MapReduce原理

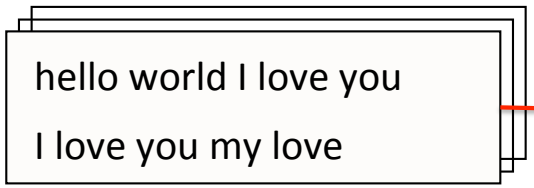
刘军（liujun@bupt.edu.cn）
北京邮电大学 数据科学中心

每周新鲜事

2.0.1稳定版发布

从WordCount开始

如何实现以下程序？



Input

单词	计数
hello	1
I	2
love	3
my	1
world	1
you	2

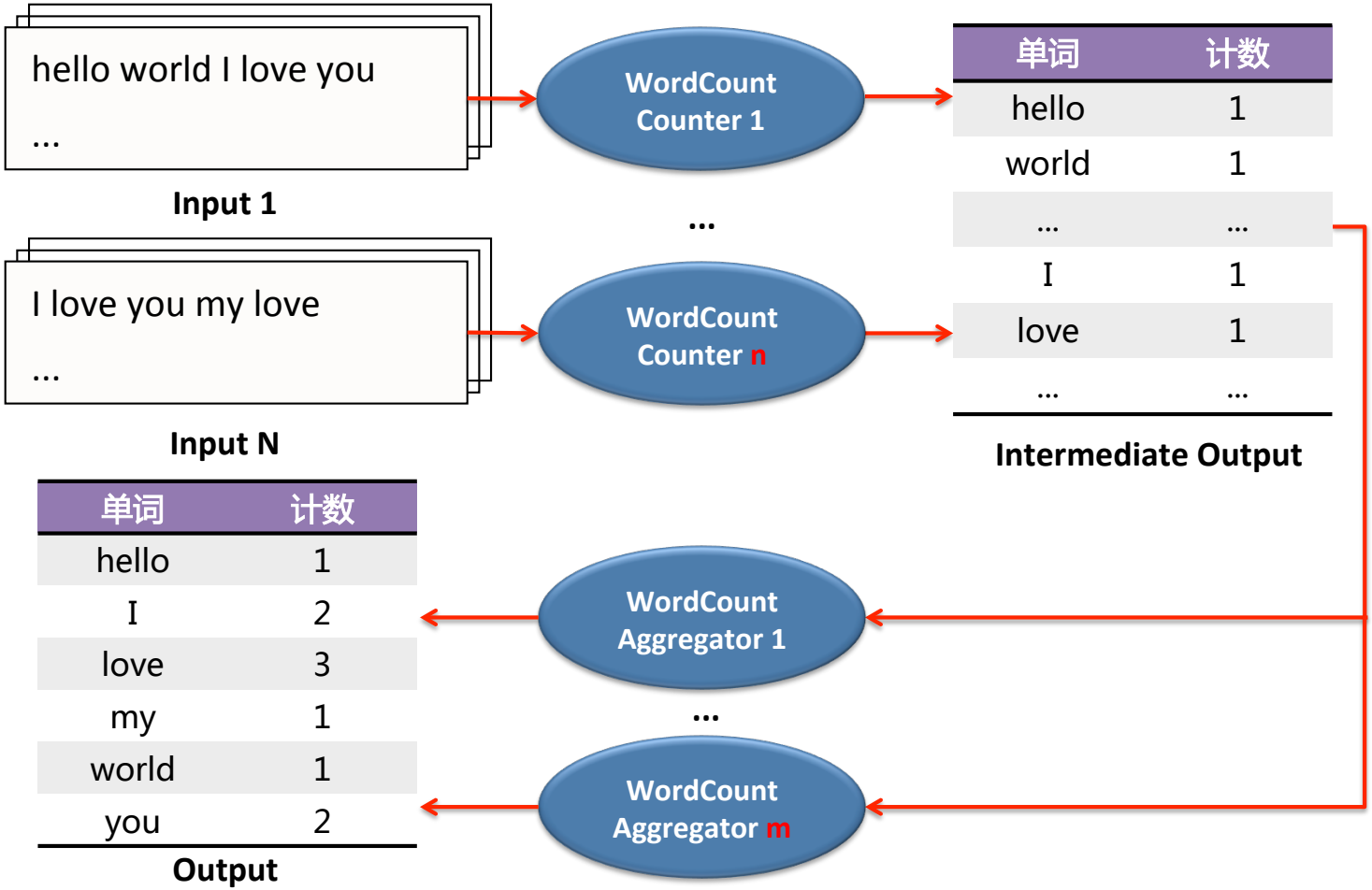
Output

朴素的串行实现

```
1:  define wordCount
2:  for each doc in docSet
3:      words = tokenize(doc)
4:      for each word in words
5:          wordCount[word] ++
6:      end for
7:  end for
8:  output(wordCount)
```

- 串行代码的问题：
  - 海量数据
  - 运行效率
- 怎么办？

并行WordCount – 思路



并行WordCount – 代码

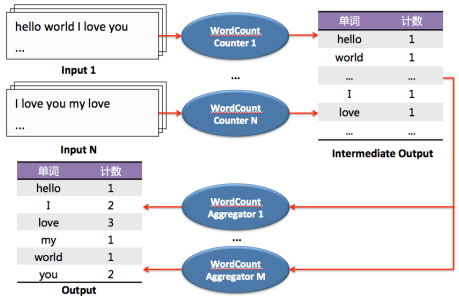
Counter

```
1:  define wordCount
2:  for each doc in docSet
3:    words = tokenize(doc)
4:    for each word in words
5:      wordCount[word] ++
6:    end for
7:  end for
8:  send (wordCount, Aggregator)
```

Aggregator

```
1:  define totalCount
2:  for each wordCount from Counter
3:    for each word in wordCount
4:      totalCount[word] += wordCount[word]
5:    end for
6:  end for
7:  output(totalCount)
```

