北京交通大学
BEIJING JIAOTONG UNIVERSITY

分布式计算架构——MapReduce

目录

- 概述
- MapReduce体系结构
- MapReduce工作流程
- 实例分析：WordCount

概述

分布式并行编程

MapReduce模型简介

Map和Reduce函数

分布式并行编程

谷歌公司最先提出了分布式并行编程模型MapReduce，Hadoop MapReduce是它的开源实现。

	传统并行计算框架	MapReduce
集群架构/容错性	共享式(共享内存/共享存储)，容错性差	非共享式，容错性好
硬件/价格/扩展性	刀片服务器、高速网、SAN，价格贵，扩展性差	普通PC机，便宜，扩展性好
编程/学习难度	what-how，难	what，简单
适用场景	实时、细粒度计算、计算密集型	批处理、非实时、数据密集型

MapReduce模型简介

- MapReduce将复杂的、运行于大规模集群上的并行计算过程高度地抽象到了两个函数：Map和Reduce
- 编程容易，不需要掌握分布式并行编程细节，也可以很容易把自己的程序运行在分布式系统上，完成海量数据的计算
- MapReduce采用“分而治之”策略，一个存储在分布式文件系统中的大规模数据集，会被切分成许多独立的分片（split），这些分片可以被多个Map任务并行处理

MapReduce模型简介

- MapReduce设计的一个理念就是“计算向数据靠拢”，而不是“数据向计算靠拢”，因为，移动数据需要大量的网络传输开销
- MapReduce框架采用了Master/Slave架构，包括一个Master和若干个Slave。Master上运行JobTracker，Slave上运行TaskTracker
- Hadoop框架是用Java实现的，但是，MapReduce应用程序则不一定要用Java来写

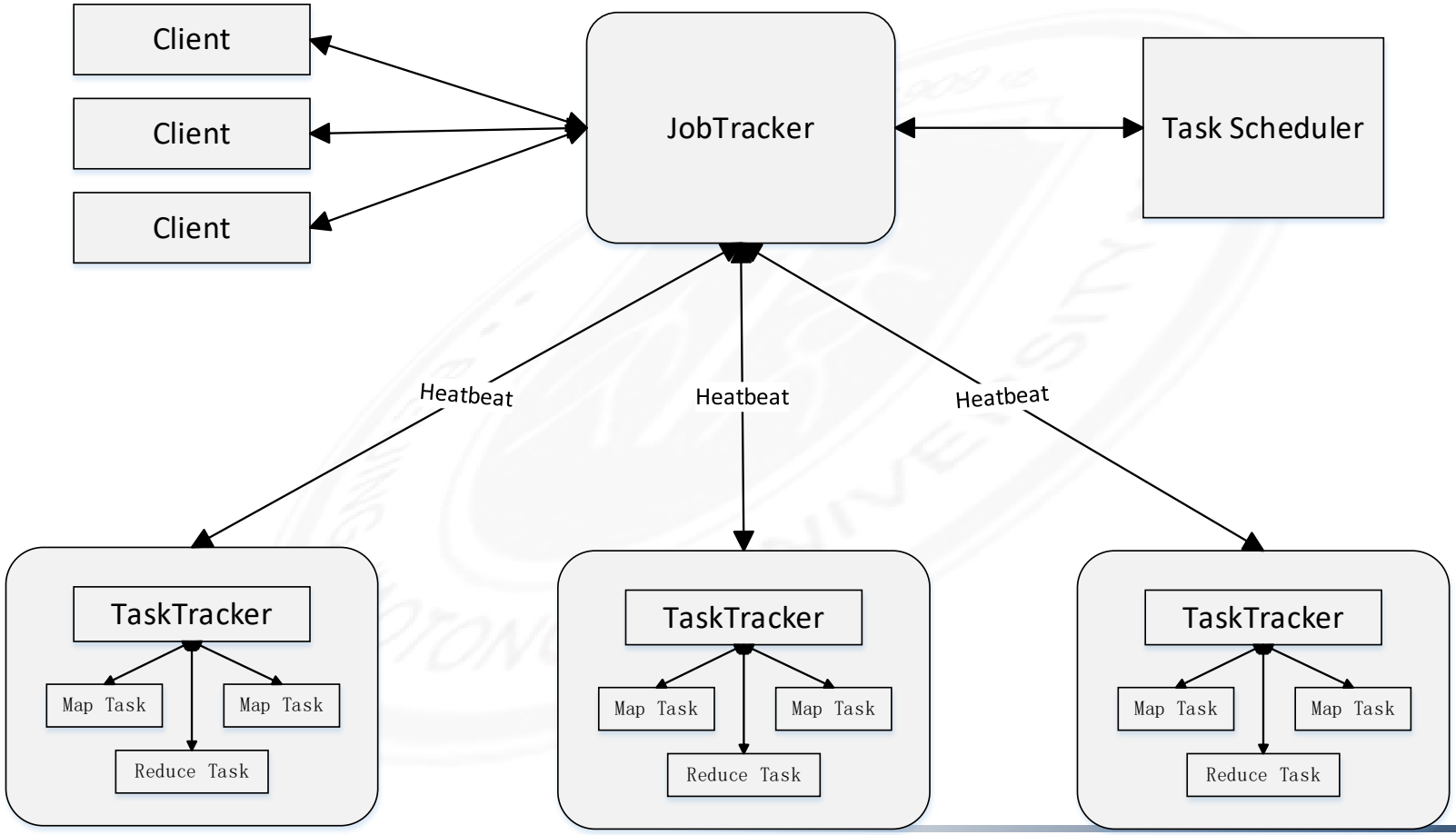
Map和Reduce函数

Map和Reduce

函数	输入	输出	说明
Map	<k_1,v_1> 如： <行号,"a b c">	List(<k_2,v_2>) 如： <"a",1> <"b",1> <"c",1>	1.将小数据集进一步解析成一批<key,value>对，输入Map函数中进行处理 2.每一个输入的<k_1,v_1>会输出一批<k_2,v_2>。<k_2,v_2>是计算的中间结果
Reduce	<k_2,List(v_2)> 如： <"a",<1,1,1>>	<k_3,v_3> <"a",3>	输入的中间结果<k_2,List(v_2)>中的List(v_2)表示是一批属于同一个k_2的value

