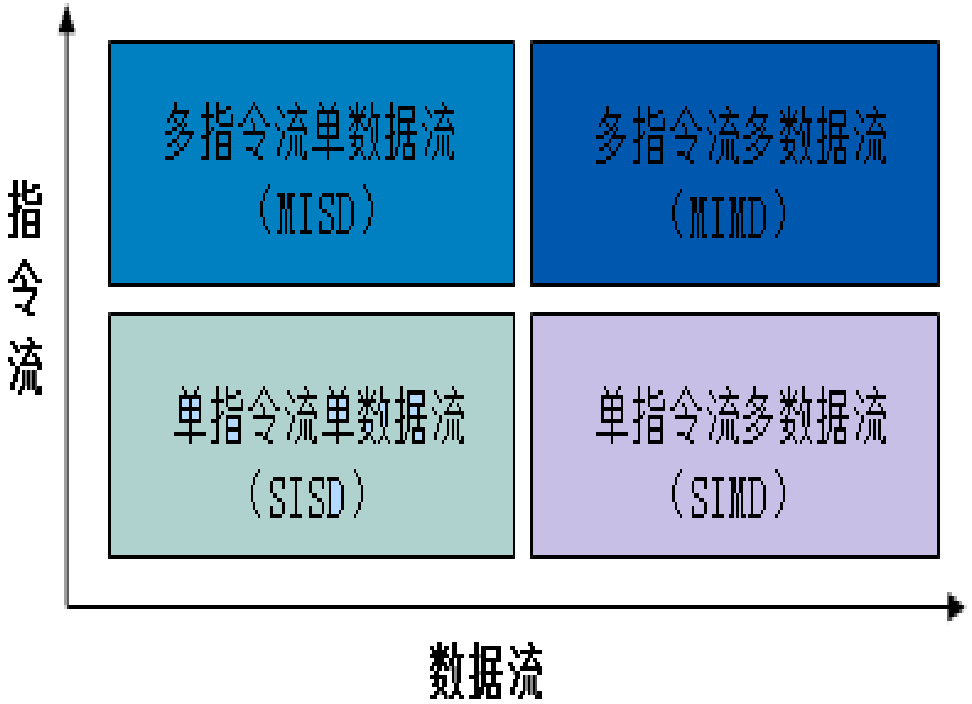
11

MapReduce计算模型

11.1 分布式并行计算系统

- 按照指令流和数据流划分：
  - 单指令流单数据流
  - 单指令流多数据流
  - 多指令流单数据流
  - 多指令流多数据流



11.1 分布式并行计算系统

- 多指令流多数据流（MIMD）
- 按照处理器是否共享内存划分：

• 多处理器共享内存机器

UMA架构和NUMA架构

• 多计算机独立内存体系

MPP和集群两种计算架构。

• 归属于MIMD体系的计算机：

并行向量处理机、对称多处理机、大规模并行处理机、工作站机群、分布式共享存储处理机

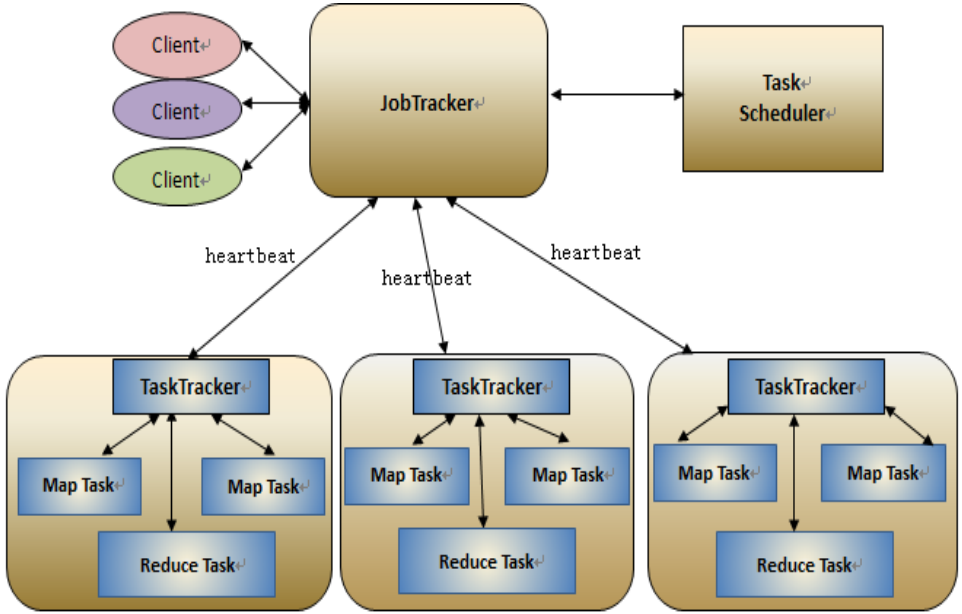
11.2 MapReduce计算架构

- 软件模块/计算单元
- MapReduce本质思想——分治法：

- 将大数据集划分为小数据集
- 小数据集划分为更小数据集
- 将最终划分的小数据分布到集群节点上
- 以并行方式完成计算处理
- 将计算结果递归融汇，得到最后结果

11.2 MapReduce计算架构

- MapReduce四大组件：
  - Client
  - JobTracker
  - TaskTracker
  - Task



11.2 MapReduce计算架构

- MapReduce主要任务——映射与简化：
- Map（映射）：负责输入数据的分片、转化、处理，输出中间结果文件；
- Reduce（简化）：以Map的输出文件为输入，对中间结果进行合并处理，得到最终结果并写入HDFS。