并行WordCount – 架构

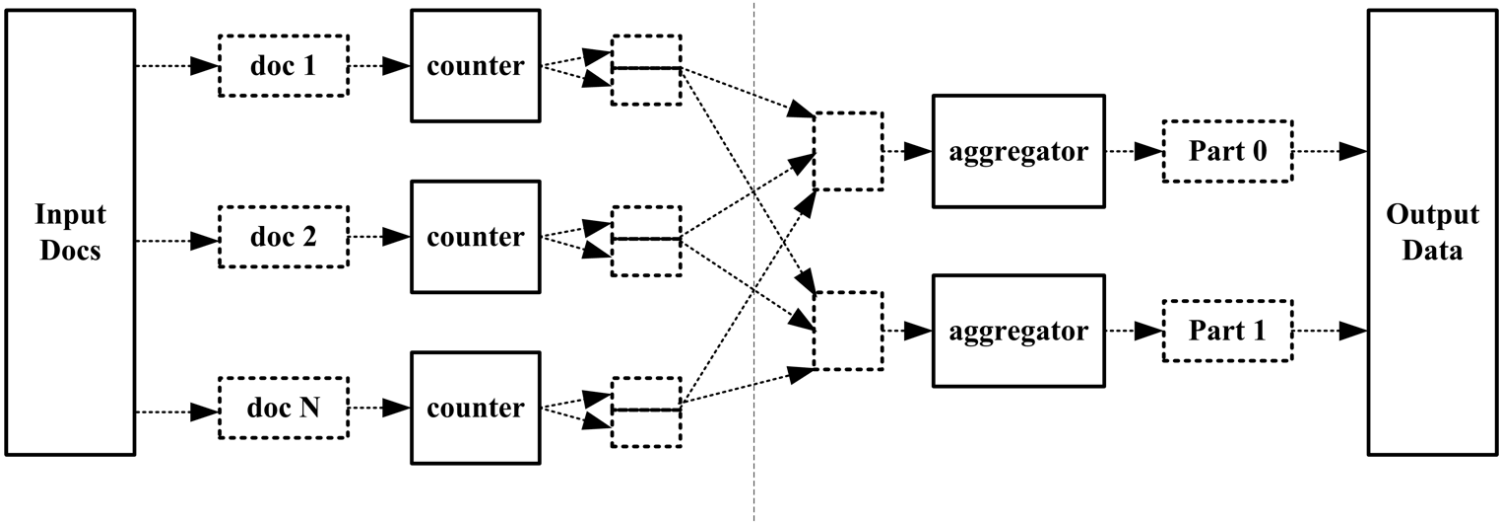


Counter

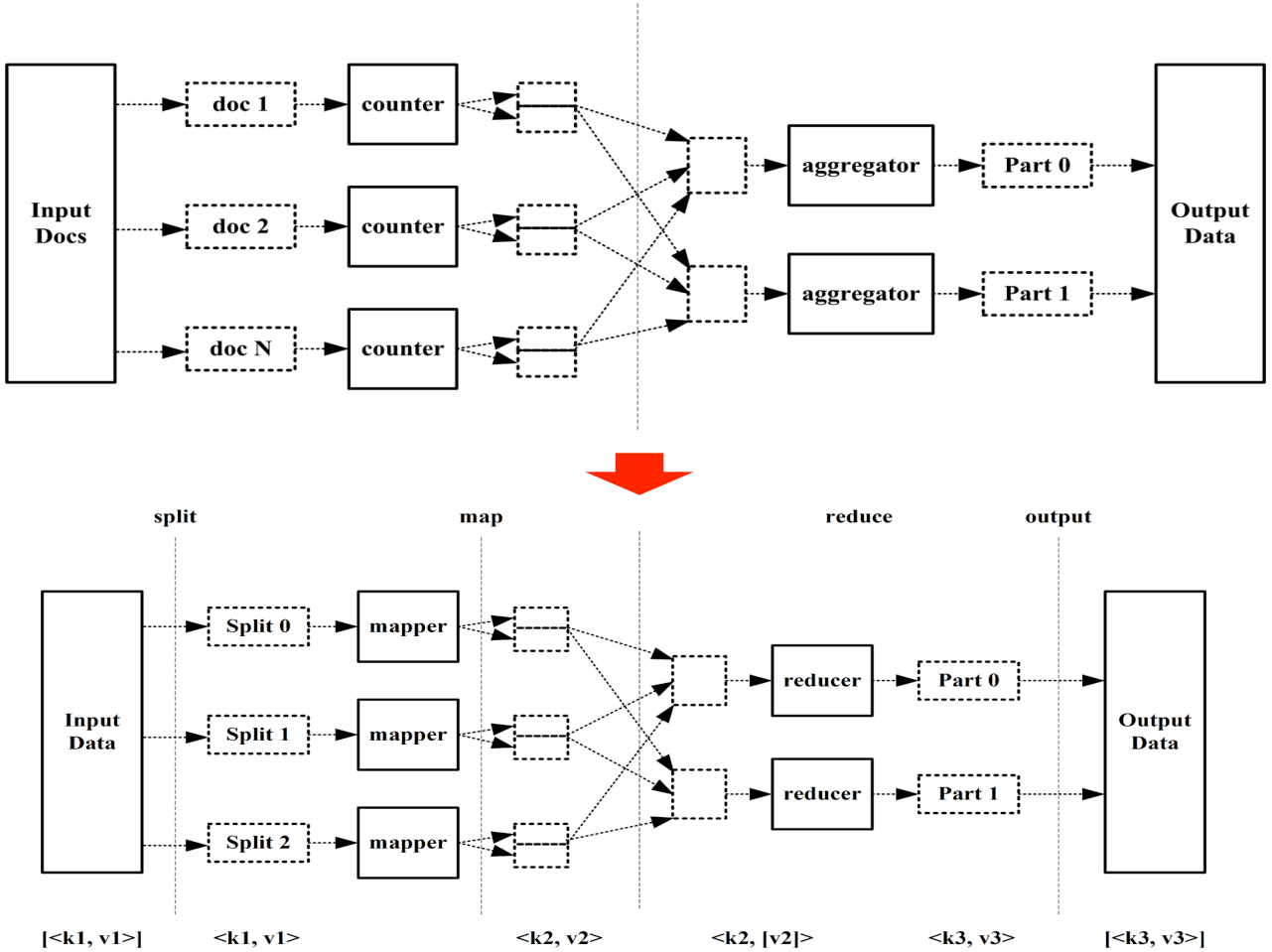
```
1: define wordCount
2: for each doc in docSet
3:   words = tokenize(doc)
4:   for each word in words
5:     wordCount[word] ++
6:   end for
7: end for
8: send (wordCount, Aggregator)
```

Aggregator

```
1: define totalCount
2: for each wordCount from Counter
3:   for each word in wordCount
4:     totalCount[word] += wordCount[word]
5:   end for
6: end for
7: output(totalCount)
```



从并行WordCount到MapReduce



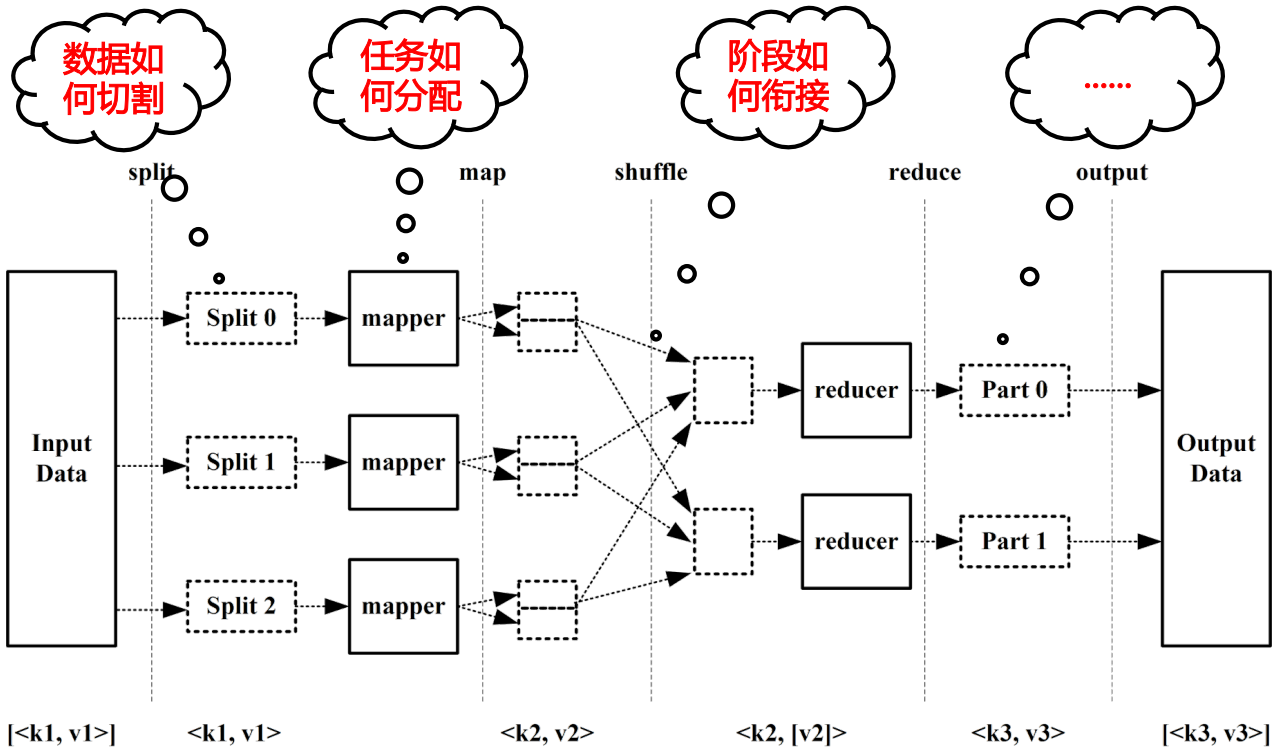
MapReduce版的WordCount

```
public static class TokenizerMapper 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);
} } }
```