MapReduce的体系结构

MapReduce体系结构主要由四个部分组成，分别是：Client、JobTracker、TaskTracker以及Task



MapReduce的体系结构

MapReduce主要有以下4个部分组成：

1）Client

•用户编写的MapReduce程序通过Client提交到JobTracker端

•用户可通过Client提供的一些接口查看作业运行状态

2）JobTracker

•JobTracker负责资源监控和作业调度

•JobTracker 监控所有TaskTracker与Job的健康状况，一旦发现失败，就将相应的任务转移到其他节点

•JobTracker 会跟踪任务的执行进度、资源使用量等信息，并将这些信息告诉任务调度器（TaskScheduler），而调度器会在资源出现空闲时，选择合适的任务去使用这些资源

MapReduce的体系结构

3）TaskTracker

•TaskTracker 会周期性地通过“心跳”将本节点上资源的使用情况和任务的运行进度汇报给JobTracker，同时接收JobTracker 发送过来的命令并执行相应的操作（如启动新任务、杀死任务等）

•TaskTracker 使用“slot”等量划分本节点上的资源量（CPU、内存等）。一个Task 获取到一个slot 后才有机会运行，而Hadoop调度器的作用就是将各个TaskTracker上的空闲slot分配给Task使用。slot 分为Map slot 和Reduce slot 两种，分别供MapTask 和Reduce Task 使用