- 两类任务都有多个进程运行在DataNode上，相互间通过Shuffle阶段交换数据

11.2 MapReduce计算架构

- 键值对与输入格式

  MapReduce是以键值对（key-value pair）格式来完成数据计算处理的

- 键是行键（RowKey）

  多半用作索引（indexing）

- 值是字符串（character string）或二进制数组（binary string）形式

  包含存储数据或信息。

11.2 MapReduce计算架构

- 文件分片——定义

- 把大数据文件进行分片，生成一个个 InputSplit （简称为 split ）
- 一个 InputSplit 对应一个计算任务（ task ），分配到计算节点，由 map/reduce 进程执行计算处理
- split是我们对数据文件出于计算需要的逻辑划分单位，但一个 HDFS 文件在集群中实际是以块（ block ）的物理形式存储的。 ——Split vs block?

11.2 MapReduce计算架构

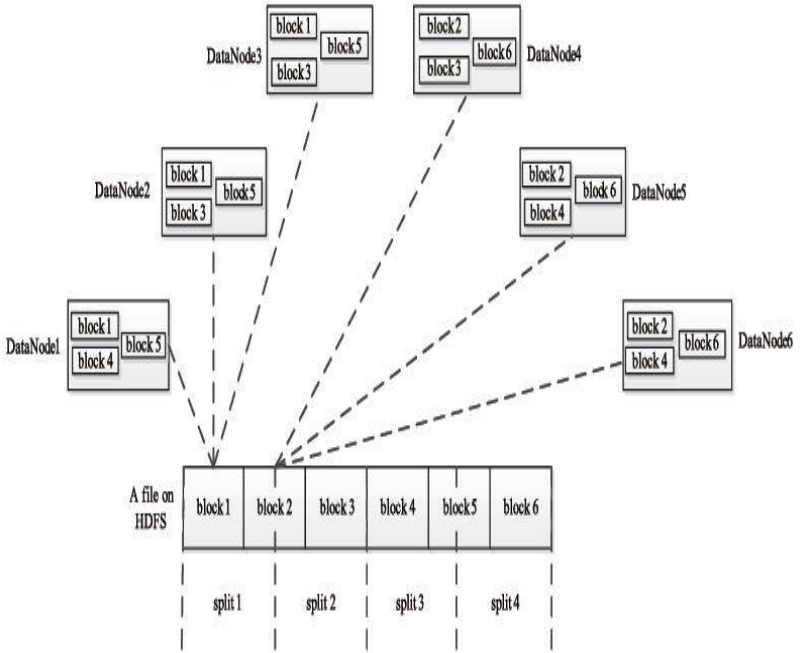
- Split与Block：
- Block

  一个HDFS文件可以按block形式进行物理存储

  HDFS的物理存储单元
- Split

  一个逻辑上对HDFS文件的划分方式

  MapReduce的计算逻辑单元



11.2 MapReduce计算架构

- Map数目设置
- 相关参数:
  - block_size : HDFS文件的block size
  - total_size : 输入文件整体的大小
  - input_file_num : 输入文件个数