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

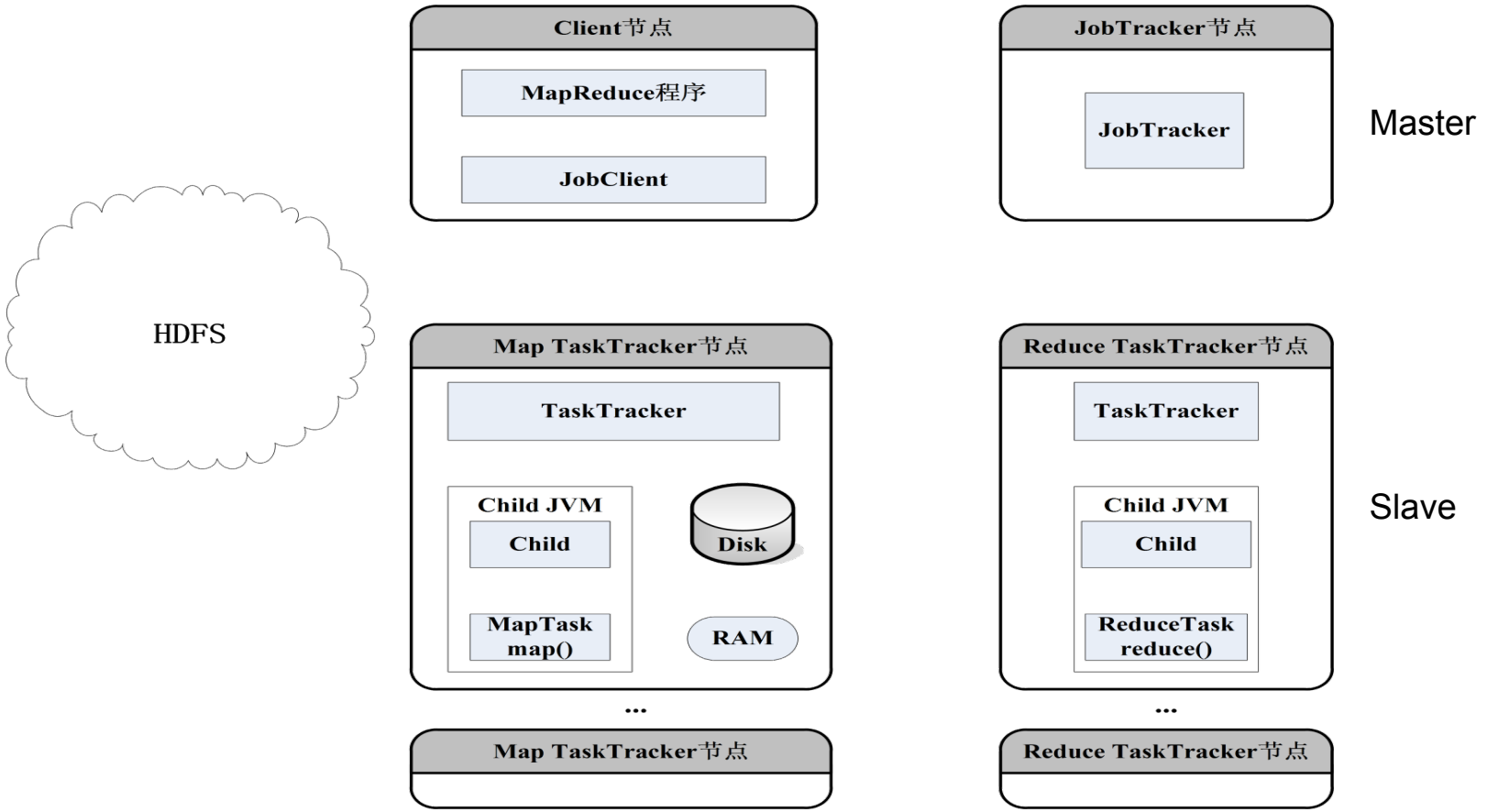
```
public static void main(String[] args) throws Exception{
  Configuration conf = new Configuration();
  String[] otherArgs = new GenericOptionsParser(
    conf, args).getRemainingArgs();
  if (otherArgs.length != 2) {
    System.err.println("Usage: wordcount <in> <out>");
    System.exit(2);
  }
  Job job = new Job(conf, "word count");
  job.setJarByClass(WordCount.class);
  job.setMapperClass(TokenizerMapper.class);
  job.setCombinerClass(IntSumReducer.class);
  job.setReducerClass(IntSumReducer.class);
  job.setOutputKeyClass(Text.class);
  job.setOutputValueClass(IntWritable.class);
  FileInputFormat.addInputPath(job,
    new Path(otherArgs[0]));
  FileOutputFormat.setOutputPath(job,
    new Path(otherArgs[1]));
  System.exit(job.waitForCompletion(true) ? 0 : 1);
}
```

要解决的问题

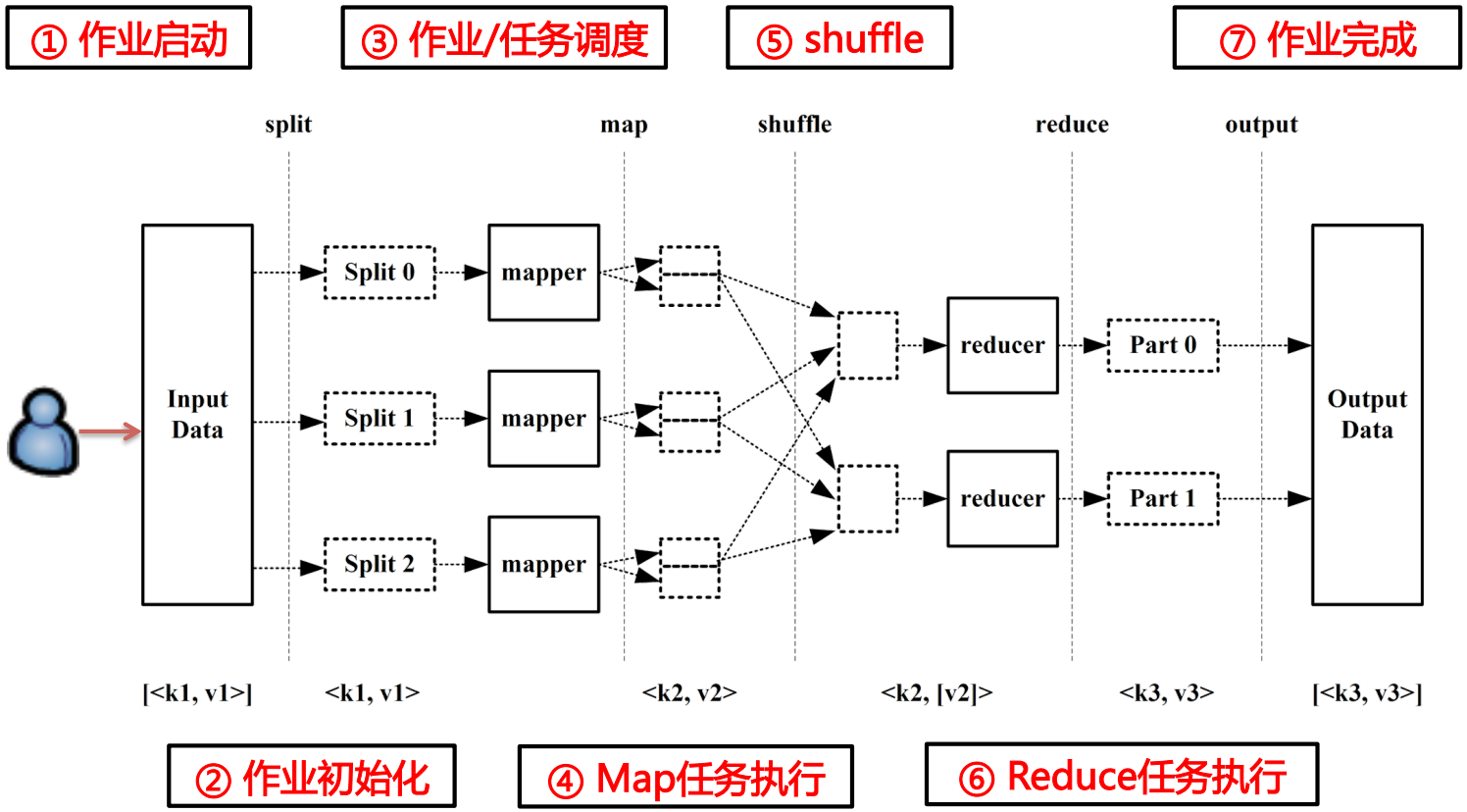

数据如
何切割
任务如
何分配
阶段如
何衔接
.....
split
map
shuffle
reduce
output
Input
Data
Split 0
Split 1
Split 2
mapper
mapper
mapper
reducer
reducer
Part 0
Part 1
Output
Data
[<k1, v1>]
<k1, v1>
<k2, v2>
<k2, [v2]>
<k3, v3>
[<k3, v3>]