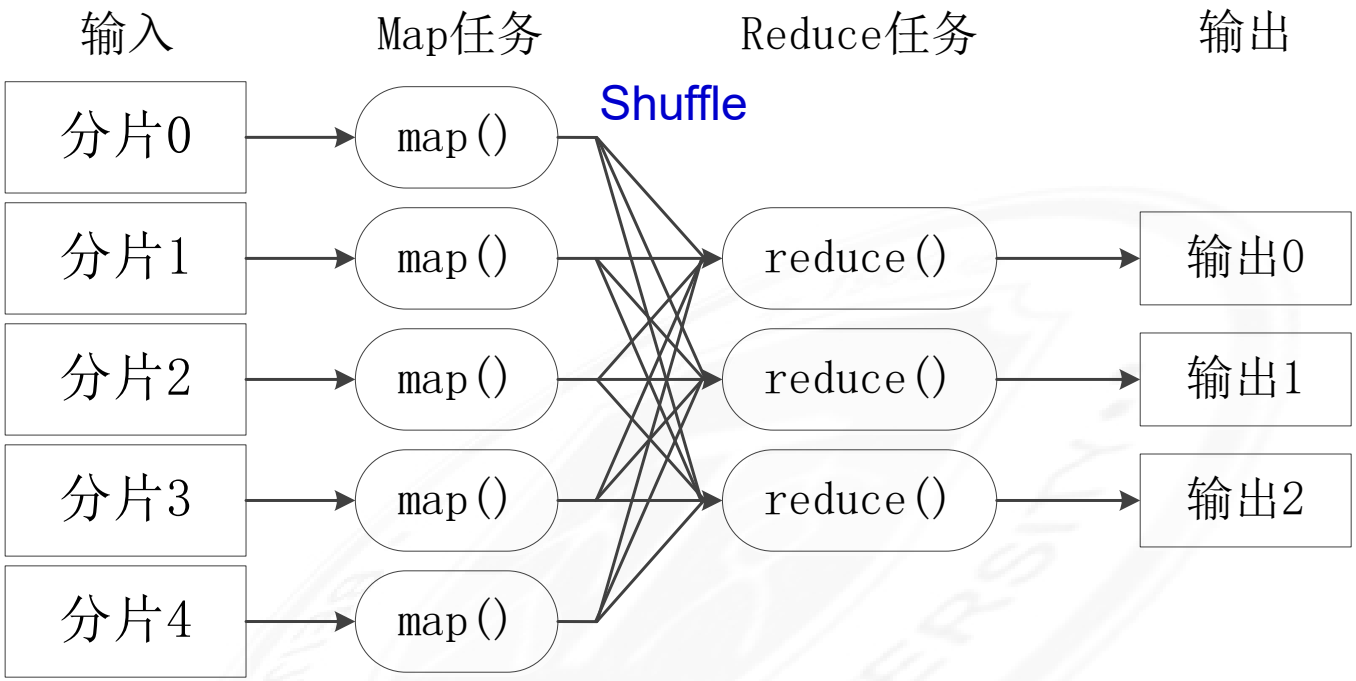
4）Task

Task 分为Map Task 和Reduce Task 两种，均由TaskTracker 启动

MapReduce工作流程

- 工作流程概述
- MapReduce各个执行阶段
- Shuffle过程详解

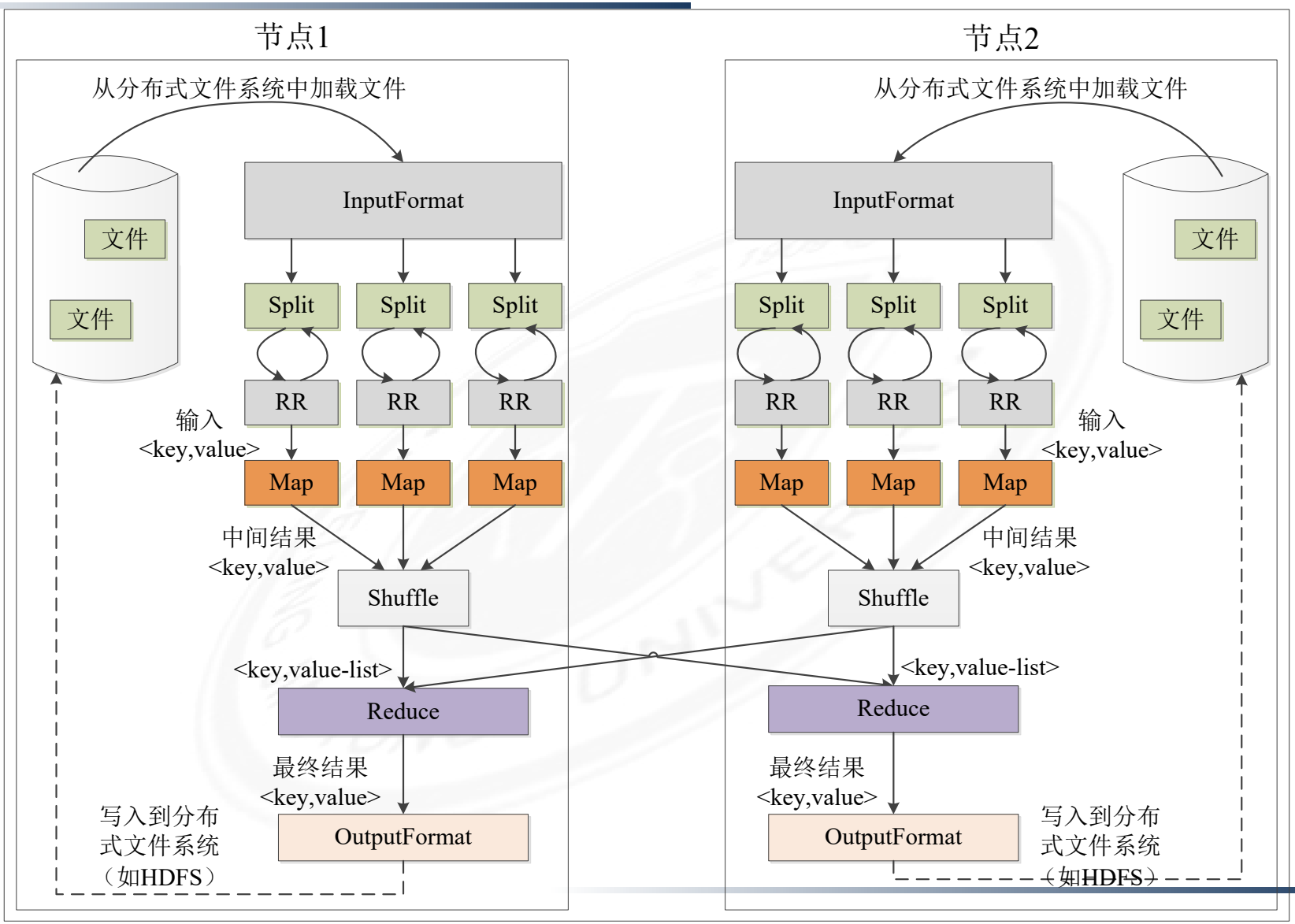
工作流程概述



MapReduce工作流程

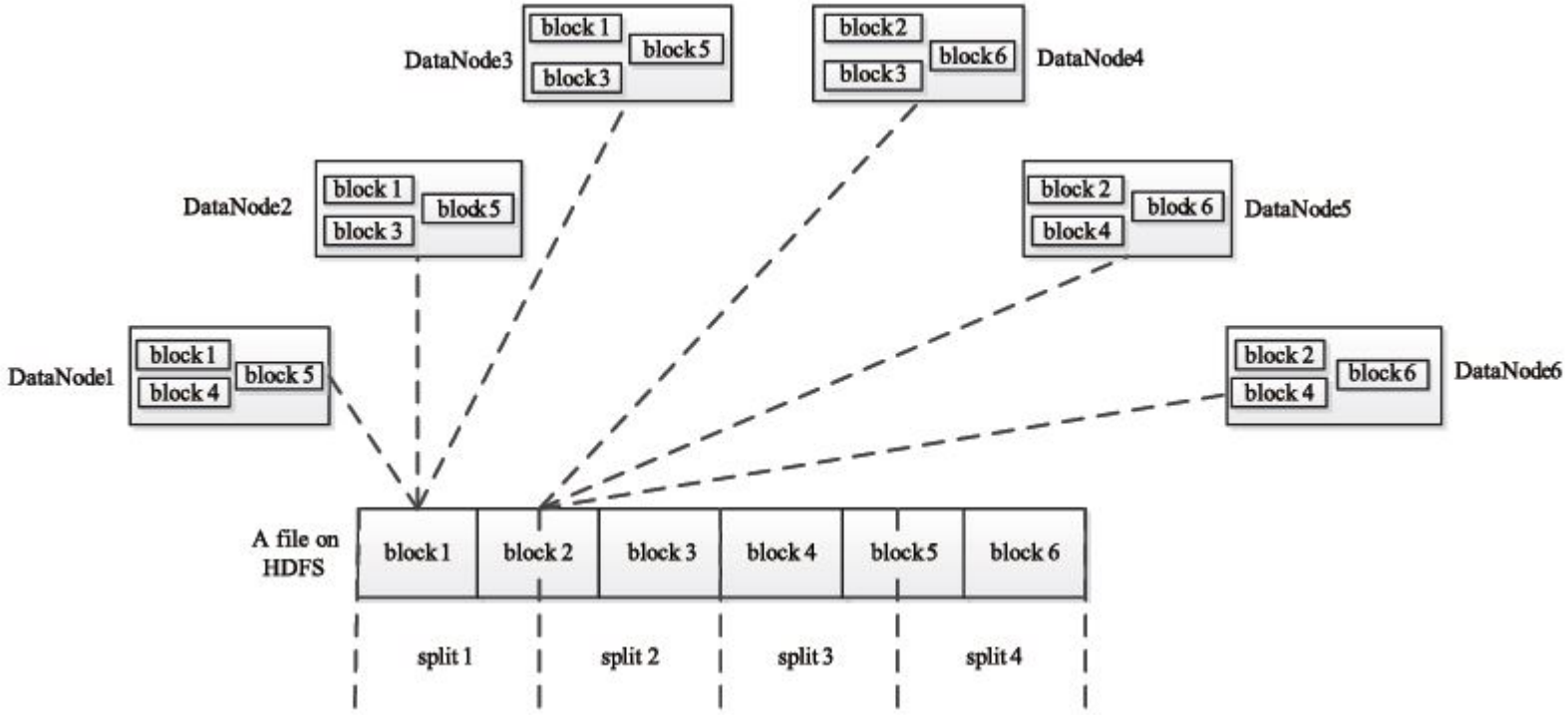
- 不同的**Map**任务之间不会进行通信
- 不同的**Reduce**任务之间也不会发生任何信息交换
- 用户不能显式地从一台机器向另一台机器发送消息
- 所有的数据交换都是通过**MapReduce**框架自身去实现的

