大数据专题技术型公开课

第1讲 分布式并行编程框架MapReduce

林子雨 博士/助理教授

厦门大学计算机科学系

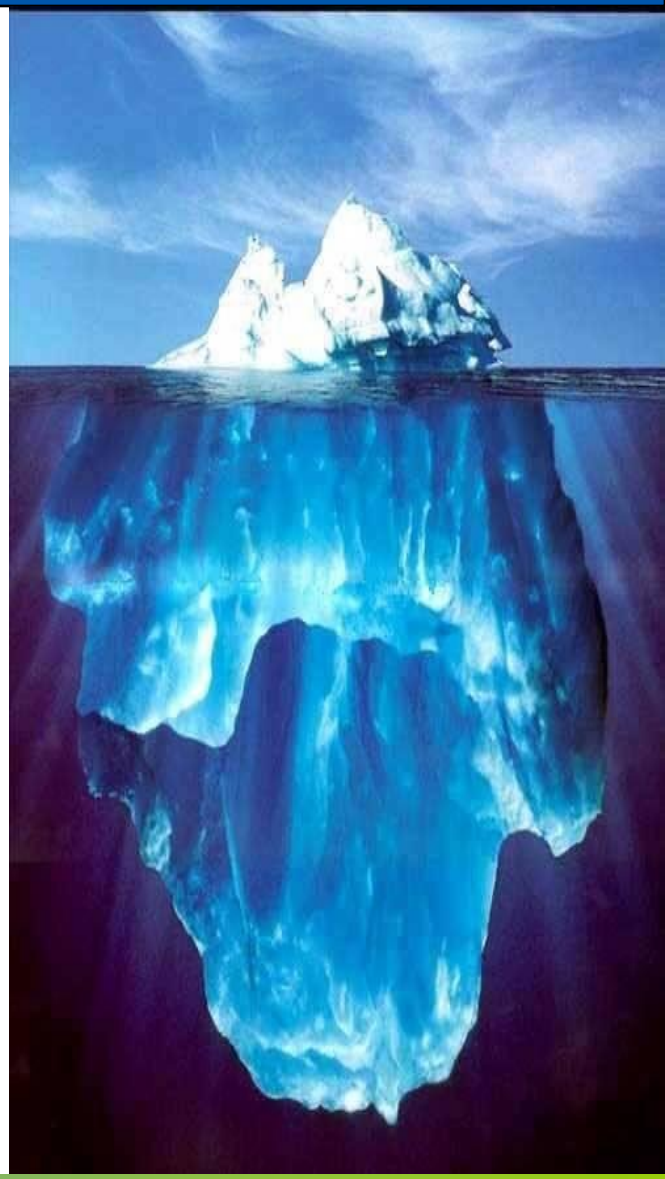
厦门大学云计算与大数据研究中心

E-mail: ziyulin@xmu.edu.cn

主页：http://www.cs.xmu.edu.cn/linziyu

提纲

- 7.1 概述
- 7.2 MapReduce工作流程
- 7.3 实例分析：WordCount
- 7.4 MapReduce编程实践



本PPT是如下教材的配套讲义：
21世纪高等教育计算机规划教材
《大数据技术原理与应用
——概念、存储、处理、分析与应用》
（2015年6月第1版）
厦门大学 林子雨 编著，人民邮电出版社
ISBN:978-7-115-39287-9

欢迎访问《大数据技术原理与应用》教材官方网站：
http://dblab.xmu.edu.cn/post/bigdata

7.1 概述

- 7.1.1 分布式并行编程
- 7.1.2 MapReduce模型简介
- 7.1.3 Map和Reduce函数

7.1.1分布式并行编程

- “摩尔定律”，大约每隔18个月性能翻一番
- 从2005年开始摩尔定律逐渐失效，人们开始借助于分布式并行编程来提高程序性能
- 分布式程序运行在大规模计算机集群上，集群中包括大量廉价服务器，可以并行执行大规模数据处理任务，从而获得海量的计算能力
- 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

7.1.2MapReduce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
- 在MapReduce中，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小数据块，这些小数据块可以被多个Map任务并行处理
- MapReduce框架会为每个Map任务输入一个数据子集，Map任务生成的结果会继续作为Reduce任务的输入，最终由Reduce任务输出最后结果，并写入到分布式文件系统中
- 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

7.1.3Map和Reduce函数

表7-1 Map和Reduce

函数	输入	输出	说明
Map	<k_1,v_1>	List(<k_2,v_2>)	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)>	<k_3,v_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