MapReduce原理

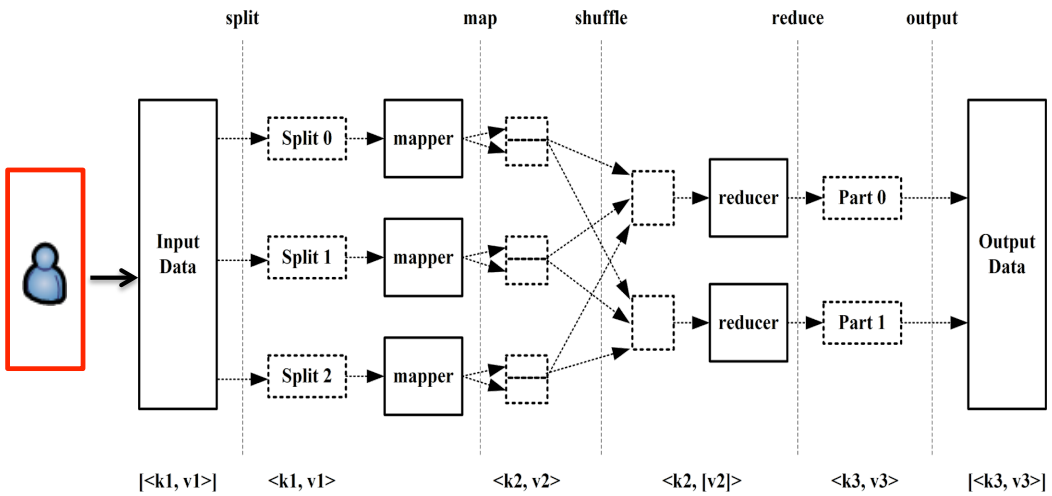
MapReduce计算框架中的各个实体



MapReduce作业的运行流程



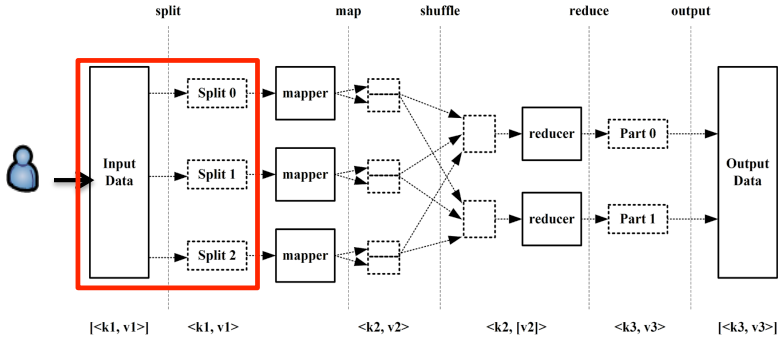
1. 作业启动

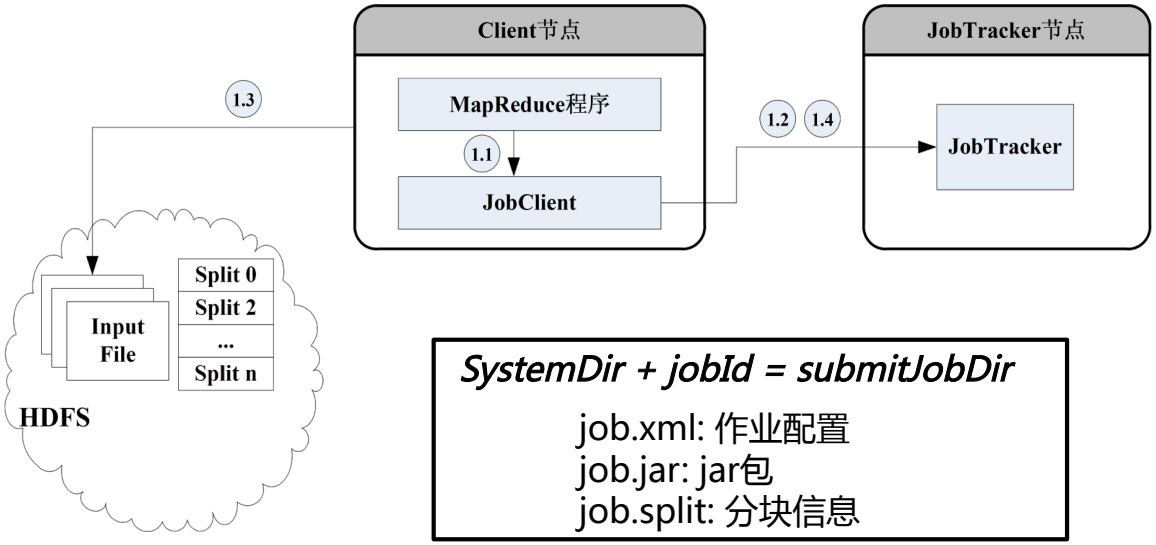


./hadoop jar ./../hadoop-examples-1.0.3.jar wordcount in out

```
public static void main(String[] args) throws Exception{
 Configuration conf = new Configuration();
 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");
 job.setJarByClass(WordCount.class);
 job.setMapperClass(TokenizerMapper.class);
 job.setCombinerClass(IntSumReducer.class);
 job.setReducerClass(IntSumReducer.class);
 job.setOutputKeyClass(Text.class);
 job.setOutputValueClass(IntWritable.class);
 FileInputFormat.addInputPath(job,
  new Path(otherArgs[0]));
 FileOutputFormat.setOutputPath(job,
  new Path(otherArgs[1]));
 System.exit(job.waitForCompletion(true) ? 0 : 1); }
```

