第5章

大数据计算与分析

5.1 Hadoop & MapReduce

Hadoop MapReduce是一个使用简易的软件框架，基于它写出来的应用程序能够运行在由上千个商用机器组成的大型集群上，并以一种可靠容错的方式并行处理上TB级别的数据集。MapReduce是一种编程模型。Hadoop MapReduce采用主从(master/slave)结构。按照其编程规范，只需要编写少量的业务逻辑代码即可实现一个强大的海量数据并发处理程序。核心思想是分而治之。Mapper负责分，把一个复杂的任务分成若干个简单的任务分发到网络上的每个节点并行执行，最后把Map阶段的结果由Reduce进行汇总，输出到HDFS中，大大缩短了数据处理的时间开销。MapReduce就是以这样一种可靠且容错的方式对大规模集群海量数据进行数据处理、数据挖掘、机器学习等方面的操作。

MapReduce是Google最早提出用来进行创建和更新索引的一种分布式计算模式，该模式提供了一个简单的分布式计算框架，来降低分布式计算编程的难度，从而很好地解决了一般商用机及服务器面对海量数据计算响应过慢甚至不能完成作业的问题。海量数据对于单机而言，由于硬件资源限制，肯定是无法胜任的，而一旦将单机版程序扩展到集群来分布式运行，将极大增加程序的复杂度和开发难度。引入MapReduce框架后，开发人员可以将绝大部分工作集中在业务逻辑的开发上，而将分布式计算中的复杂性交由MapReduce框架来处理。

目前MapReduce主要用于海量数据的分布式计算，它主要起源于函数编程思想中的Map函数和Reduce函数操作。Map函数的操作原理是把一组数据映射为一个键值，Reduce函数的工作原理是对Map函数输出结果进行合并处理。

用户只需编写Map(映射)和Reduce(归约)两个函数，即可完成简单的分布式程序的设计。Map函数以键/值(key/value)对作为输入，产生另外一系列键/值对作为中间输出写入本地磁盘。MapReduce框架会自动将这些中间数据按照key值进行聚集，且key值相同(用户可设定聚集策略，默认情况下是对key值进行哈希取模)的数据被统一交给Reduce函数处理。Reduce函数以key及

对应的 value 列表作为输入，经合并 key 相同的 value 值后，产生另外一系列键/值对作为最终输出。

Map 函数能够在一堆混杂的数据中按照开发者的意向提取需要的数据特征，交给 Reduce 函数来归纳输出最终的结果。

一道真实的大数据面试题

你有 1GB 的文本数据，需要写一个 Python 程序统计这 1GB 数据中每个单词出现的次数，但是这个程序每个进程最多只允许占用 256MB 内存，请问你应如何统计？

5.1.1　用 MapReduce 解决一个问题

我们还是通过档案馆管理员的例子来描述 MapReduce。图 5.1 是 MapReduce 的实际执行过程——Shuffle，并且把 Shuffle 的细节进行了形象的描述。