7.2 MapReduce工作流程

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

7.2.1工作流程概述

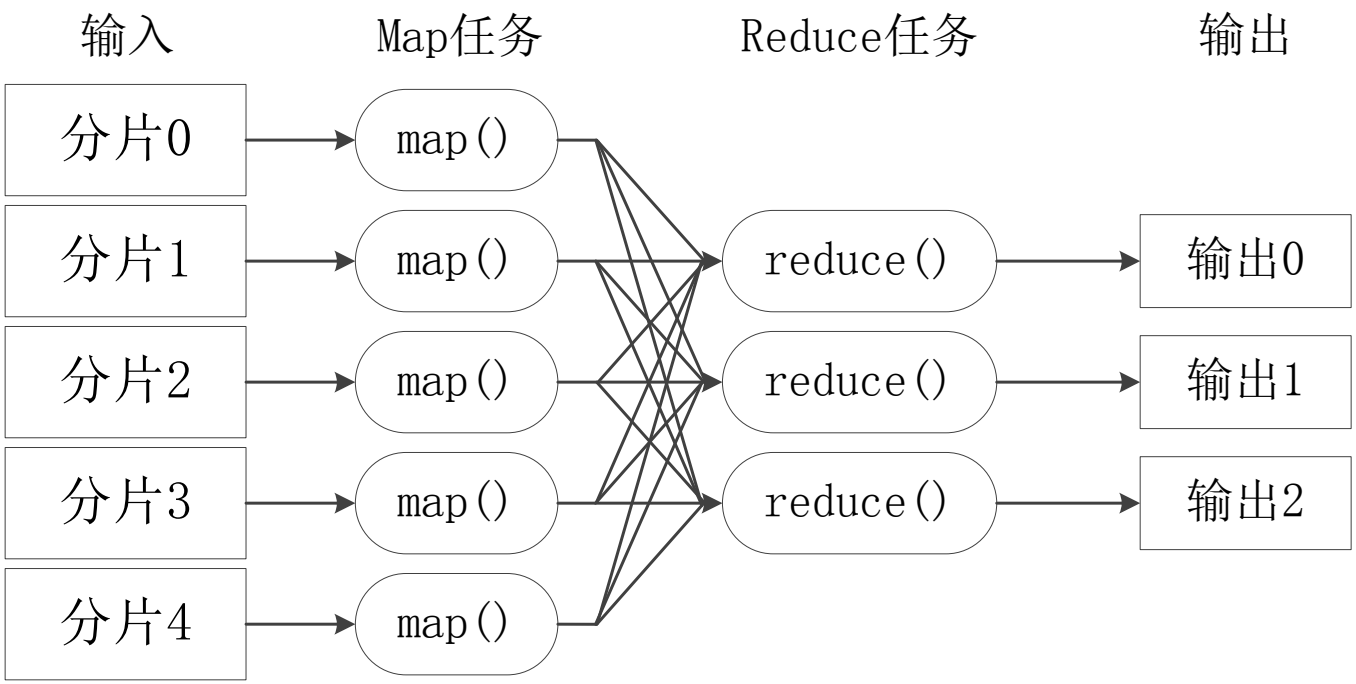


图7-1 MapReduce工作流程

7.2.2MapReduce各个执行阶段

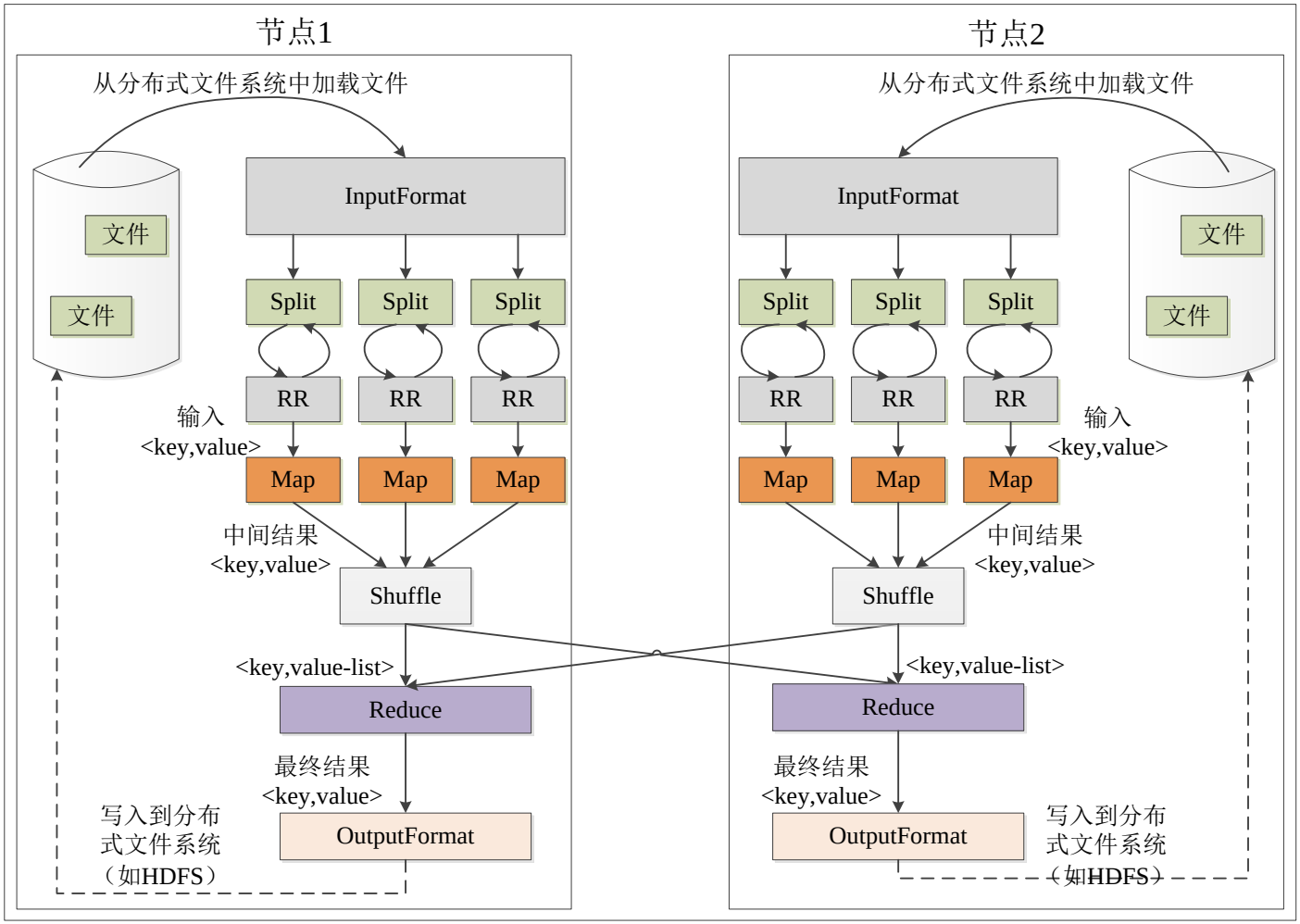