2. 作业初始化 - 创建作业



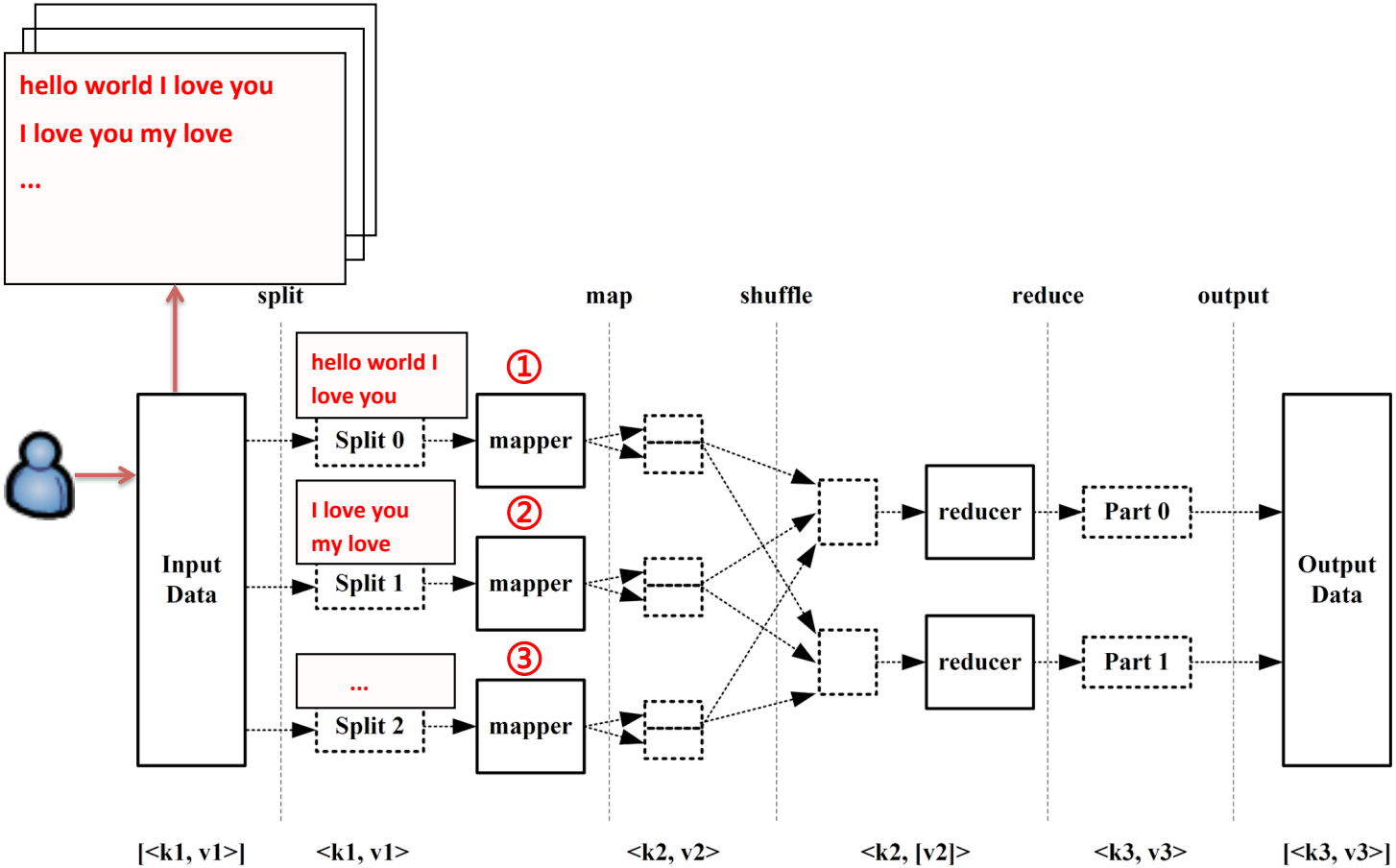


SystemDir + jobId = submitJobDir

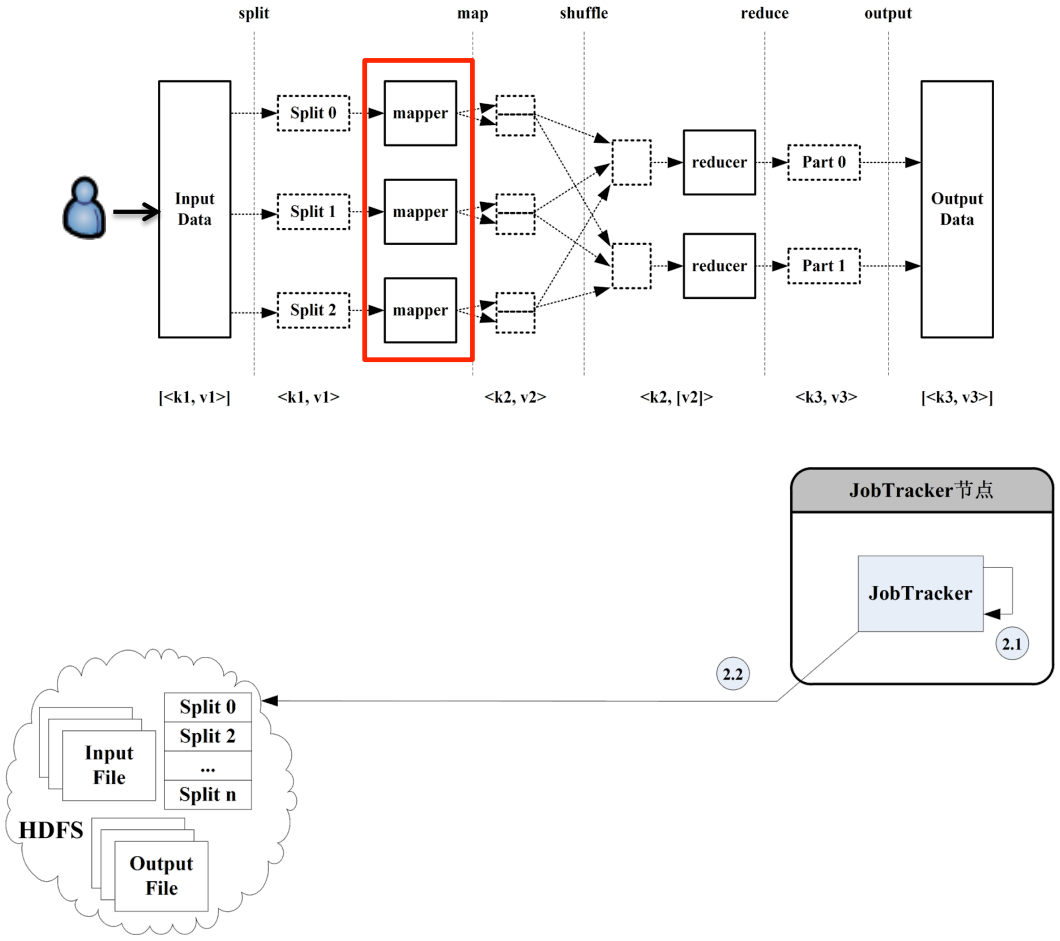
job.xml: 作业配置
job.jar: jar包
job.split: 分块信息

1. MapReduce程序创建新的JobClient实例
2. JobClient向JobTracker请求获得一个新的JobId标识本次作业
3. JobClient将运行作业需要的相关资源放入作业对应的HDFS目录、计算分片数量和map任务数量
4. 向JobTracker提交作业，并获得作业的状态对象句柄

2. 作业初始化 - 切分数据

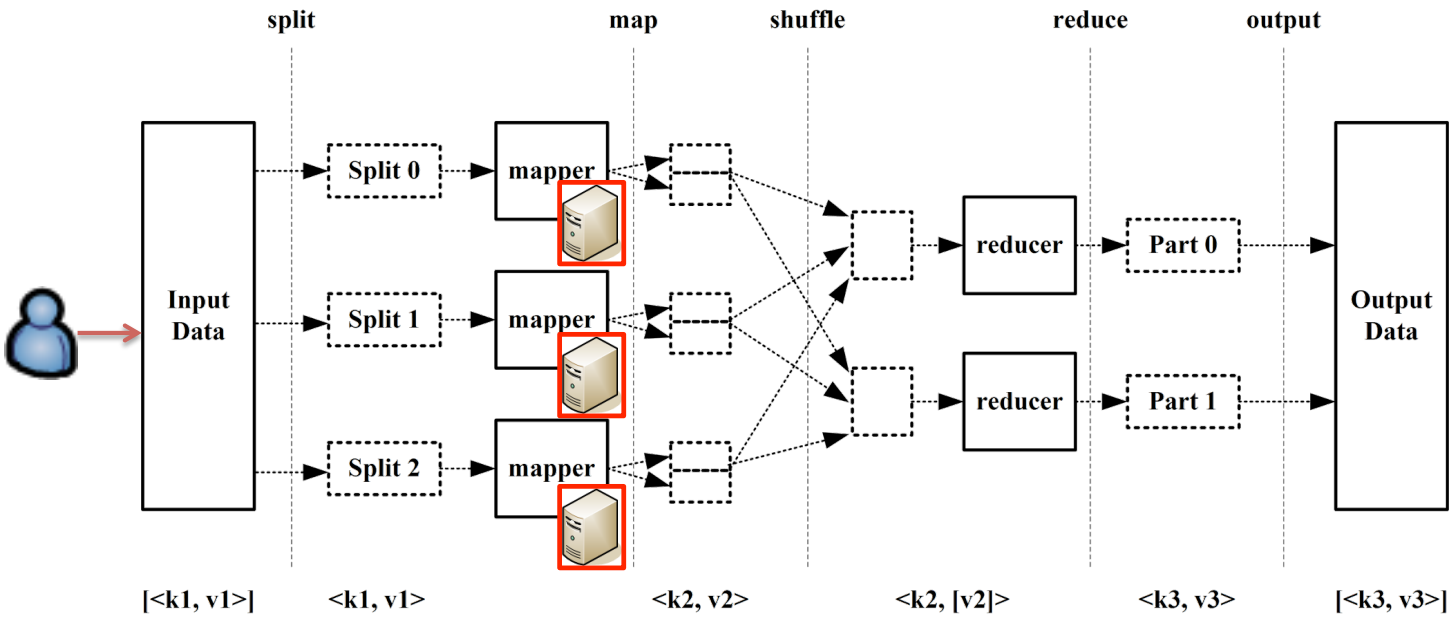

hello world I love you
I love you my love
...
split
map
shuffle
reduce
output
Input Data
hello world I love you
Split 0
I love you my love
Split 1
...
Split 2
①
②
③
mapper
mapper
mapper
reducer
reducer
Part 0
Part 1
Output Data
[<k1, v1>]
<k1, v1>
<k2, v2>
<k2, [v2]>
<k3, v3>
[<k3, v3>]