11.2 MapReduce计算架构

- 计算流程：

1）使用默认map数

如果不进行任何设置，默认的map数由blcok_size决定：

default_num = total_size / block_size;

2）预设map数目

可通过参数mapred.map.tasks来设置期望的map数目，但是这个数只有在大于default_num的时候才会生效：

goal_num = mapred.map.task

11.2 MapReduce计算架构

- 计算流程：

3）设置分片大小（split size）

可以通过mapred.min.split.size 设置每个task处理的split的大小，但是这个大小只有在大于block_size的时候才会生效。

split_size = max(mapred.min.split.size，block_size);

split_num = total_size / split_size;

4）计算map数目

compute_map_num = min(split_num，max(default_num, goal_num))

5）每一个map处理的分片是不能跨越文件的，所以，最终的map个数应该为：

final_map_num = max(compute_map_num, input_file_num)

11.2 MapReduce计算架构

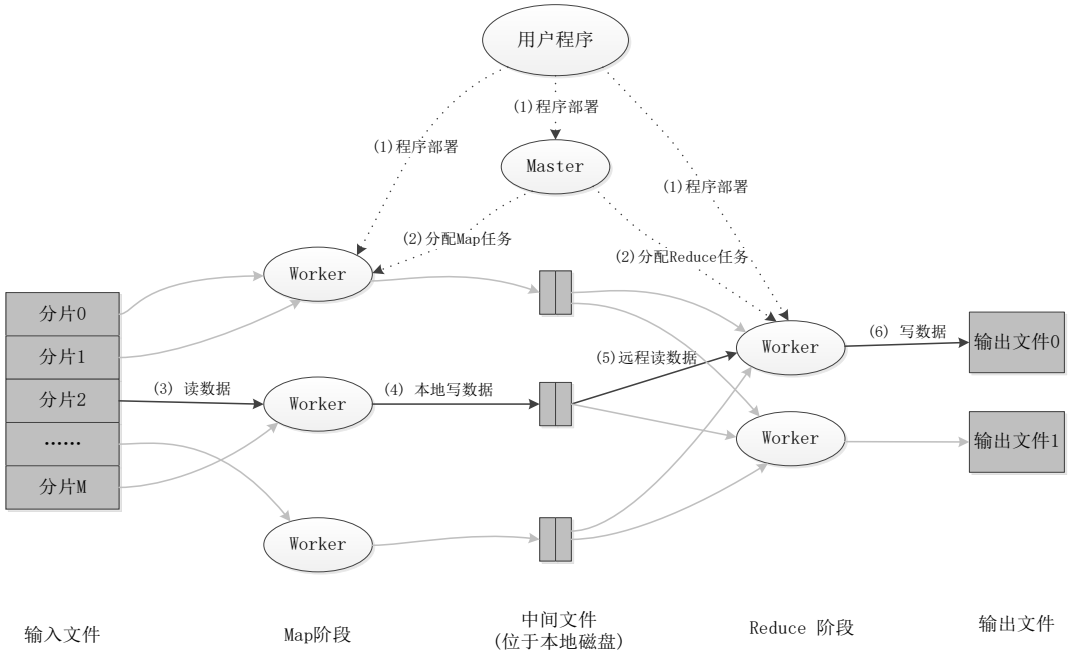
- Map数目设置准则：
- 增加map个数→设置mapred.map.tasks 为一个较大的值；
- 减小map个数→设置mapred.min.split.size 为一个较大的值；
- 输入中有很多小文件，依然想减少map数目→需将小文件merge为大文件，然后使用第二点准则
- 输入格式处理：
- 将不同格式数据转换为键值表的步骤
- 基础类InputFormat类：
  - 选择作为输入的文件或对象
  - 提供把文件分片的InputSplits（）方法
  - 为RecordReader读取文件提供一个工厂方法