MapReduce各个执行阶段



MapReduce各个执行阶段

关于**Split**（分片）

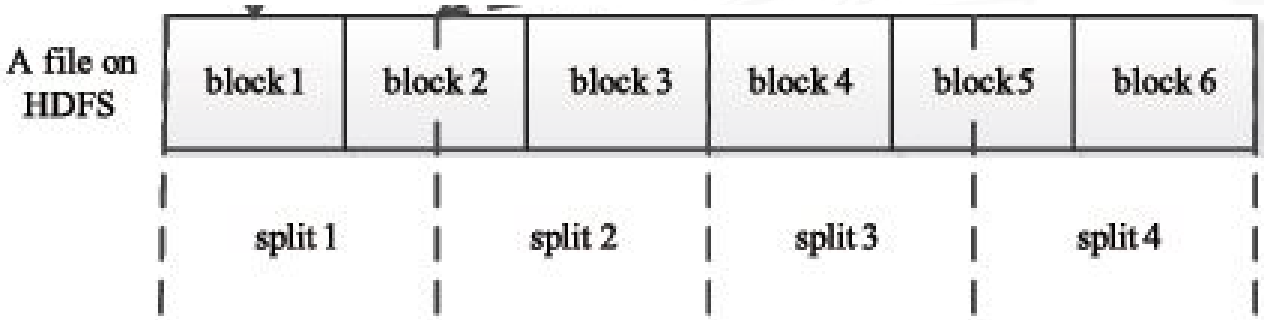


HDFS 以固定大小的block 为基本单位存储数据，而对于MapReduce 而言，其处理单位是split。split 是一个逻辑概念，它只包含一些元数据信息，比如数据起始位置、数据长度、数据所在节点等。它的划分方法完全由用户自己决定。