图7-2 MapReduce工作流程中的各个执行阶段

7.2.3Shuffle过程详解

1. Shuffle过程简介

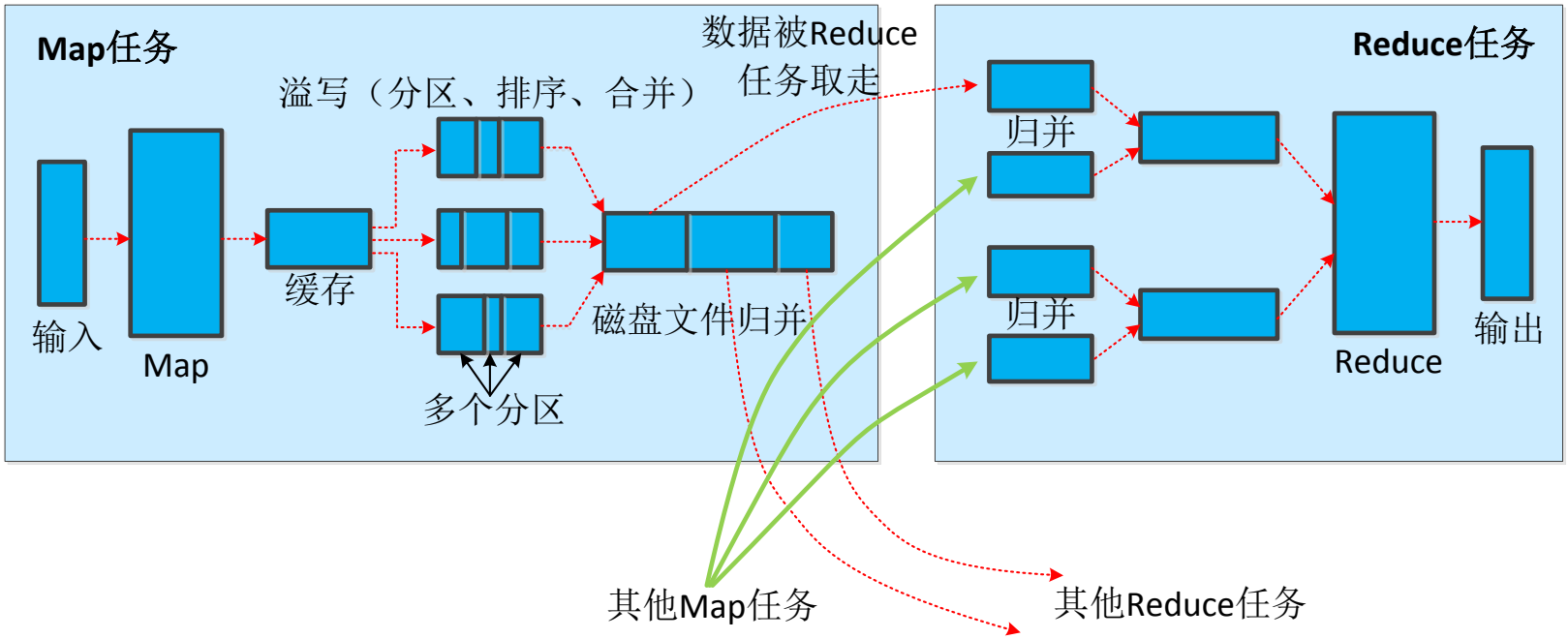


图7-3 Shuffle过程

7.2.3Shuffle过程详解

2. Map端的Shuffle过程

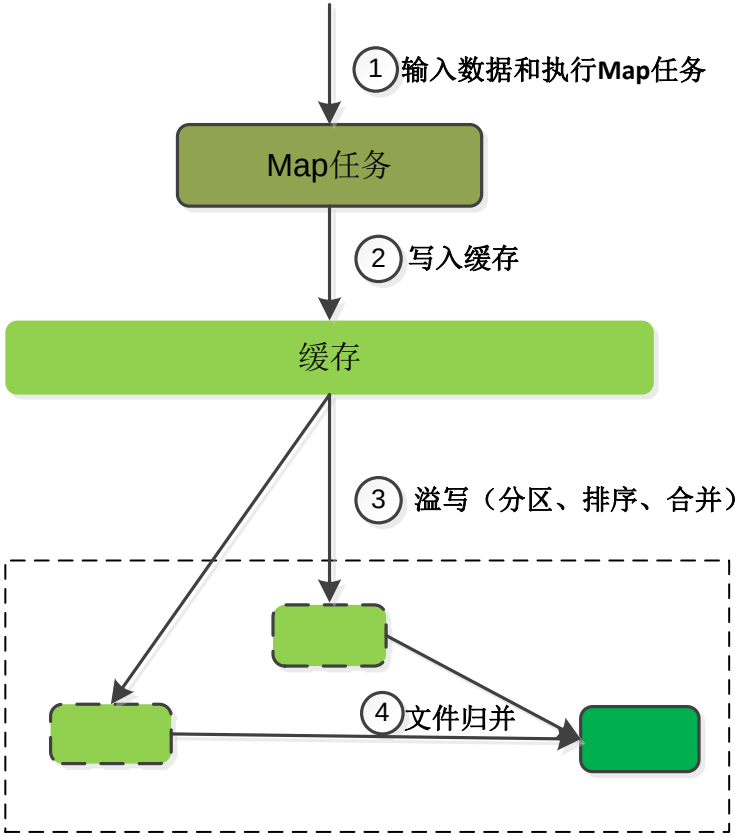


图7-4 Map端的Shuffle过程