11.2 MapReduce计算架构

输入格式	描述	键	值
TextInputFormat	默认格式，读取文件的行	行的字节偏移量	行字符串内容
KeyValueInputFormat	把行解析为键值对	第一个tab字符前的所有字符	行剩下的字符串内容
SequenceFileInputFormat	Hadoop定义的二进制格式	用户自定义	用户自定义
SequenceFileAsTextInputFormat	是SequenceFileInputFormat的变体，它将键和值的顺序值转换为text。转换的时候会调用键和值的toString方法。这个格式可以是顺序文件作为流操作的输入。	转换后的键字符串	转换后的值字符串
SequenceFileAsBinaryInputFormat	SequenceFileAsBinaryInputFormat是SequenceFileInputFormat的另一种变体，它将顺序文件的二进制格式键和值封装为BytesWritable对象，应用程序可以任意地将这些字节数组解释为需要的类型。		
DBInputFormat	DBInputForma是一个使用JDBC并且从关系数据库中读取数据的一种输入格式。由于它没有任何碎片技术，所以在访问数据库的时候必须非常小心，太多的mapper可能会事数据库受不了。因此DBInputFormat最好在加载小量数据集的时候用。		

11.2 MapReduce计算架构

- Map/Shaffle/Reduce任务与实现
- MapReduce工作流程：



11.2 MapReduce计算架构

- MapReduce主要三个工作阶段：
- Map（映射）

  Mapper执行map task，将输出结果写入中间文件
- Shuffle（归并）

  把Mapper的输出数据归并整理后分发给Reducer处理，包括merge，combine，sort和partition几个步骤；
- Reduce（化简）

  Reducer执行reduce task，将最后结果写入HDFS。

11.2 MapReduce计算架构

- Map（映射）阶段

- 对于每一个split，系统都生成一个map task，调用Mapper来执行，将读入数据转换成键值对格式
- 完成计算处理后，将输出结果写入中间文件
- 一个Map任务可以在集群的任何计算节点上运行
- 多个Map任务可以并行地运行在集群上
- Mapper的输入数据来源：MapContext
- MapContext的实现依赖于：MapContextImpl
- MapContextImpl内部组合：InputSplit和RecordReader
  - 提供了读取和封装输入数据键值对（key, value）的方法。

11.2 MapReduce计算架构

- Shuffle（归并）阶段
- 主要任务
  - 将每个map task的输出结果进行归并、排序、然后按照一定的规则分发给Reducer去执行化简步骤
- Shuffle任务实际上涉及到Map阶段的输出，以及Reduce阶段的输入
- Shuffle阶段的两个部分：Map相关部分和Reduce相关部分

11.2 MapReduce计算架构

- Shuffle（归并）阶段——Map端的Shuffle

- Mapper从HDFS读取split，然后执行map ()；
- 根据key或value以及Reducer数量划分输出中间键值表，决定交由哪个reduce task来处理。
- 将中间数据写入内存缓冲区。
- map task完成时，将全部溢写文件归并到一起合成一个溢写文件（Merge）

11.2 MapReduce计算架构

- Shuffle（归并）阶段——Reduce端的Shuffle
- Copy领取数据
- Merge归并数据
- 最后的归并文件作为Reducer的输入文件发送给Reducer执行
- 最后输出结果存入HDFS
- 将多个Map任务的输出结果按照不同的分区copy到不同的reduce节点执行reduce task
- 每个reduce task都会创建一个Reducer实例
- 通过执行Reducer的reduce ()方法将来自不同Map的具有相同的key值的键值对进行合并处理
- Reducer会通过调用OutputCollector对象将它所完成的化简结果写入HDFS文件系统，输出文件格式由OutputFormat类来控制

