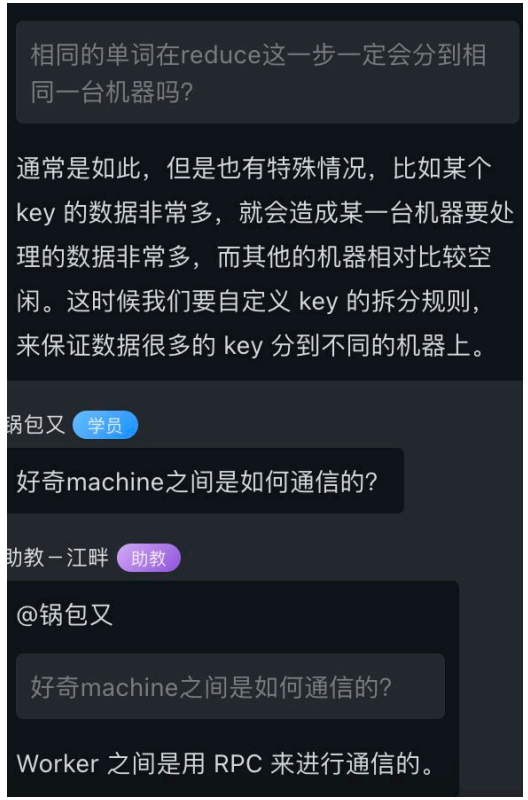
- 所以MapReduce帮我们把框架大部分实现好，我们只用实现Map Reduce解决逻辑计算的问题。

Map Reduce Steps

- Question1?
- Map 多少台机器？ Reduce 多少台机器？
  - 全由自己决定。一般1000map，1000reduce规模

- Question2? （加分）
- 机器越多就越好么？
  - Advantage:
    - 机器越多，那么每台机器处理的就越少，总处理数据就越快
  - Disadvantage:
    - 启动机器的时间相应也变长了。

- Question3? （加分）
  - 如果不考虑启动时间，Reduce 的机器是越多就一定越快么？
    - Key的数目就是reduce的上限



Break

Apple Interviewer: Build inverted index with MapReduce?

http://www.lintcode.com/en/problem/inverted-index-map-reduce/#

http://www.jiuzhang.com/solutions/inverted-index-map-reduce/

Read More:
Novice/Expert, http://url.cn/fsZ927

Input

0: Deer Bear River
1: Car River
2: Deer Car Bear

Output

Bear: 0,2
Car: 1,2
Deer: 0,2
River: 0,1

```
public class InvertedIndex {

    public static class Map {
        public void map(String _, Document value,
                        OutputCollector<String, Integer> output) {
            // Write your code here
            // Output the results into output buffer.
            // Ps. output.collect(String key, int value);
            StringTokenizer tokenizer = new StringTokenizer(value.content
                );
            while (tokenizer.hasMoreTokens()) {
                String word = tokenizer.nextToken();
                output.collect(word, value.id);
            }
        }
    }

    public static class Reduce {
        public void reduce(String key, Iterator<Integer> values,
                           OutputCollector<String, List<Integer>> output)
                               {
            // Write your code here
            // Output the results into output buffer.
            // Ps. output.collect(String key, List<Integer> value);
            List<Integer> results = new ArrayList<>();
            int left = -1;
            while (values.hasNext()) {
                int now = values.next();
                if (left != now) {
                    results.add(now);
                }
                left = now;
```

描述 控制台 笔记

Accepted

100%

100% 数据通过测试 总耗时 1904 ms

您的提交打败了 83.60% 的提交!