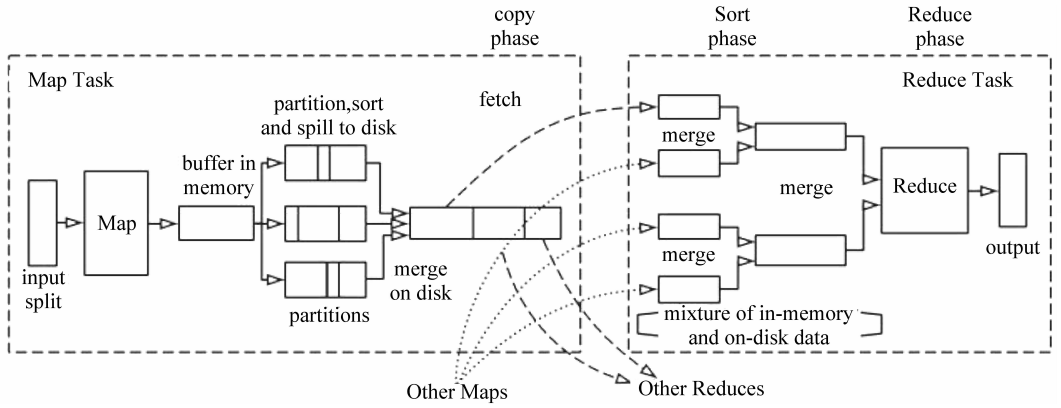

图 5.1　MapReduce 实际执行过程

与 HDFS 不同，HDFS 是解决数据如何存储的问题，对于如何统计没有描述。MapReduce 要解决的是如何统计的问题，即具体是如何计算出“大连理工大学”出现多少次的。需要注意的是，两个例子描述的是两个不同层面的问题，请不要过度关联。待理解了整个过程以后，会对它们有新的认识。

新学期到了，档案馆管理员给甲、乙、丙 3 个同学安排了新的文件任务，这次的文档都是英文，任务是统计每个单词出现了多少次。

一共有 4 个英文文件需要处理。

(1) Split：首先，管理员根据文件大小分配任务。甲处理文件 1、2，乙处理文件 3，丙处理文件 4。

(2) Partition：按照单词首字母分类，甲、乙、丙分别先将所负责文件的单词用铅笔记录在一张纸上，如图 5.2(a)所示。

(3) Sort and Spill：每次这张纸写满后，把纸上记录的内容录入计算机。录入时，顺便将每部分的单词排序录入，如图 5.2(b)所示。然后用橡皮将纸上的内容擦掉，重新记录。

(4) Merge：当甲将自己负责的文件都输入计算机后，通过计算机再次将有相同首字母的部分合并到一起，如图 5.3 所示。乙、丙也做相同的操作。

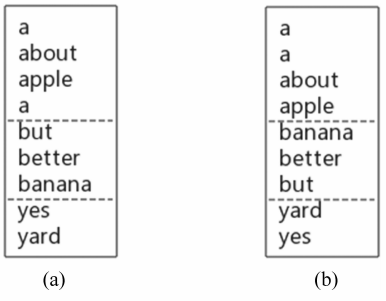

图 5.2 第一阶段文件 Split，Partition，Sort 和 Spill 操作

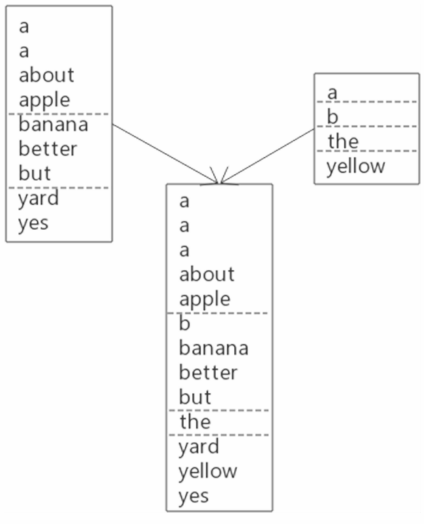

图 5.3 第一阶段 Merge 操作

甲、乙、丙都完成了第一阶段工作以后，管理员开始安排第二阶段任务。

(1) Fetch：管理员让甲将第一阶段统计的 a 开头的单词都收集起来，如图 5.4 所示。由于甲、乙、丙各得出一个结果，所以收集了 3 组 a 开头单词的结果。