7.2.3Shuffle过程详解

3. Reduce端的Shuffle过程

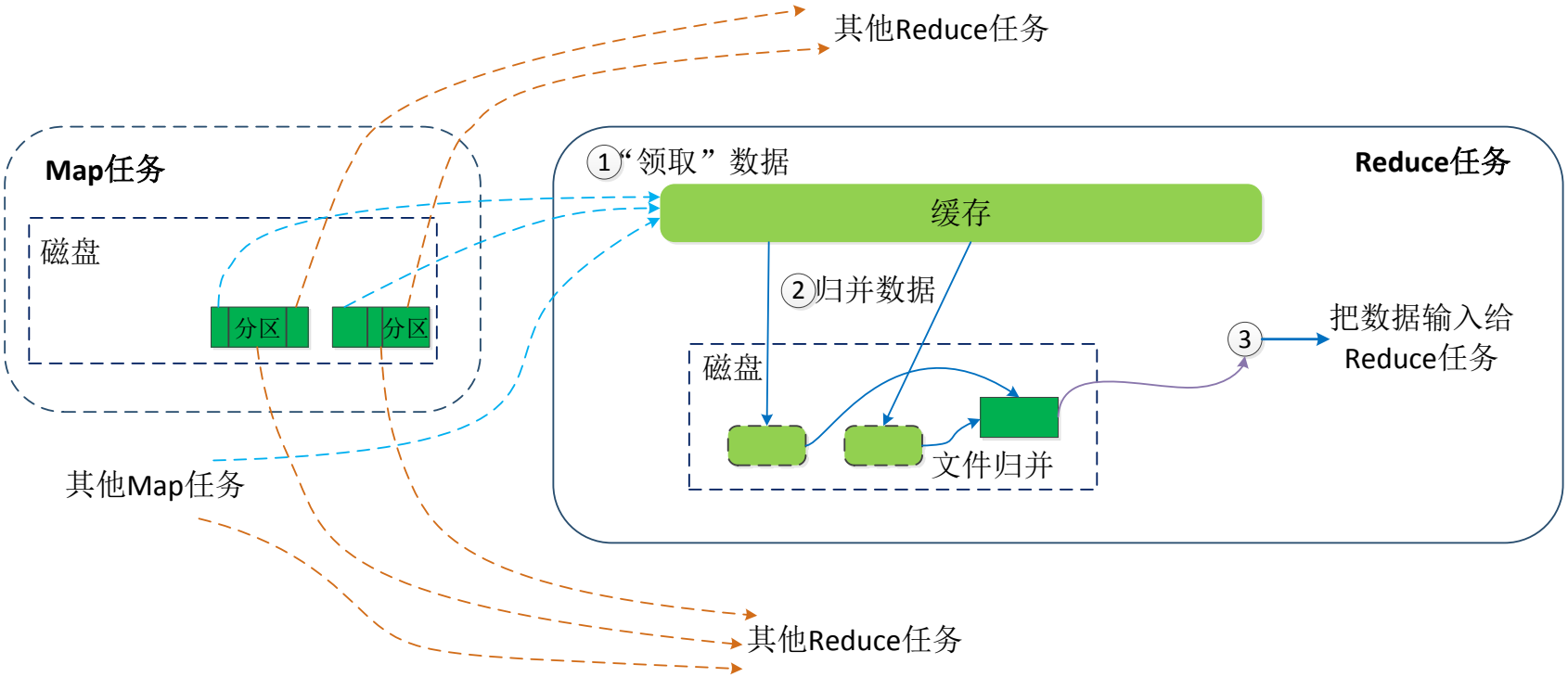


图7-5 Reduce端的Shuffle过程

7.3 实例分析：WordCount

- 7.3.1 WordCount程序任务
- 7.3.2 WordCount设计思路
- 7.3.3 MapReduce具体执行过程
- 7.3.4 一个WordCount执行过程的实例