MapReduce各个执行阶段

Map任务的数量

•Hadoop为每个split创建一个Map任务，split 的多少决定了Map任务的数目。大多数情况下，理想的分片大小是一个HDFS块



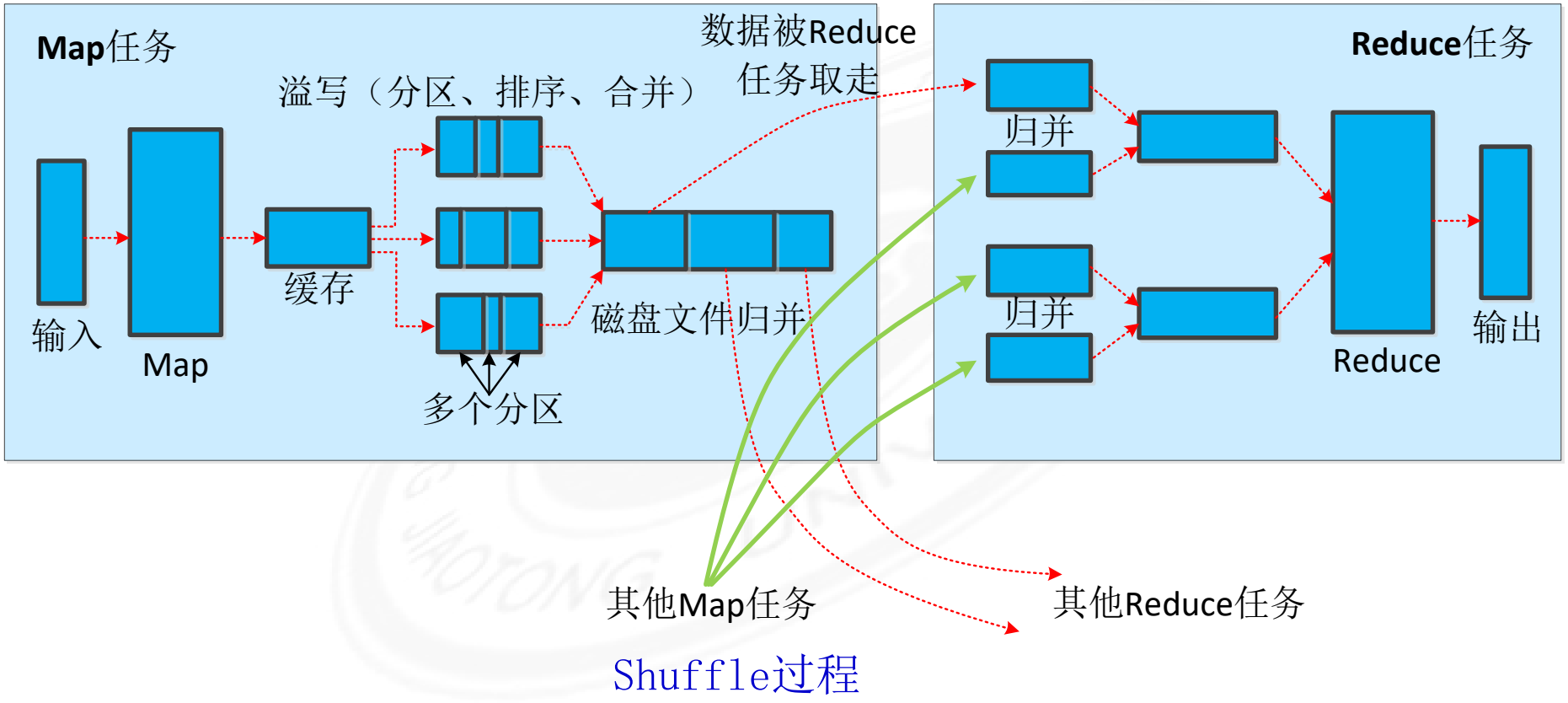
Reduce任务的数量

•最优的Reduce任务个数取决于集群中可用的reduce任务槽(slot)的数目

•通常设置比reduce任务槽数目稍微小一些的Reduce任务个数（这样可以预留一些系统资源处理可能发生的错误）

Shuffle过程详解

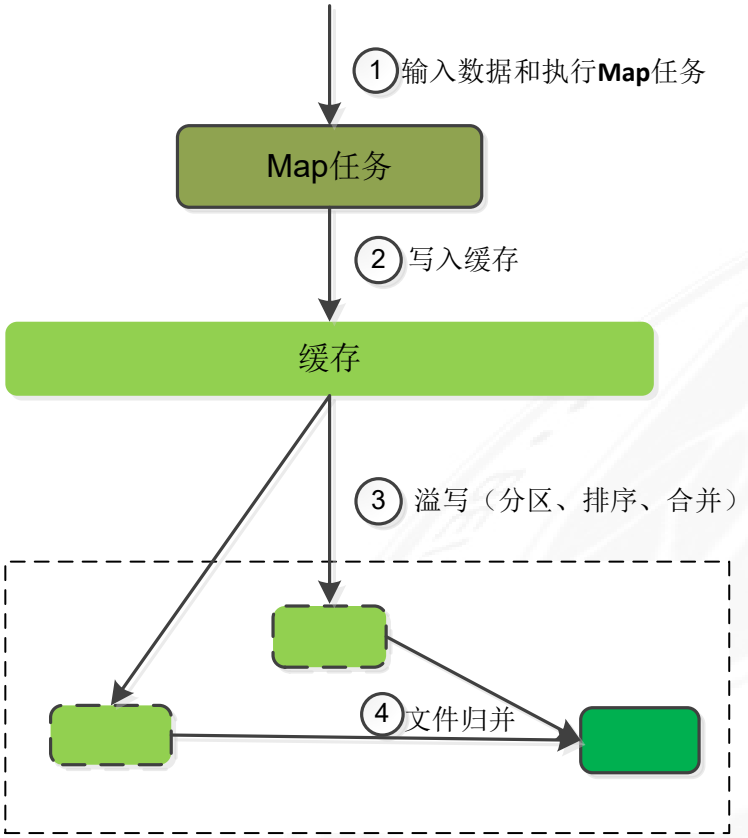
1. Shuffle过程简介



Shuffle过程

Shuffle过程详解

2. Map端的Shuffle过程



- 每个Map任务分配一个缓存
- MapReduce默认100MB缓存

- 设置溢写比例0.8
- 分区默认采用哈希函数
- 排序是默认的操作
- 排序后可以合并（Combine）
- 合并不能改变最终结果

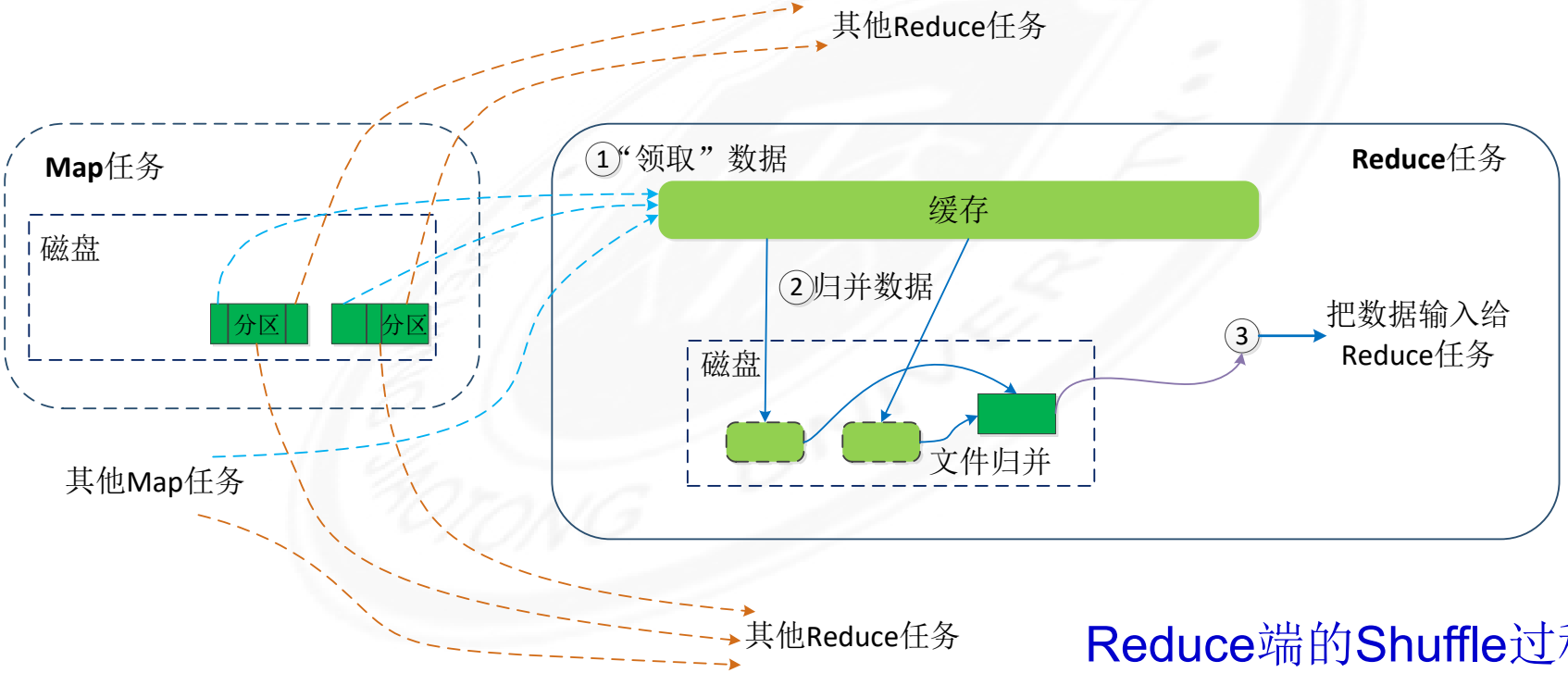
- 在Map任务全部结束之前进行归并
- 归并得到一个大的文件，放在本地磁盘
- 文件归并时，如果溢写文件数量大于预定值（默认是3）则可以再次启动Combiner，少于3不需要
- JobTracker会一直监测Map任务的执行，并通知Reduce任务来领取数据