(2) Merge：首先甲将第一阶段甲、乙的结果进行合并，然后再与第一阶段丙的结果合并，得到最终结果，如图 5.5 所示。

图 5.4 第二阶段 Fetch 操作

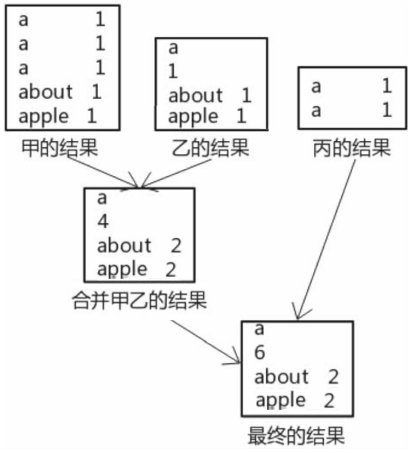

图 5.5 第二阶段 Merge 操作

(3) 与此同时，乙、丙在合并其他首字母的结果。甲合并完首字母为 a 的结果后，也去合并剩余首字母的结果。当 a～z 首字母都合并完，任务结束。

5.1.2　MapReduce 模型

MapReduce 的核心思想是分而治之，把大的任务分成若干个小任务，并行执行小任务，最后把所有的结果汇总，因此整个作业的过程被分成两个阶段：Map 阶段和 Reduce 阶段。Map 阶段主要负责分，即把复杂的任务分解为若干个简单的任务来处理。这里简单的任务不但指数据或计算的规模相对原任务要大大缩小，同时这些小任务彼此间几乎没有依赖关系，可以并行计算。最后要注意就近计算原则，即任务应该分配到存放着所需数据的节点上进行计算。Reduce 阶段负责对 Map 阶段的结果进行汇总。

例如，输入信息“Whether the weather be fine or whether the weather be not. Whether the weather be cold or whether the weather be hot. We will weather the weather whether we like it or not. ”在此信息上运行 Map 和 Reduce 操作输出的结果如图 5.6 所示。