查看详情

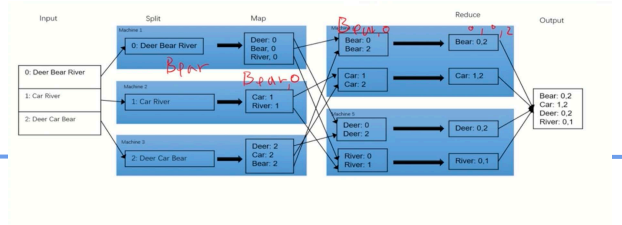
用 Chrome 插件看看更好的参考答案

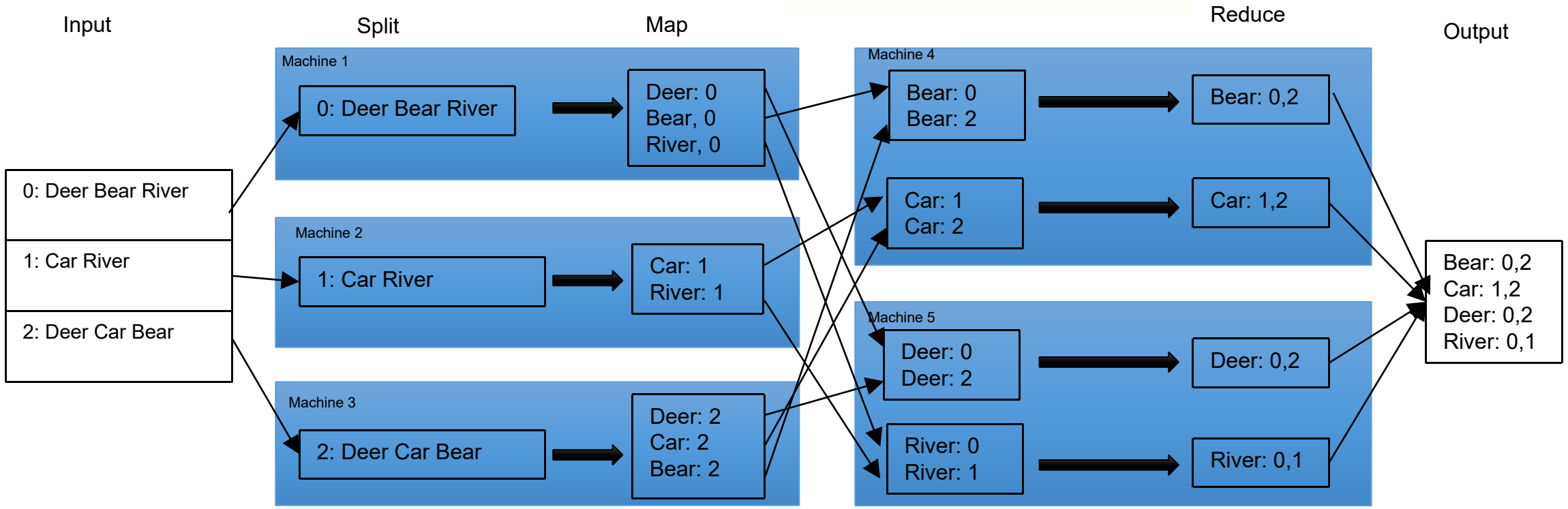
和你的朋友们分享你的AC

输入测试数据 (每行一个参数)

[{"id":1,"content":"This is the content of document1"},
{"id":2,"content":"This is the content of document2"}]

Build inverted index with MapReduce?





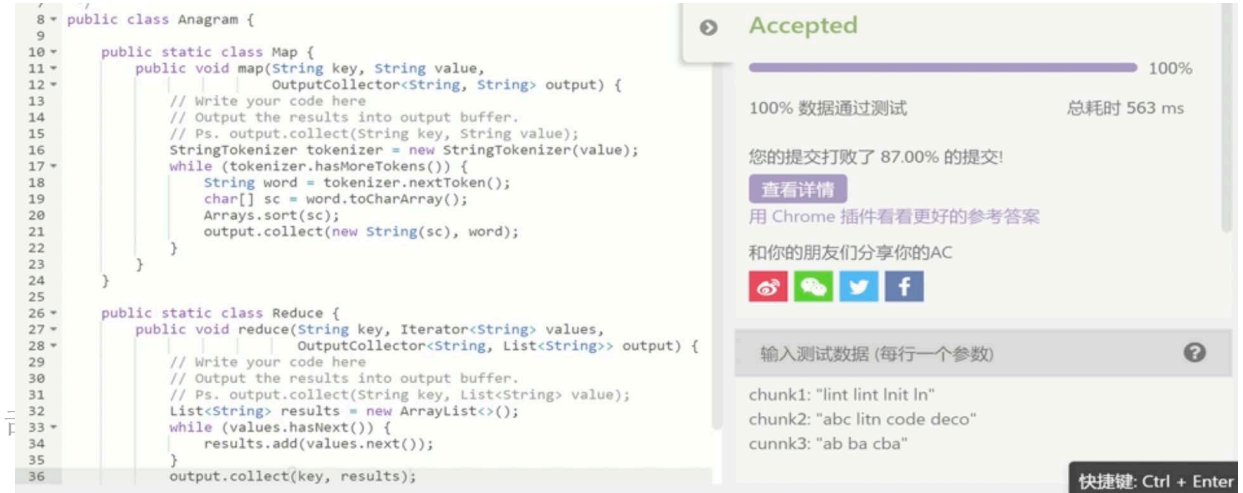

```
//key: the id of a doc
//value: the content of the line
Map( string key, string value)
    for each word in value:
        Output( word, key);
```

```
//key: the name of a word
//valueList: the appearances of this word in documents
Reduce( string key, list valueList )
    List sumList;
    for value in valueList:
        sumList.append(value);
    OutputFinal( key, sumList );
```