3. 作业（Job）/任务（Task）调度

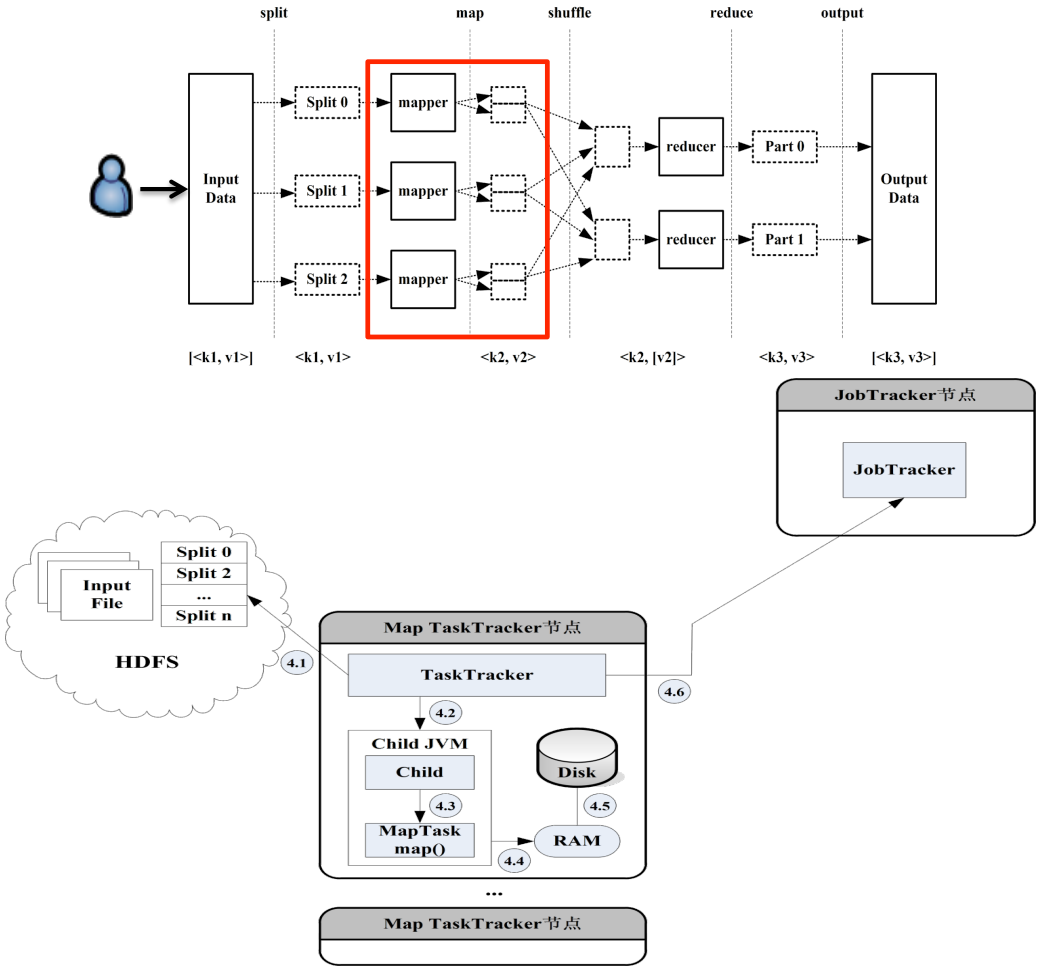


1. 作业提交请求放入队列等待调度，JobTracker执行调度
2. JobTracker从HDFS中取出作业分片信息，创建对应数量的TaskInProgress调度和监控Map任务