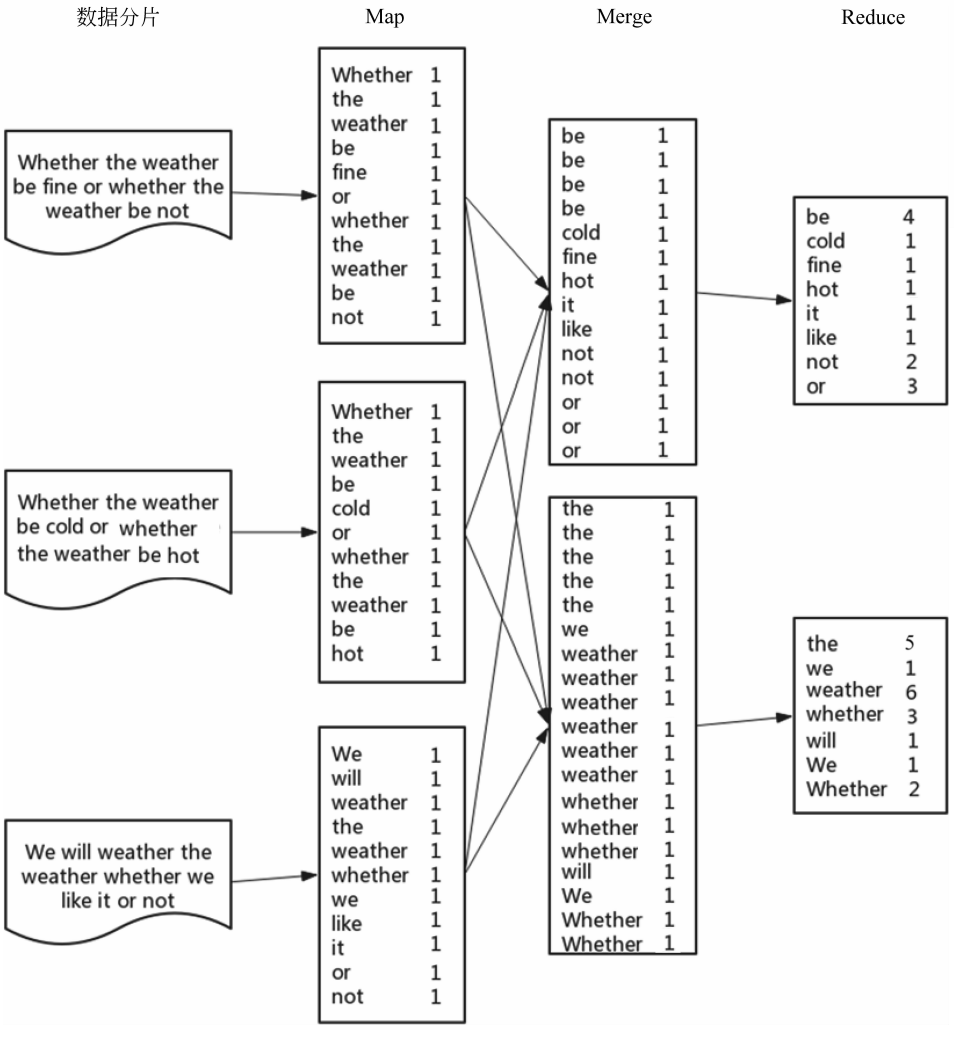

图 5.6　Map 和 Reduce 操作的数据处理过程实例

5.1.3 Hadoop 中的 MapReduce

MapReduce 作为一个分布式运算程序的编程框架，是 Hadoop 的四大组件之一，是用户开发基于 Hadoop 的数据分析应用的核心框架，其核心功能是将用户编写的业务逻辑代码和自带默认组件整合成一个完整的分布式运算程序，并发运行在一个 Hadoop 集群上。

图 5.7 为 Hadoop 中 Map 和 Reduce 操作的数据处理和传输过程。

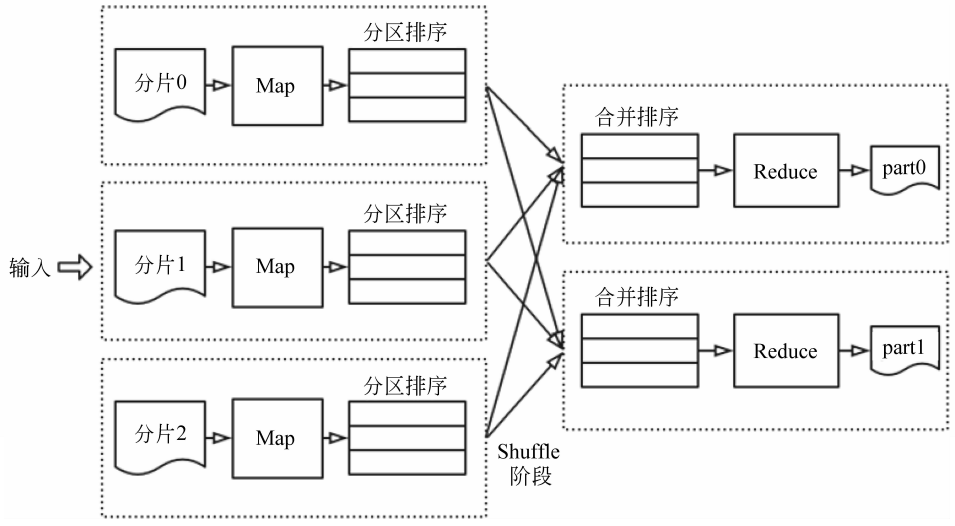

图 5.7 Hadoop 中 Map 和 Reduce 操作的数据处理和传输过程

(1) 首先，输入数据被切割成数据分片，每一个分片会复制多份到 HDFS 中。

(2) 在存储有输入数据分片的节点上运行 Map 任务。

(3) Map 任务的输出结果在本地进行分区、排序。分区方法时通常将 key 值相同的数据放在同一个分区，Reduce 阶段同一个分区的数据会被安排到同一个 Reduce 中。

(4) Shuffle 过程把 key 值相同的 value 合并成列表(list)作为 Reduce 的输入。

(5) 每一个 Reduce 将所有 Map 对应分区的数据复制过来，进行合并和排序，将相同的 key 值对应的数据统一处理。在 Reduce 计算阶段，Reduce 的输入键是 key，而输入值是相同的 key 数据对应的 value 所构成的一个迭代器数据结构。

(6) Reduce 处理后的输出结果可以存储到 HDFS 中。这些输出结果可以进一步作为另一个 MapReduce 任务的输入，进行下阶段的任务计算。

在 Hadoop 框架中，MapReduce 以组件的模式工作，主要包括 JobTracker 和 TaskTrackers 两个组成部分。JobTracker 是一个 master 服务，负责接收及分配作业，并调度作业对应的子任务运行在 TaskTracker 上，MapReduce 组件框架图如图 5.8 所示。