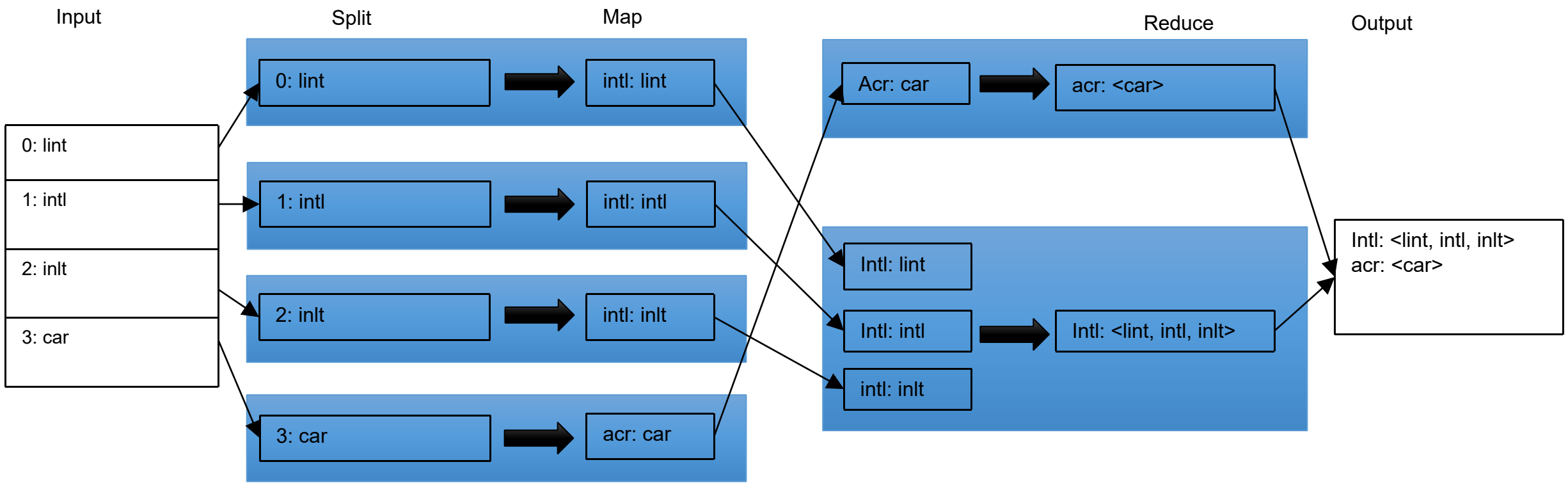
Apple Interviewer: Anagram - Map Reduce

http://www.lintcode.com/en/problem/anagram-map-reduce/

http://www.jiuzhang.com/solutions/anagram-map-reduce/



Anagram - Map Reduce