3. 作业（Job）/任务（Task）调度完成后

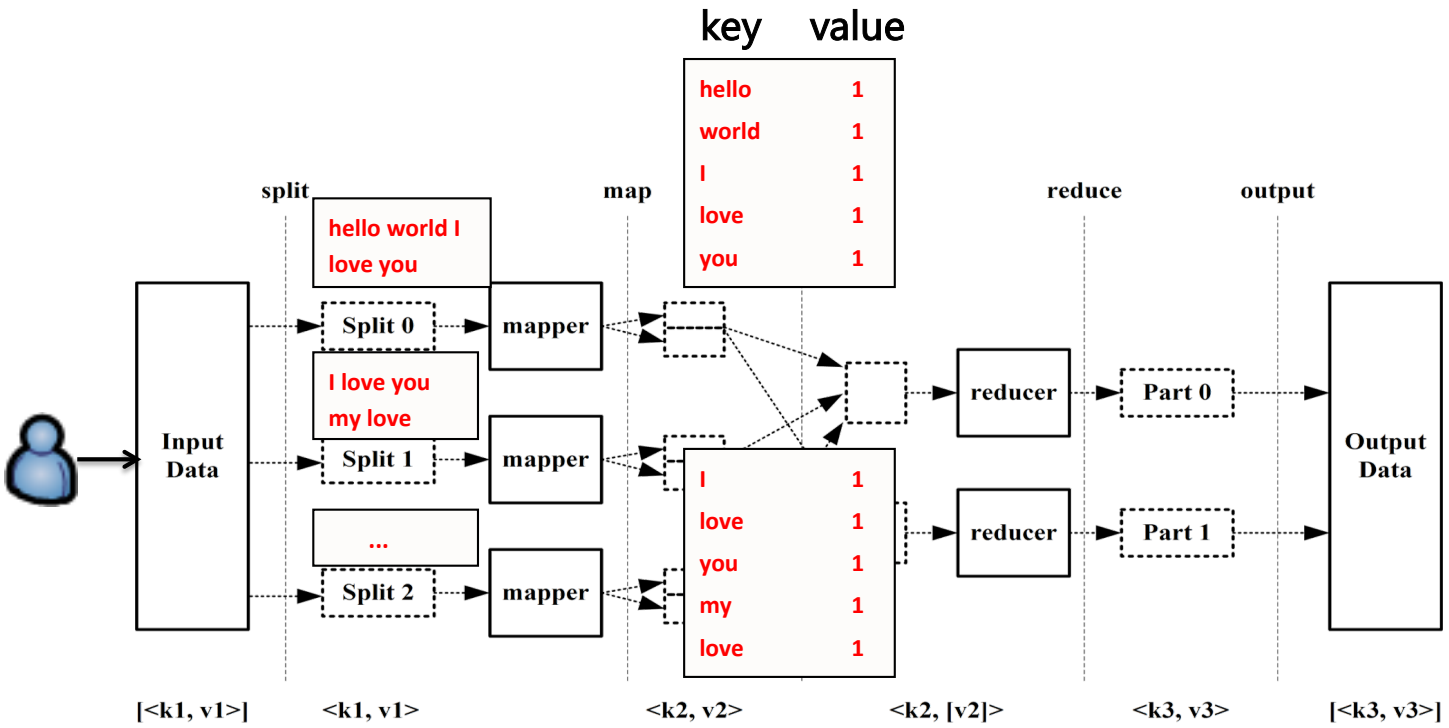


4. Map

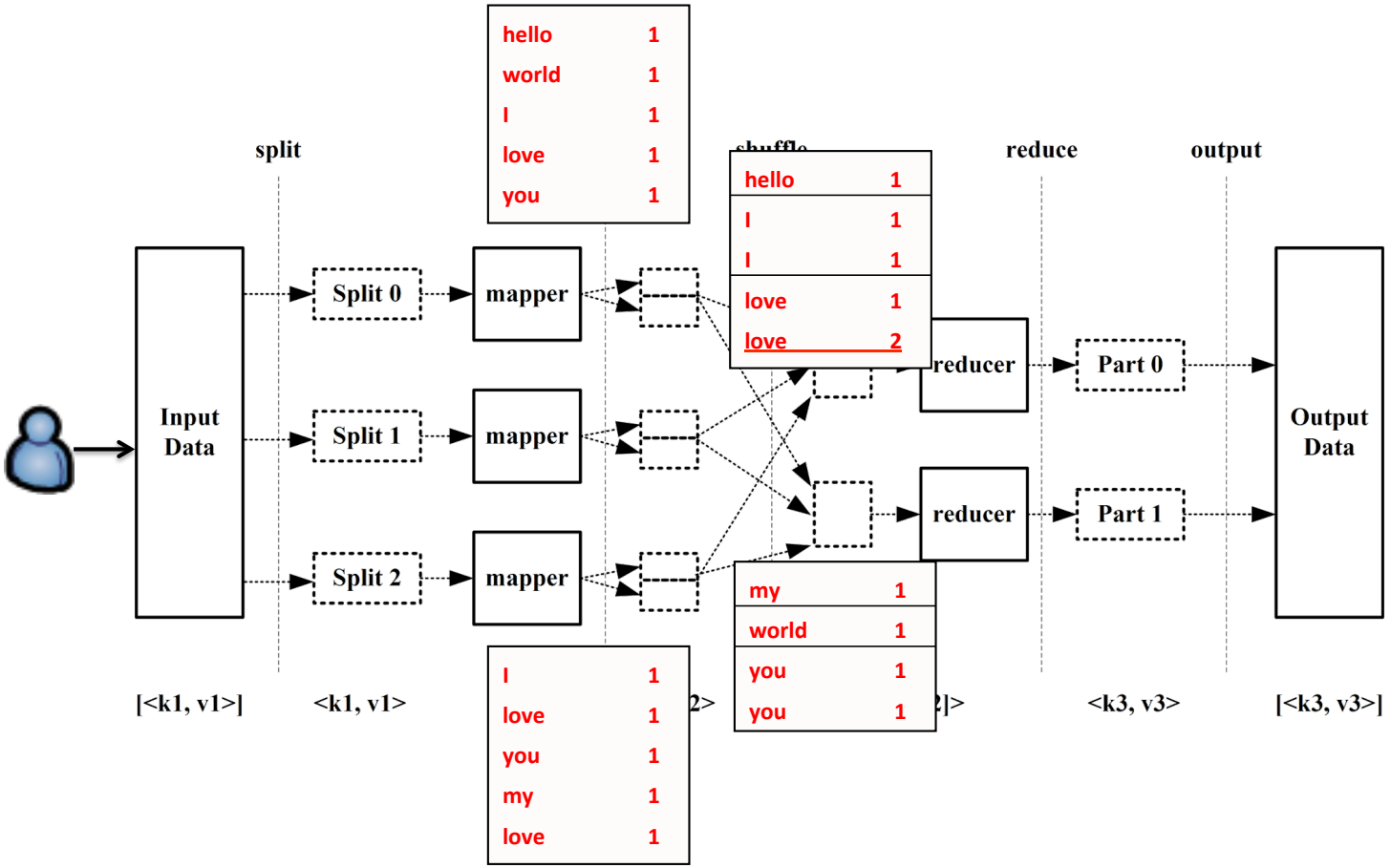


1. 从HDFS提取相关资源（Jar包、数据）
2. 创建TaskRunner运行Map任务
3. 在单独的JVM中启动MapTask执行map函数
4. 中间结果数据定期存入缓存
5. 缓存写入磁盘
6. 定期报告进度

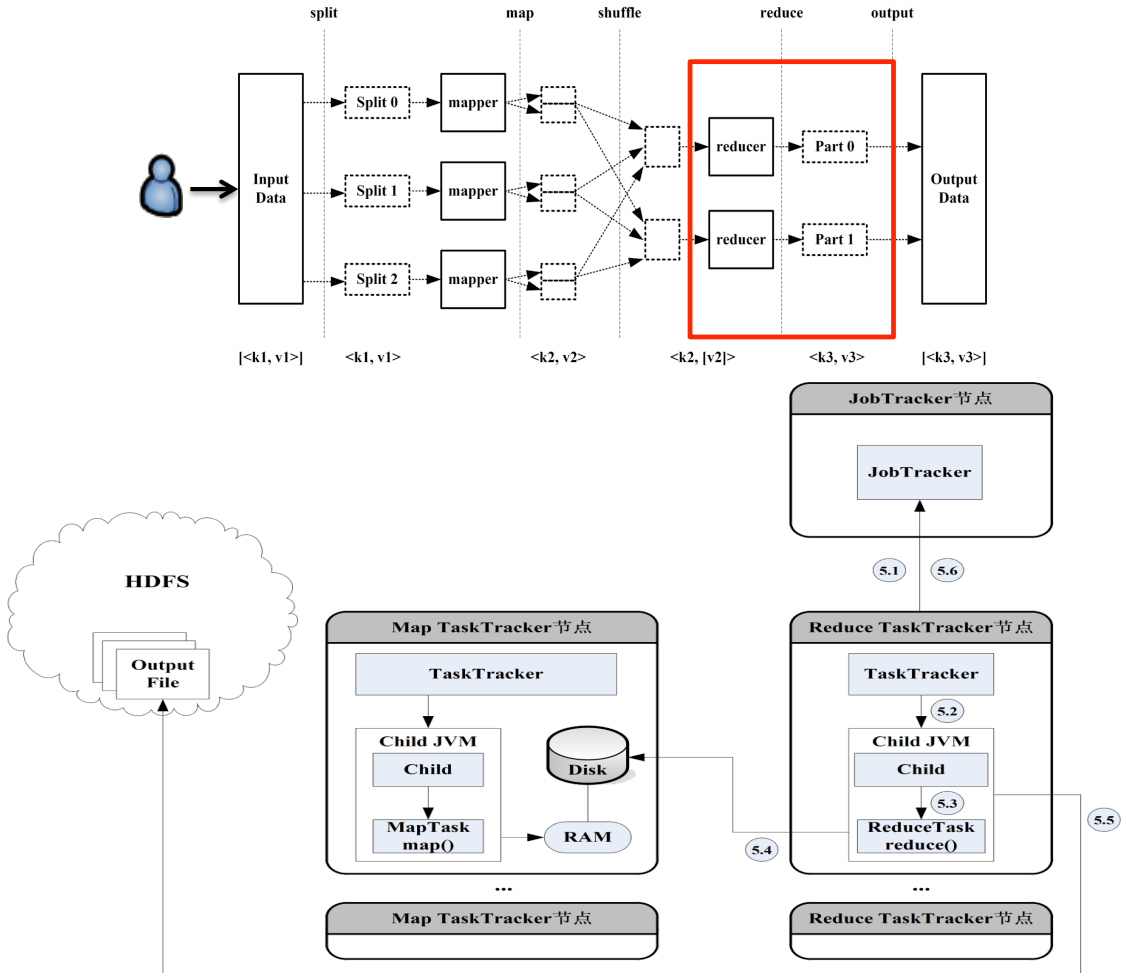
4. Map完成后


key value
hello 1
world 1
I 1
love 1
you 1
split
map
reduce
output
hello world I
love you
Split 0
mapper
I love you
my love
Split 1
mapper
...
Split 2
mapper
Input Data
I 1
love 1
you 1
my 1
love 1
reducer
reducer
Part 0
Part 1
Output Data
[<k1, v1>]
<k1, v1>
<k2, v2>
<k2, [v2]>
<k3, v3>
[<k3, v3>]

5. Shuffle

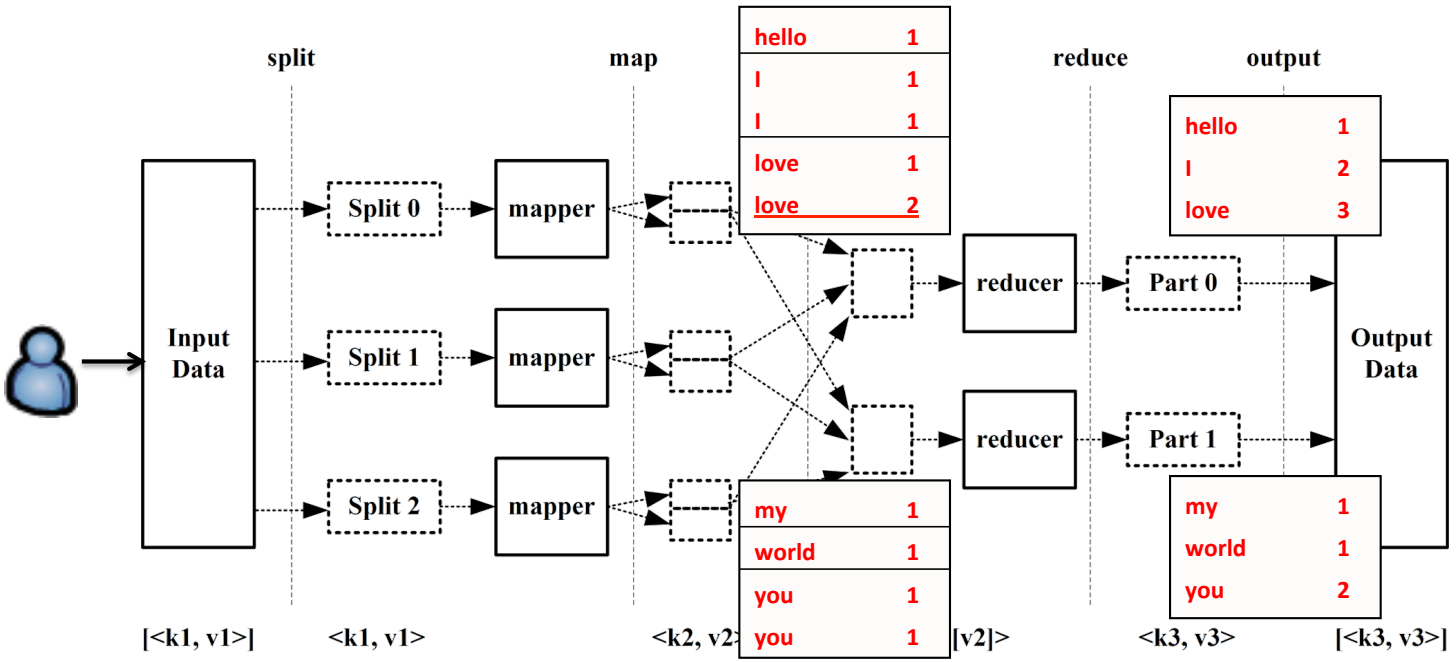

split
reduce
output
Input Data
Split 0
Split 1
Split 2
mapper
mapper
mapper
hello 1
world 1
I 1
love 1
you 1
I 1
love 1
you 1
my 1
love 1
hello 1
I 1
I 1
love 1
love 2
my 1
world 1
you 1
you 1
reducer
reducer
Part 0
Part 1
Output Data
[<k1, v1>]
<k1, v1>
<k3, v3>
[<k3, v3>]