合并（Combine）和归并（Merge）的区别：
两个键值对<“a”,1>和<“a”,1>，如果合并，会得到<“a”,2>，如果归并，会得到<“a”,<1,1>>

Shuffle过程详解

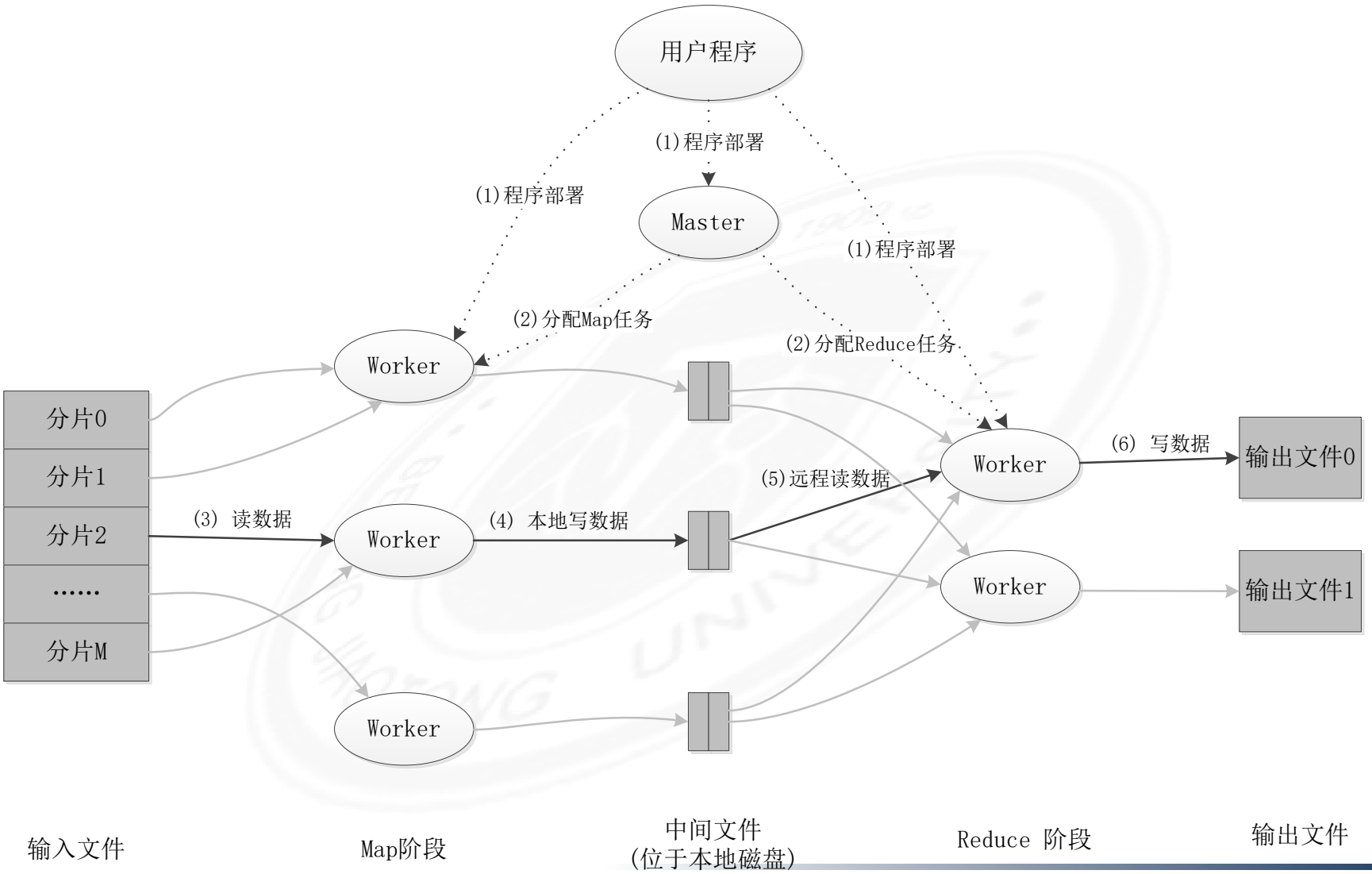
3. Reduce端的Shuffle过程

- Reduce任务通过RPC向JobTracker询问Map任务是否已经完成，若完成，则领取数据
- Reduce领取数据先放入缓存，来自不同Map机器，先归并，再合并，写入磁盘
- 多个溢写文件归并成一个或多个大文件，文件中的键值对是排序的
- 当数据很少时，不需要溢写到磁盘，直接在缓存中归并，然后输出给Reduce



Reduce端的Shuffle过程

MapReduce应用程序执行过程



实例分析：WordCount

- WordCount程序任务
- WordCount设计思路
- 一个WordCount执行过程的实例

WordCount程序任务

WordCount程序任务

程序	WordCount
输入	一个包含大量单词的文本文件
输出	文件中每个单词及其出现次数（频数），并按照单词字母顺序排序，每个单词和其频数占一行，单词和频数之间有间隔