11.3 实际案例展示

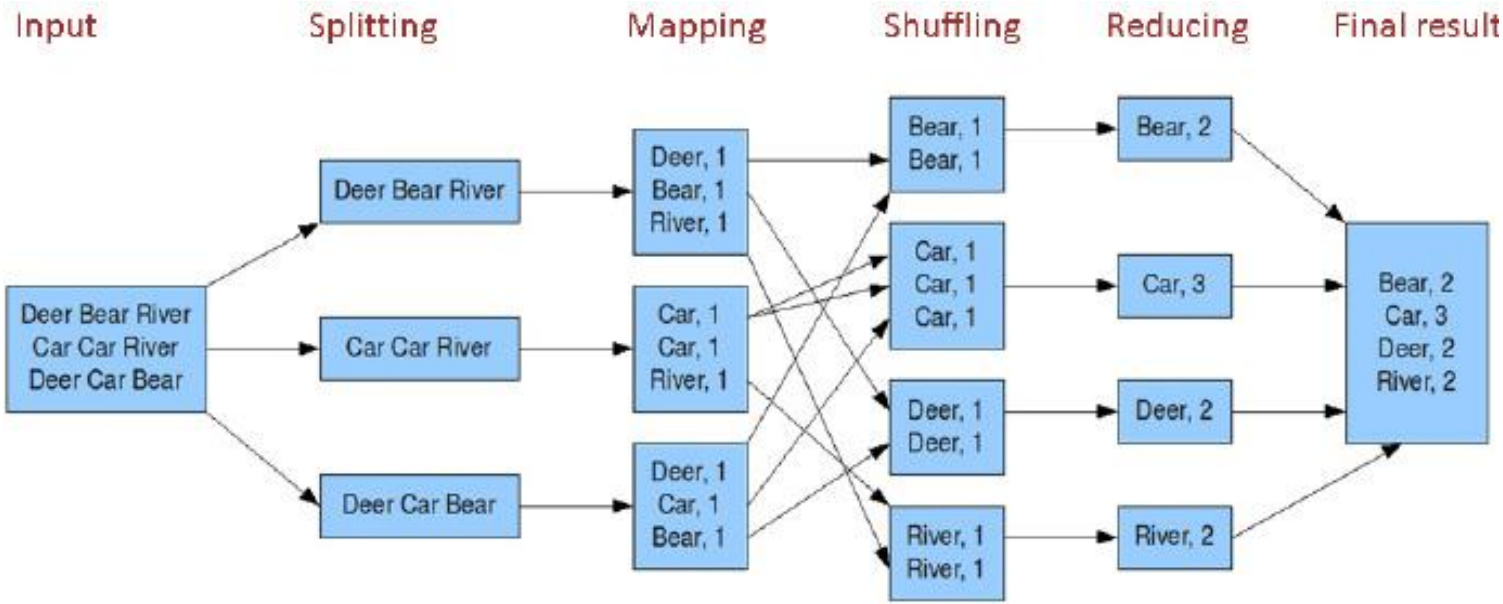
- 案例描述

• 输入文件：一个包含3行文字的文本文件（每个单词间用空格隔开，图（见下页）最左侧Input列所示）；

• 输出结果：该文件的词频统计，每一行输出一个键值对“单词，出现次数”（图（见下页）最右侧Final result列所示）；

• 计算模型：MapReduce

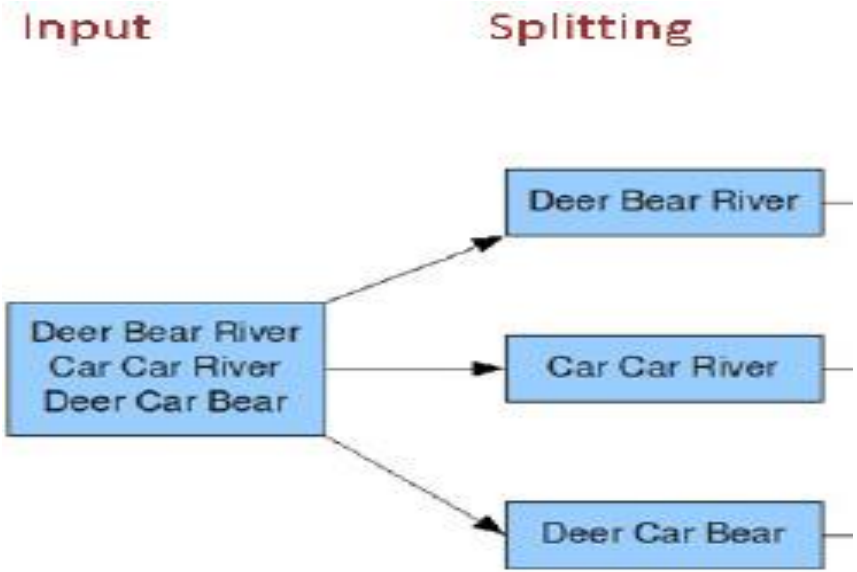
11.3 实际案例展示


Input
Splitting
Mapping
Shuffling
Reducing
Final result
Deer Bear River
Car Car River
Deer Car Bear
Deer Bear River
Car Car River
Deer Car Bear
Deer, 1
Bear, 1
River, 1
Car, 1
Car, 1
River, 1
Deer, 1
Car, 1
Bear, 1
Bear, 1
Bear, 1
Car, 1
Car, 1
Car, 1
Deer, 1
Deer, 1
River, 1
River, 1
Bear, 2
Car, 3
Deer, 2
River, 2
Bear, 2
Car, 3
Deer, 2
River, 2