Map

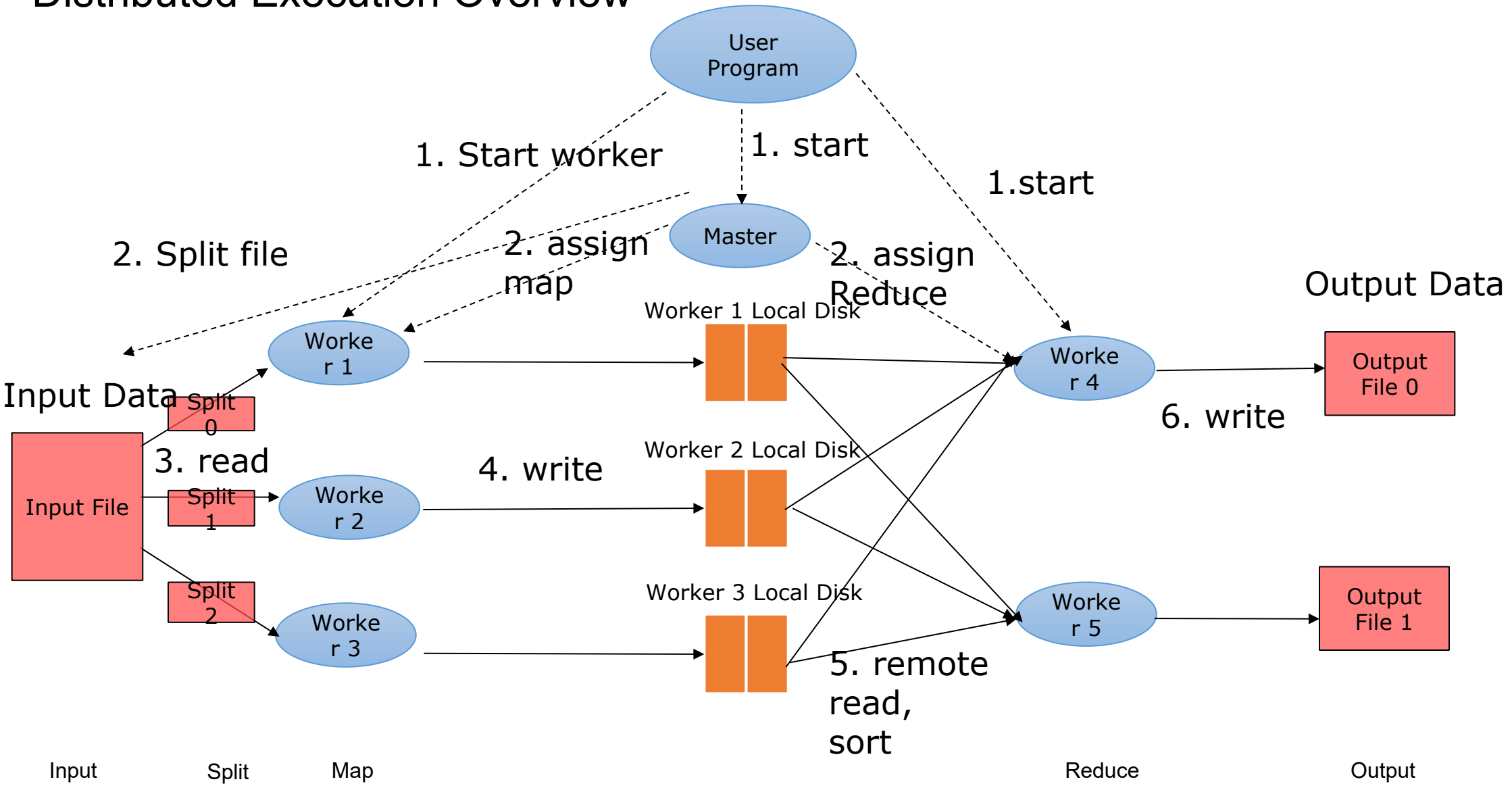
Input	Output
k = lineID	k = sort(word)
v = word	v = word

Reduce

Input	Output
k = sort(word)	k = anagram0
v = {anagram0, anagram1, ...}	v = {anagram1, anagram2,...}