6. Reduce

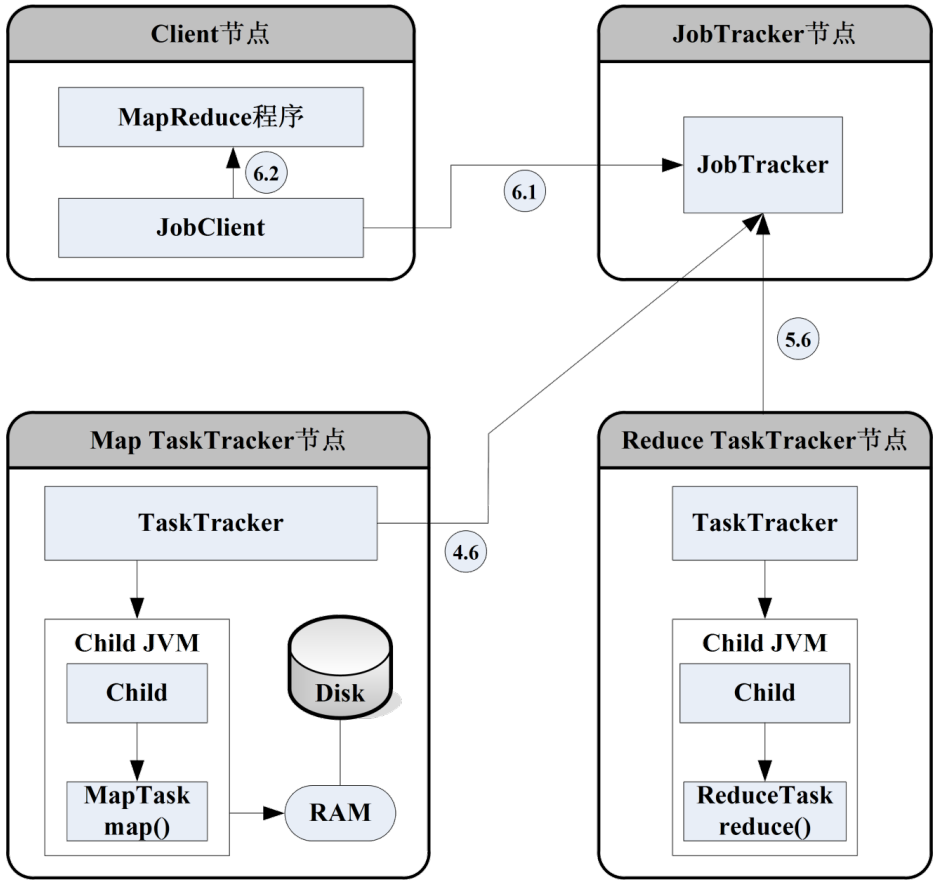


1. 分配Reduce任务
2. 创建TaskRunner运行Reduce任务
3. 在单独的JVM中启动ReduceTask执行reduce函数
4. 从Map节点下载中间结果数据
5. 输出结果临时文件
6. 定期报告进度

6. Reduce完成后


split
map
reduce
output
Input Data
Split 0
Split 1
Split 2
mapper
mapper
mapper
hello 1
I 1
I 1
love 1
love 2
my 1
world 1
you 1
you 1
reducer
reducer
Part 0
Part 1
hello 1
I 2
love 3
my 1
world 1
you 2
Output Data
[<k1, v1>]
<k1, v1>
<k2, v2>
[v2]>
<k3, v3>
[<k3, v3>]

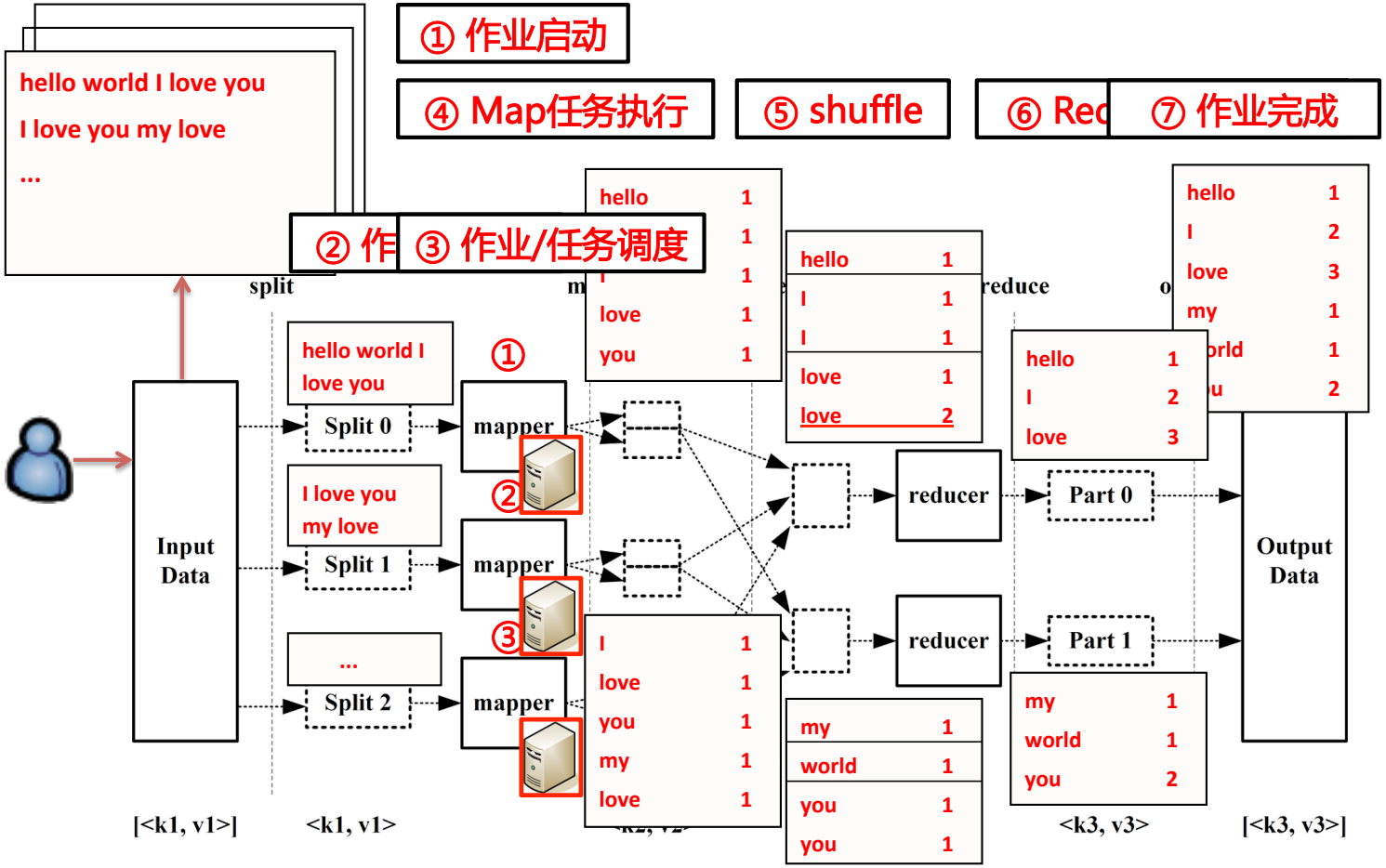
7. 作业完成



Map和Reduce执行时定时更新进度

1. JobClient轮询获知任务完成
2. 通知用户

重温MapReduce完整过程



下周课前请准备（实践课）

- 练习实践（尽力课下全部完成）：
  - HDFS上access.log文件（/home/hadooptest/Hadoop_2016_Autumn）
  - 编写MapReduce程序统计每个用户访问每个SP的次数、上行、下行流量
    - ✓ 在Git创建以自己名字汉语拼音全拼的分支，将代码和结果提交
    - ✓ 在wiki页面，提交运行结果图，及必要的文字说明

- （剩余工作）下周课上一起完成

刘军
北京邮电大学 信通院 宽带网络监控教研中心
北邮本部 明光楼七层
邮件地址：liujun@bupt.edu.cn
新浪微博：北邮刘军
电　　话：010-62283742