11.3 实际案例展示

- 第一步：Split

  假设将输入数据文件分为3个split，每个split包含一行文字



- 第二步：Map

首先将split的每一行文字转换成如下的键值对（每行第一个字符的字节偏移量作为Key）：

Key	Value
0	Deer Bear River

split 2

Key	Value
16	Car Car River

split 3

Key	Value
30	Deer Car Bear

11.3 实际案例展示

- 第二步：Map

然后针对每一个split执行map ( )方法，此处为对上述键值对表的每一行进行词频统计，每一个Map任务（针对一个split）都会生成如下的键值对：

split 1

Key	Value
0	Deer Bear River

map 1

map1 输出中间结果

< Deer， 1 >
< Bear， 1>
< River，1>

split 2

Key	Value
16	Car Car River

map 2

map 2 输出中间结果

< Car， 1 >
< Car， 1>
< River，1>

split 3

Key	Value
30	Deer Car Bear

map 3

map 3 输出中间结果

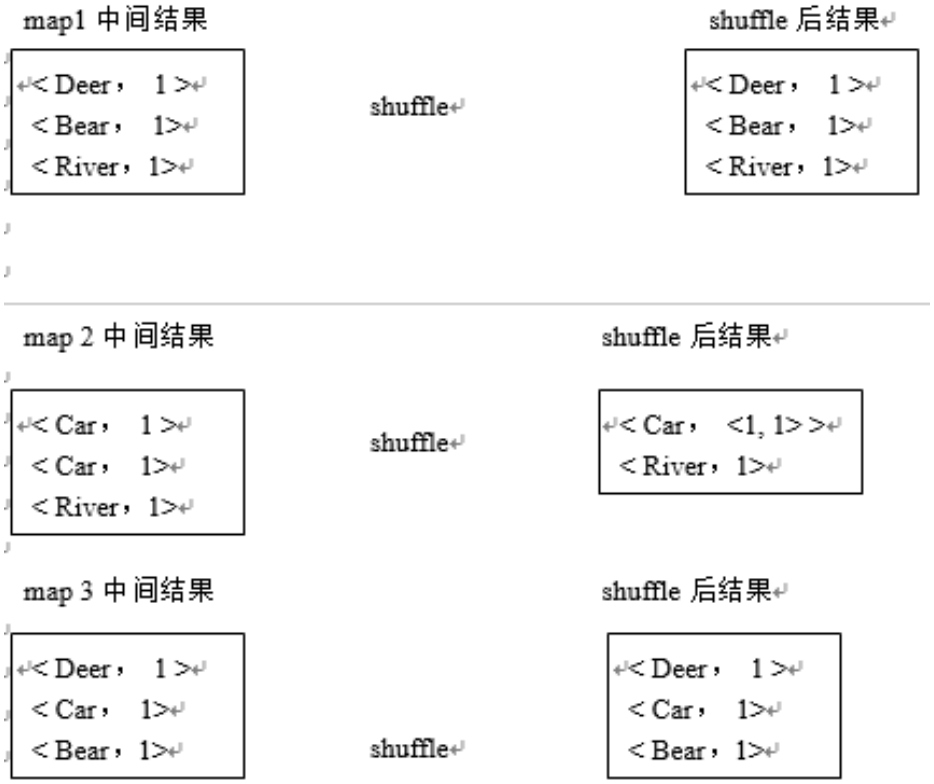
< Deer， 1 >
< Car， 1>
< Bear，1>

11.3 实际案例展示