Interviewer:
Design a MapReduce system

Distributed Execution Overview



常见问题答疑

1. Mapper 和 Reducer是同时工作还是先Mapper 工作还是 Reducer工作的么?
   Mapper要结束了后Reducer才能运行

2. 运行过程中一个Mapper或者Reducer挂了怎么办?
   重新分配一台机器做

3. Reducer一个机器Key特别大怎么办?
   加一个random后缀，类似Shard Key

4. Input 和 Output 存放在哪?
   存放在GFS里面

5. Local Disk 上面的data有木有必要保存在GFS上面? 要是丢了怎么办?
   不需要，丢了重做就好

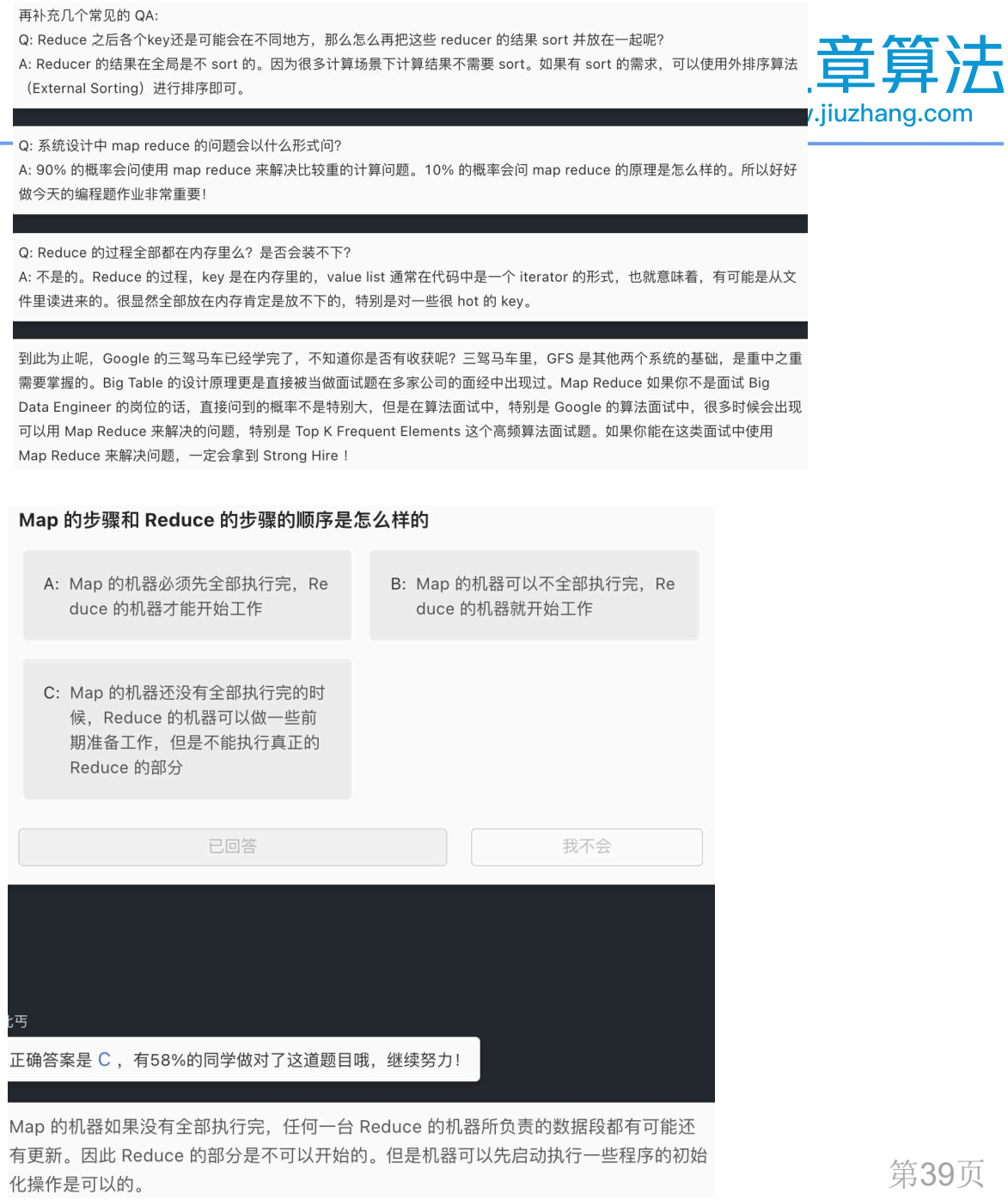
6. Mapper 和 Reducer 可以放在同一台机器么?
   这样设计并不是特别好，Mapper 和Reducer之前都有很多需要预处理的工作。两台机器可以并行的预处理。

MapReduce Whole process

1. (Start)User program start master and worker.
2. (Assign Task)Master assign task to the map worker and reduce worker. (Assign Map and Reduce code)
3. (Split)Master Split the input data.
4. (Map Read)Each map worker read the split input data.
5. (Map)Each map worker do the “Map” job on their machine.
6. (Map output)Each map worker output the file in the local disk of its worker.
7. (Reduce Fetch)Each reduce worker fetch the data from the map worker.
8. (Reduce)Each Reducer worker do the “Reduce” job on their machine.
9. (Reduce output) Reduce worker output the final output data.

MapReduce Framework

- Map Reduce Solve Problem
  - Words Count
  - Inverted index
  - Anagrams
  - Top K Frequency (http://bit.ly/25D8Q7I)
  - PageRank (http://bit.ly/1TOwoyV)
- Map Reduce Step
  - Step1 Input
  - Step2 Split
  - Step3 Map
  - Step4 传输
  - Step5 Reduce
  - Step6 Output
- Map Reduce System
  - Master and Worker
- More
  - 大数据班敬请请期待.....



相关阅读资料

- Novice, http://langyu.iteye.com/blog/992916
- Expert, http://data.qq.com/article?id=543
- Expert, https://www.jianshu.com/p/0ddf3ae19b49
- Expert/Master, http://novoland.github.io/%E5%B7%A5%E4%BD%9C/2014/09/04/MapReduce%20Algorithms.html
- Expert/Master, https://www.slideshare.net/romain_jacotin/the-google-mapreduce
- Master, http://www.cnblogs.com/yepei/p/6292440.html