7.3.1WordCount程序任务

表7-2 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

表7-3 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

7.3.2WordCount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

7.3.3MapReduce具体执行过程

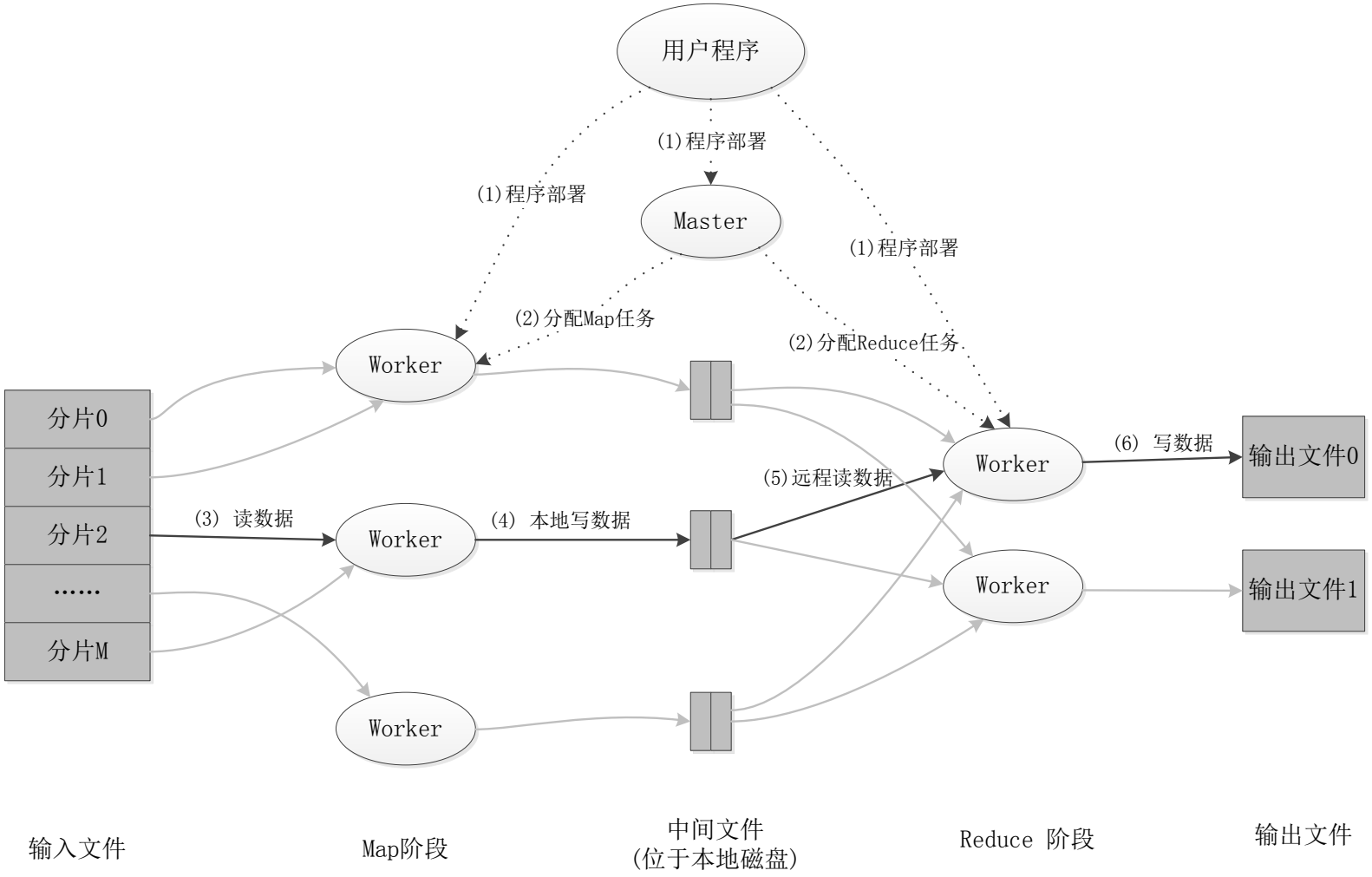


图7-6 WordCount执行过程

7.3.4一个WordCount执行过程的实例

图7-7 Map过程示意图

7.3.4一个WordCount执行过程的实例

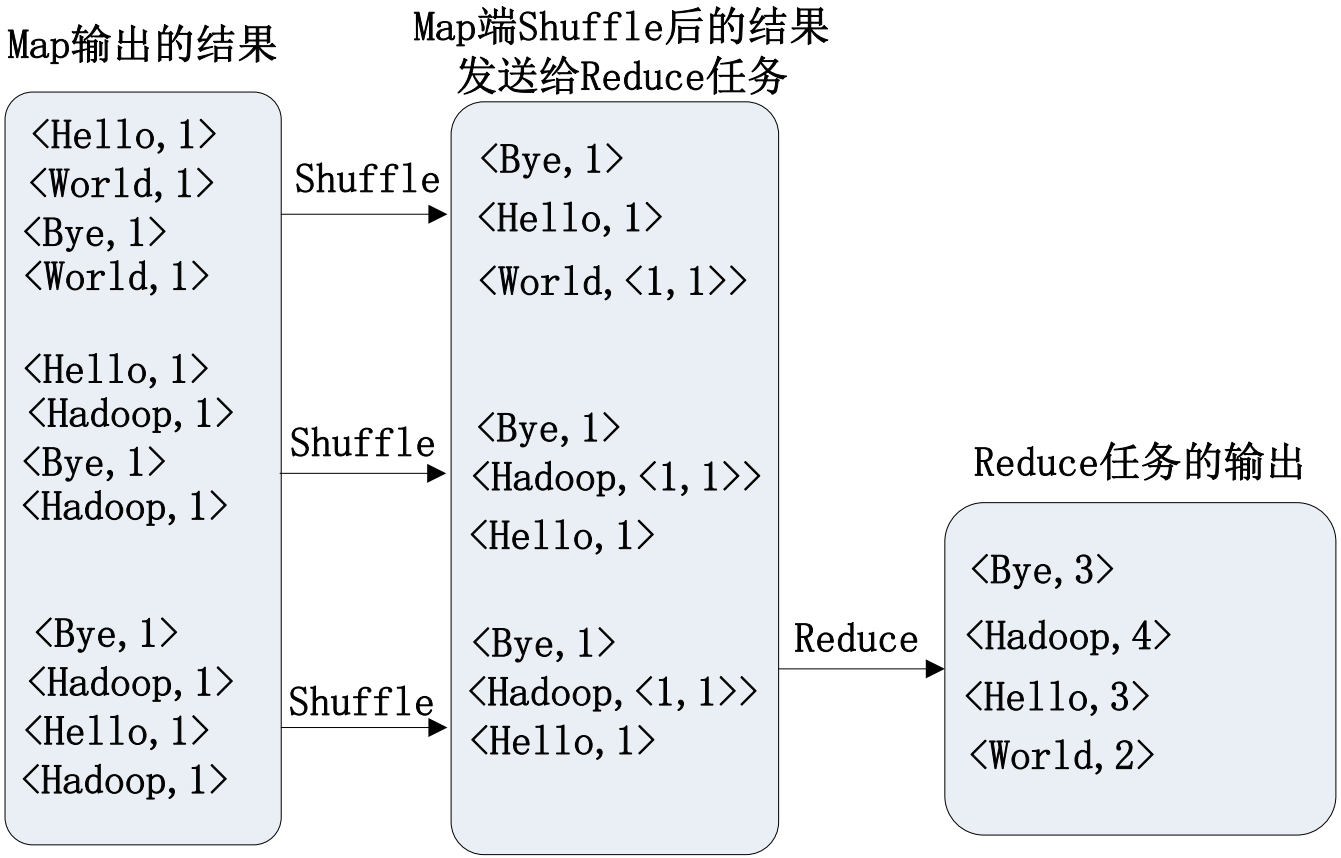


图7-8 用户没有定义Combiner时的Reduce过程示意图

7.3.4一个WordCount执行过程的实例

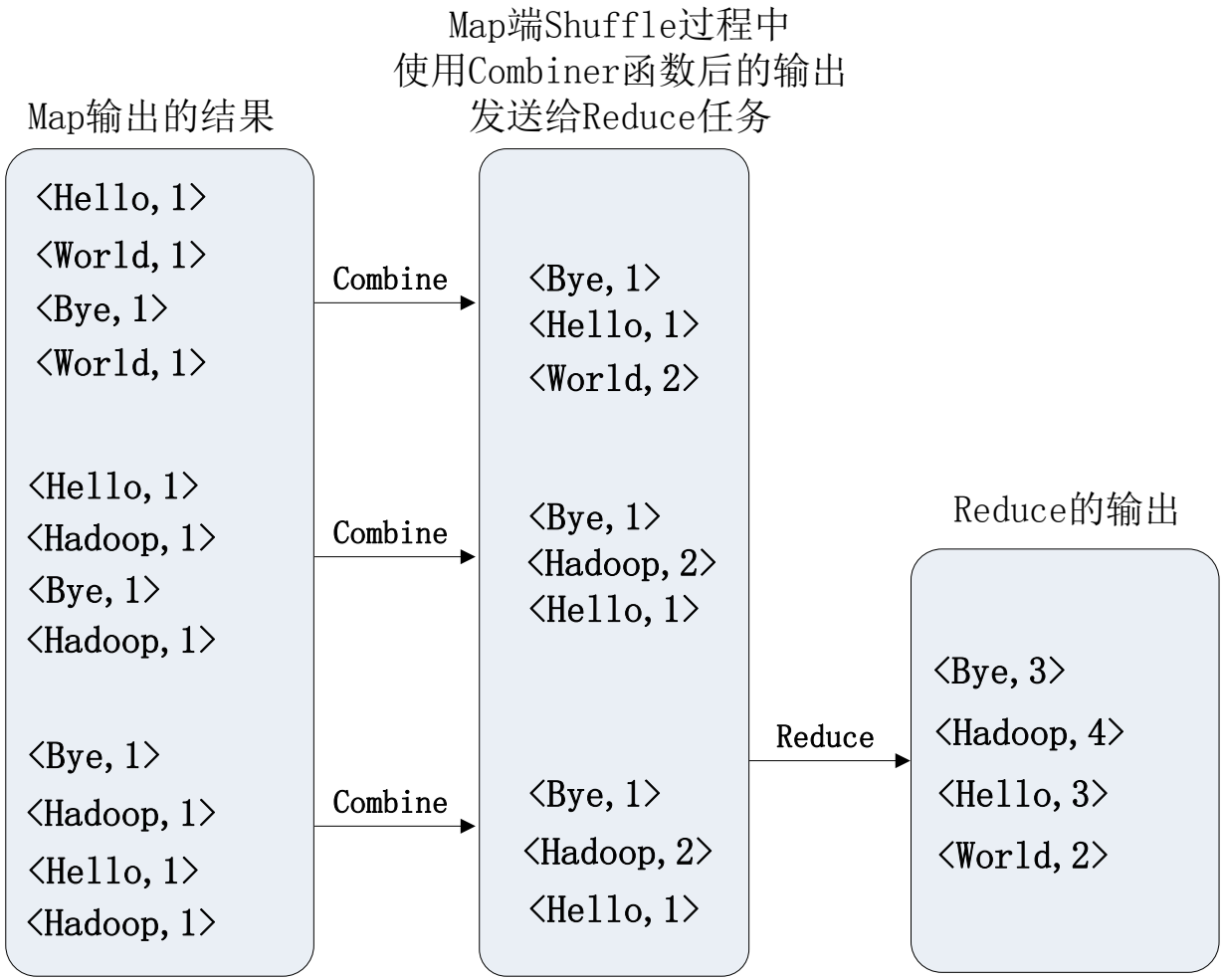


图7-9 用户有定义Combiner时的Reduce过程示意图

7.4 MapReduce编程实践

- 7.4.1任务要求
- 7.4.2编写Map处理逻辑
- 7.4.3编写Reduce处理逻辑
- 7.4.4 编写main方法
- 7.4.5 编译打包代码以及运行程序

7.4.1 任务要求

文件A的内容如下：

```
China is my motherland
I love China
```

文件B的内容如下：

```
I am from  China
```

期望结果如右侧所示：

I	2
is	1
China	3
my	1
love	1
am	1
from	1