- 第三步：Shuffle

  Map端shuffle

  没有定义Combiner：

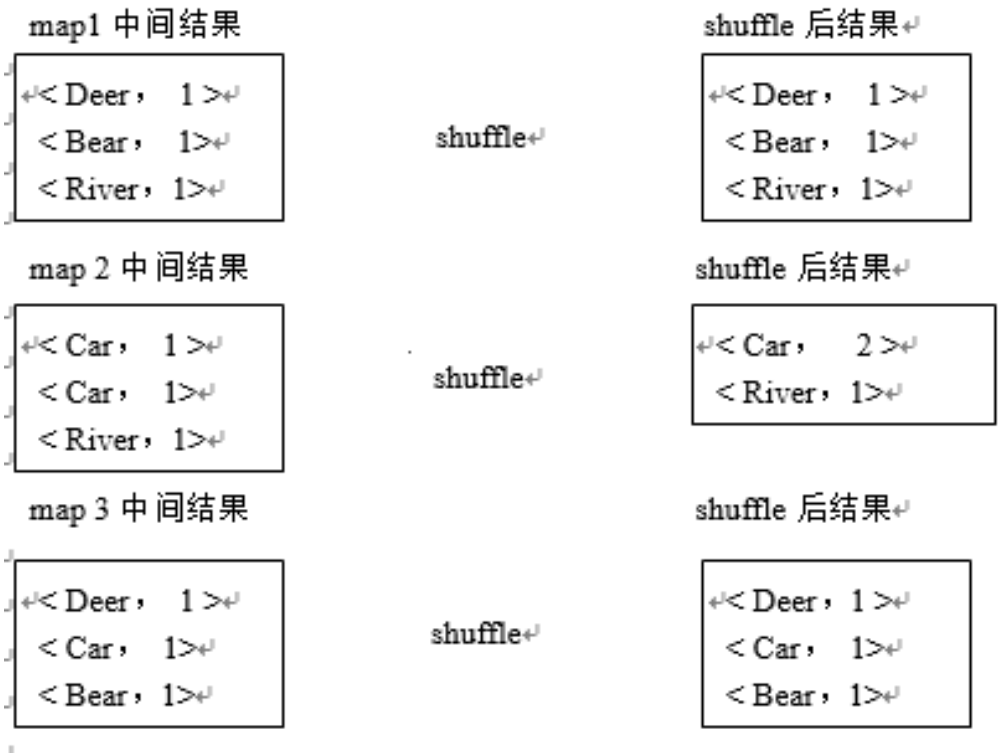


11.3 实际案例展示

- 第三步：Shuffle

  Map端shuffle

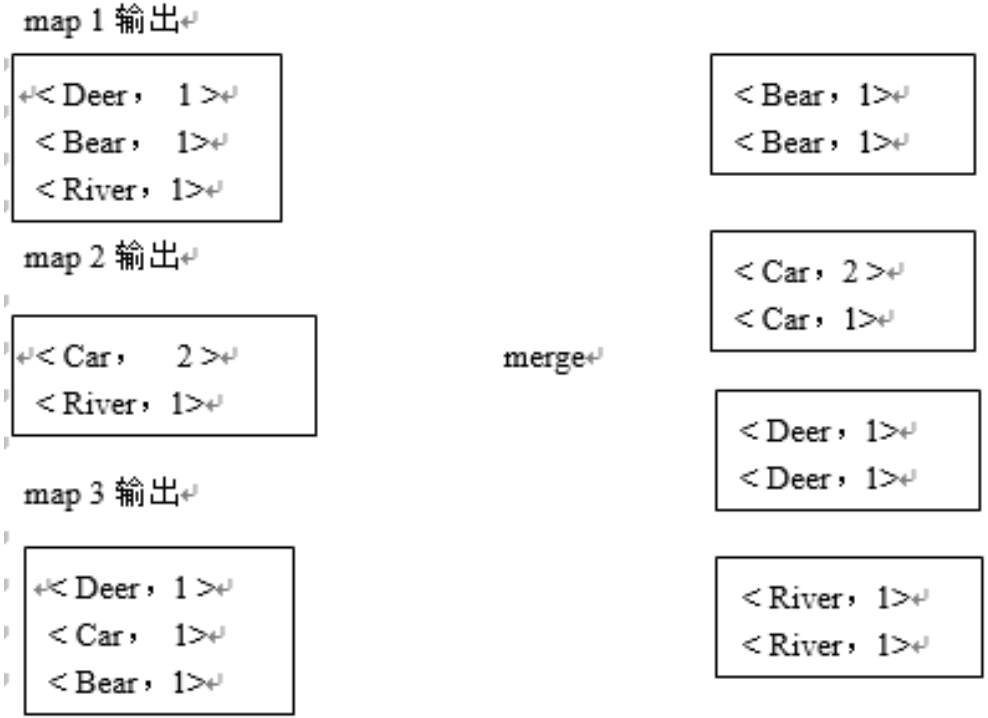
  定义了Combiner：



11.3 实际案例展示

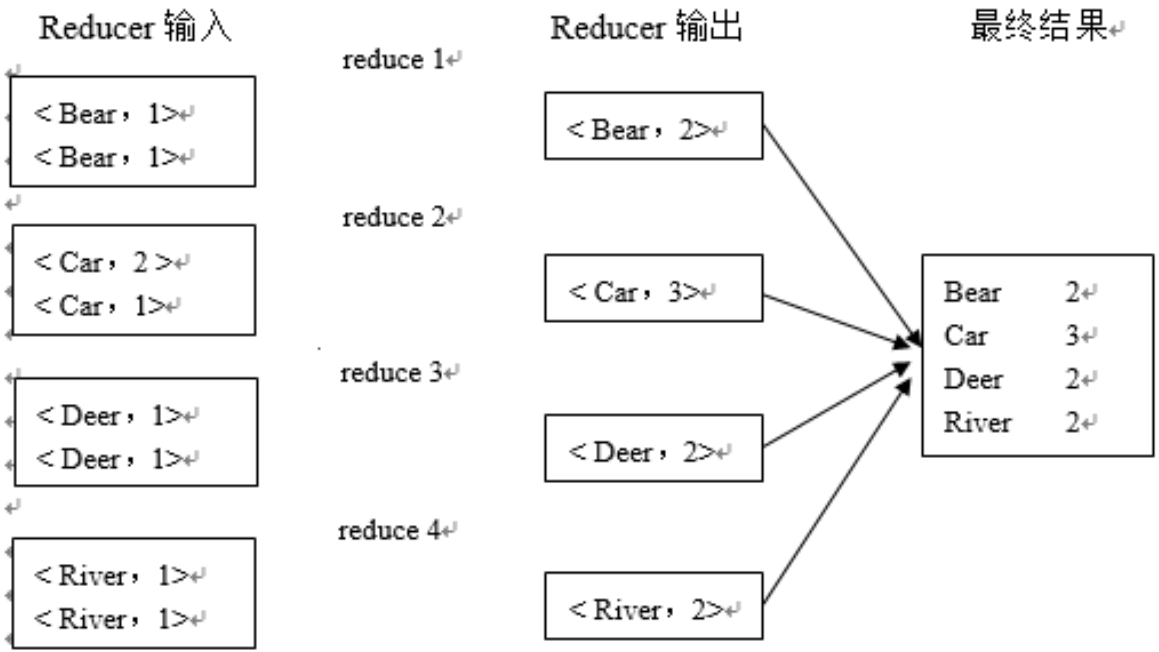
- 第三步：Shuffle

  Reduce端shuffle：



11.3 实际案例展示

- 第四步：Reduce



End of Session
Thank you!