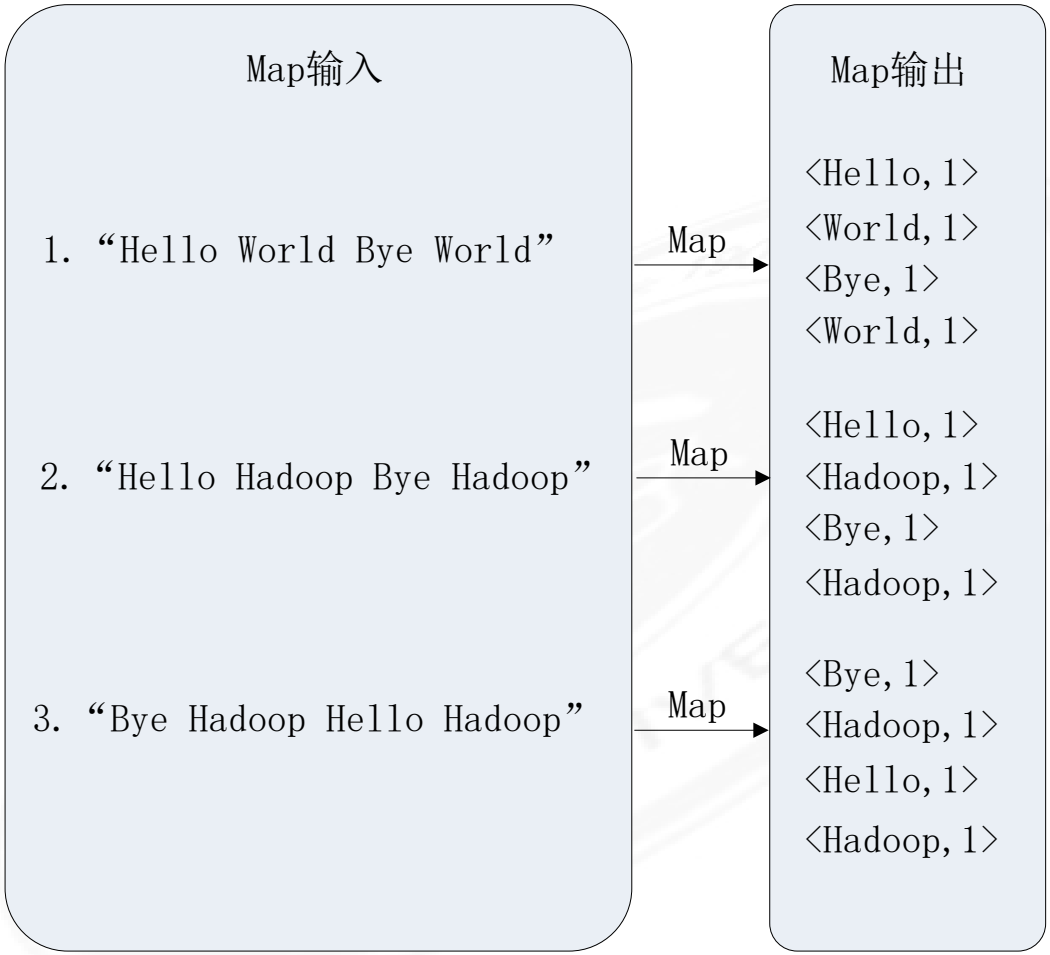
一个WordCount的输入和输出实例

输入	输出
Hello World Hello Hadoop Hello MapReduce	Hadoop 1 Hello 3 MapReduce 1 World 1

WordCount设计思路

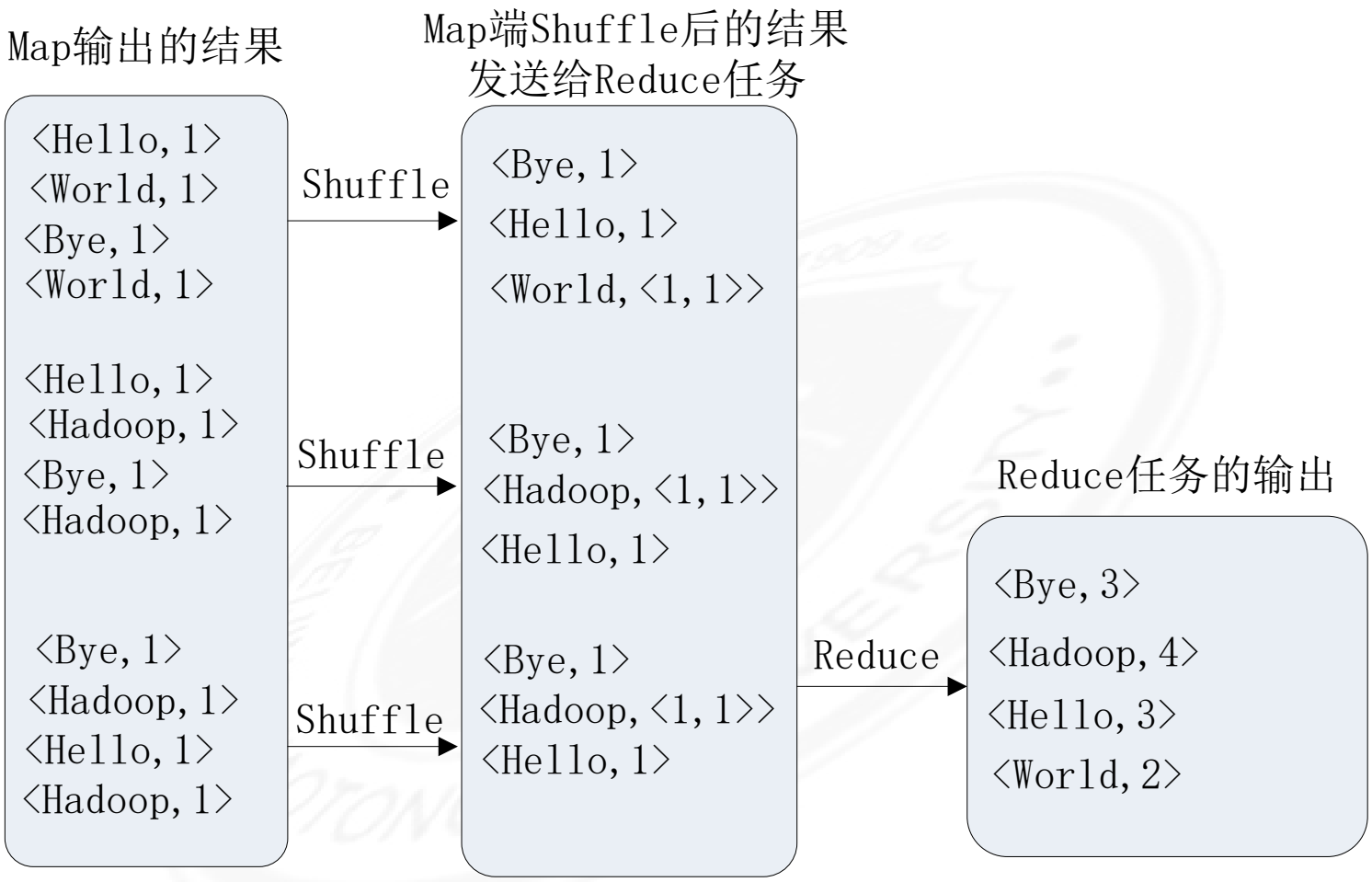
- 首先，需要检查WordCount程序任务是否可以采用 MapReduce来实现
- 其次，确定MapReduce程序的设计思路
- 最后，确定MapReduce程序的执行过程