motherland	1

7.4.2 编写Map处理逻辑

- Map输入类型为<key,value>
- 期望的Map输出类型为<单词，出现次数>
- Map输入类型最终确定为<Object,Text>
- Map输出类型最终确定为<Text,IntWritable>

```
public static class MyMapper extends
Mapper<Object,Text,Text,IntWritable>{
                private final static IntWritable one = new
IntWritable(1);
                private Text word = new Text();
                public void map(Object key, Text value, Context
context) throws IOException,InterruptedException{
                        StringTokenizer itr = new
StringTokenizer(value.toString());
                        while (itr.hasMoreTokens())
                        {
                                word.set(itr.nextToken());
                                context.write(word,one);
                        }
                }
        }
```

7.4.3 编写Reduce处理逻辑

- 在Reduce处理数据之前，Map的结果首先通过Shuffle阶段进行整理
- Reduce阶段的任务：对输入数字序列进行求和
- Reduce的输入数据为<key,Iterable容器>

Reduce任务的输入数据：
<”I”,<1,1>>
<”is”,1>
……
<”from”,1>
<”China”,<1,1,1>>

7.4.3 编写Reduce处理逻辑

```
public static class MyReducer extends
Reducer<Text,IntWritable,Text,IntWritable>{
                    private IntWritable result = new IntWritable();
                    public void reduce(Text key,
Iterable<IntWritable> values, Context context) throws
IOException,InterruptedException{
                        int sum = 0;
                        for (IntWritable val : values)
                        {
                            sum += val.get();
                        }
                        result.set(sum);
                        context.write(key,result);
                    }
    }
```

7.4.4 编写main方法

```
public static void main(String[] args) throws Exception{
        Configuration conf = new Configuration();  //程序运行时参数
        String[] otherArgs = new GenericOptionsParser(conf,args).getRemainingArgs();
        if (otherArgs.length != 2)
        {
            System.err.println("Usage: wordcount <in> <out>");
            System.exit(2);
        }
        Job job = new Job(conf,"word count");  //设置环境参数
        job.setJarByClass(WordCount.class);  //设置整个程序的类名
        job.setMapperClass(MyMapper.class);  //添加MyMapper类
        job.setReducerClass(MyReducer.class);  //添加MyReducer类
        job.setOutputKeyClass(Text.class);  //设置输出类型
        job.setOutputValueClass(IntWritable.class);  //设置输出类型
        FileInputFormat.addInputPath(job,new Path(otherArgs[0]));  //设置输入文件
        FileOutputFormat.setOutputPath(job,new Path(otherArgs[1])); //设置输出文件
        System.exit(job.waitForCompletion(true)?0:1);
    }
```

7.4.5 编译打包代码以及运行程序

包	功能
org.apache.hadoop.conf	定义了系统参数的配置文件处理方法
org.apache.hadoop.fs	定义了抽象的文件系统API
org.apache.hadoop.dfs	Hadoop分布式文件系统（HDFS）模块的实现
org.apache.hadoop.mapred	Hadoop分布式计算框架MapReduce的实现，包括任务的分发调度等
org.apache.hadoop.ipc	网络服务端和客户端的工具，封装了网络异步I/O的基础模块
org.apache.hadoop.io	定义了通用的I/O API，用于针对于网络、数据库、文件等数据对象进行读写操作等

7.4.5 编译打包代码以及运行程序

实验步骤：

- 使用java编译程序，生成.class文件
- 将.class文件打包为jar包
- 运行jar包
- 查看结果

本章小结

- 本章介绍了MapReduce编程模型的相关知识。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，并极大地方便了分布式编程工作，编程人员在不会分布式并行编程的情况下，也可以很容易将自己的程序运行在分布式系统上，完成海量数据集的计算
- MapReduce执行的全过程包括以下几个主要阶段：从分布式文件系统读入数据、执行Map任务输出中间结果、通过 Shuffle阶段把中间结果分区排序整理后发送给Reduce任务、执行Reduce任务得到最终结果并写入分布式文件系统。在这几个阶段中，Shuffle阶段非常关键，必须深刻理解这个阶段的详细执行过程
- 本章最后以一个单词统计程序为实例，详细演示了如何编写MapReduce程序代码以及如何运行程序

主讲教师

主讲教师：林子雨

单位：厦门大学计算机科学系
E-mail: ziyulin@xmu.edu.cn
个人网页：http://www.cs.xmu.edu.cn/linziyu
数据库实验室网站：http://dblab.xmu.edu.cn

扫一扫访问个人主页

林子雨，男，1978年出生，博士（毕业于北京大学），现为厦门大学计算机科学系助理教授（讲师），曾任厦门大学信息科学与技术学院院长助理、晋江市发展和改革局副局长。中国高校首个“数字教师”提出者和建设者，厦门大学数据库实验室负责人，厦门大学云计算与大数据研究中心主要建设者和骨干成员，2013年度厦门大学奖教金获得者。主要研究方向为数据库、数据仓库、数据挖掘、大数据、云计算和物联网，编著出版中国高校第一本系统介绍大数据知识的专业教材《大数据技术原理与应用》并成为畅销书籍，编著并免费网络发布40余万字中国高校第一本闪存数据库研究专著《闪存数据库概念与技术》；主讲厦门大学计算机系本科生课程《数据库系统原理》和研究生课程《分布式数据库》《大数据技术基础》。具有丰富的政府和企业信息化培训经验，曾先后给中国移动通信集团公司、福州马尾区政府、福建省物联网科学研究院、石狮市物流协会、厦门市物流协会等多家单位和企业开展信息化培训，累计培训人数达2000人以上。

大数据学习教材推荐

《大数据技术原理与应用——概念、存储、处理、分析与应用》，由厦门大学计算机科学系林子雨博士编著，是中国高校第一本系统介绍大数据知识的专业教材。

全书共有13章，系统地论述了大数据的基本概念、大数据处理架构Hadoop、分布式文件系统HDFS、分布式数据 库HBase、NoSQL数据库、云数据库、分布式并行编程模型MapReduce、流计算、图计算、数据可视化以及大数据在互联网、生物医学和物流等各个领域的应用。在Hadoop、HDFS、HBase和MapReduce等重要章节，安排了入门级的实践操作，让读者更好地学习和掌握大数据关键技术。

本书可以作为高等院校计算机专业、信息管理等相关专业的大数据课程教材，也可供相关技术人员参考、学习、培训之用。

欢迎访问《大数据技术原理与应用——概念、存储、处理、分析与应用》教材官方网站：
http://dblab.xmu.edu.cn/post/bigdata

扫一扫访问教材官网

课程建设单位

厦門大學数据库实验室
Database Lab of Xiamen University

厦門大學 云计算与大数据研究中心
XIAMEN UNIVERSITY Center for Cloud Computing and Big Data

海峡云计算与大数据应用研究中心
Strait Cloud Computing and Big Data Application Research Center


Thank You!