一个WordCount执行过程的实例



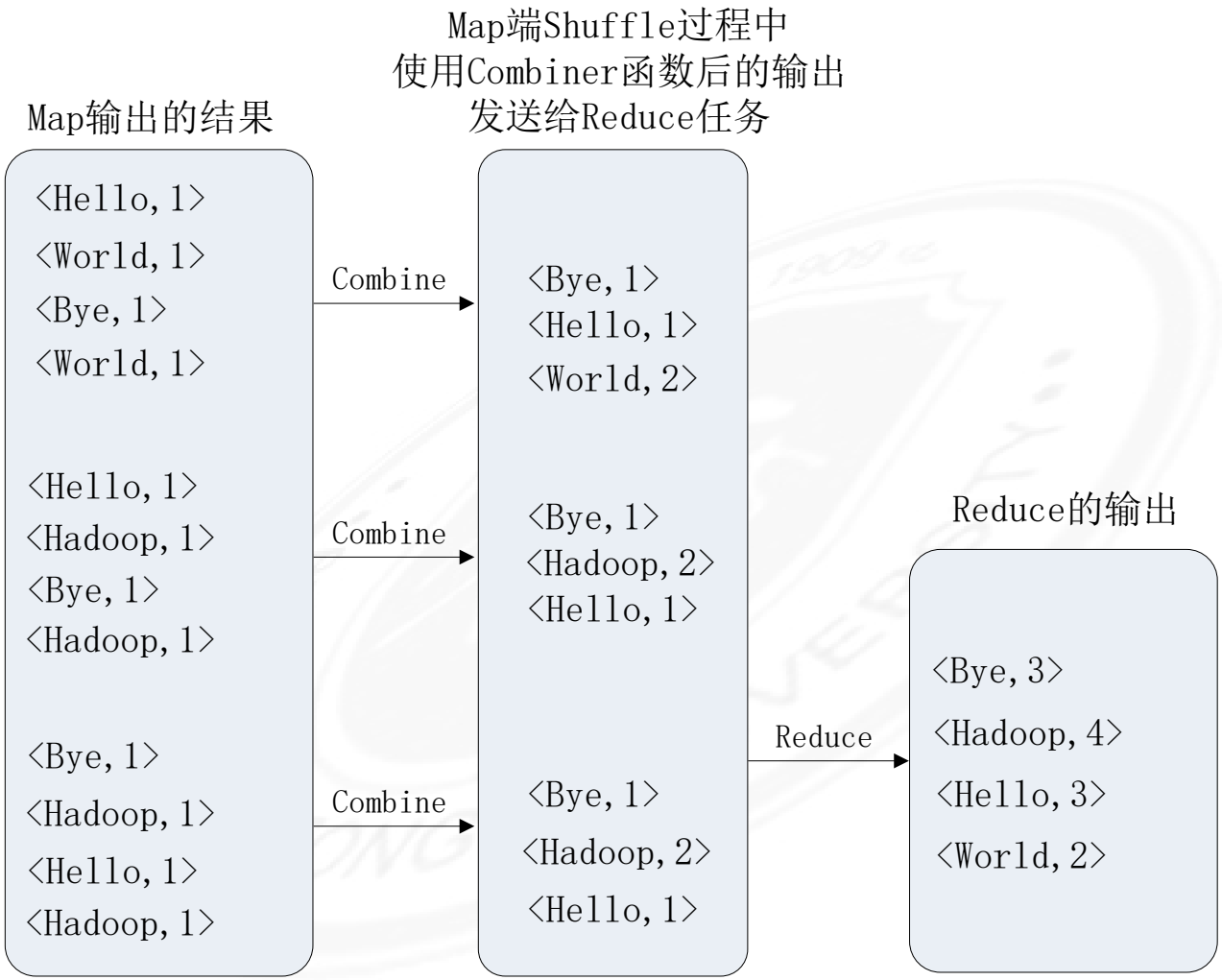
Map过程示意图

一个WordCount执行过程的实例



用户没有定义Combiner时的Reduce过程示意图

一个WordCount执行过程的实例



用户有定义Combiner时的Reduce过程示意图