如图 5.8 所示，MapReduce 的工作流程如下所述。

(1) 用户提交一个作业，该作业被发送到 JobTracker 服务器上，JobTracker 是 MapReduce 的核心，它通过心跳机制管理所有的作业。

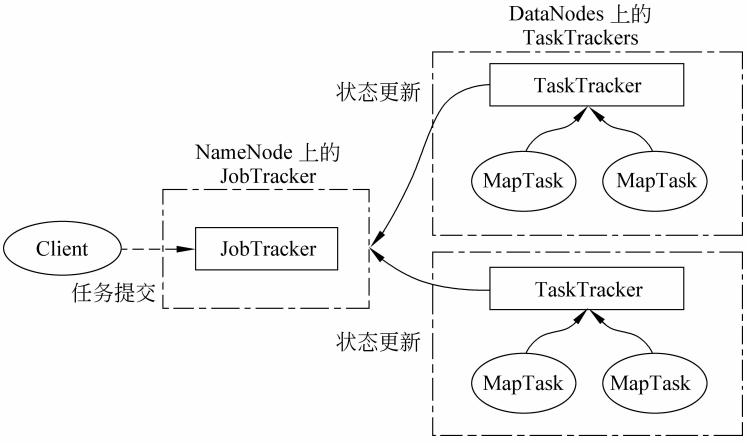

图 5.8　MapReduce 组件框架图

(2) TaskTracker 为 MapReduce 集群中的一个工作单元，主要完成 JobTracker 分配的任务。

(3) TaskTracker 监控主机任务运行情况，通过心跳机制向 JobTracker 反馈自己的工作状态。

此过程中，使用者只需要在 MapReduce 模型上进行开发，并将数据以键/值形式表示，而不用考虑集群中的计算机之间的任务调度、容错处理、各节点之间的通信等细节问题。MapReduce 编程模型将借助 Hadoop 分布式文件系统，自动将数据计算分布到集群上调度作业运行。其中，涉及的类或进程功能如下所述。

(1) JobTracker：一般应该部署在单独机器上的 master 服务，功能是接收 Job，负责调度 Job 的每一个子任务 Task 运行在 TaskTracker 上，并且对它们进行监控，如果发现有失败的 Task 就重启。

(2) TaskTracker：运行于多节点的 slaver 服务，功能是主动通过心跳与 JobTracker 进行通信接收作业，并且负责执行每一个任务。

(3) MapTask 和 ReduceTask：Mapper 根据 JobJar 中定义的输入数据＜key1，value1＞读入，生成临时的＜key2，value2＞，如果定义了 Combiner，MapTask 会在 Mapper 完成后调用该 Combiner 将相同 key 的值做合并处理，目的是为了减少输出结果。MapTask 全部完成后交给 ReduceTask 进程调用 Reducer 函数处理，生成最终结果＜key3，value3＞。

5.1.4　Hadoop Streaming

Hadoop Streaming 是 Hadoop 的一个工具，它帮助用户创建和运行一类特殊的 Map/Reduce 作业，这些特殊的 Map/Reduce 作业是由一些可执行文件或脚本文件充当 Mapper 或者 Reducer。Mapper 和 Reducer 都是可执行文件，它们从标准输入读入数据(一行一行读)，并把计算结果发给标准输出。Streaming 工具会创建一个 Map/Reduce 作业，并把它发送给合适的集群，同时监视这个作业的整个执行过程。

如果一个可执行文件被用于 Mapper，则在 Mapper 初始化时，每一个 Mapper 任务会把这个可执行文件作为一个单独的进程启动。Mapper 任务运行时，它把输入切分成行并把每一行提供给可执行文件进程的标准输入。同时，Mapper 收集可执行文件进程标准输出的内容，并把收到的每一行内容转化成键/值对，作为 Mapper 的输出。默认情况下，一行中第一个 tab 之前的部分作为 key，之后的(不包括 tab)作为 value。如果没有 tab，整行作为 key，value 为 null。不过，这可以定制。

如果一个可执行文件被用于 Reducer，每个 Reducer 任务会把这个可执行文件作为一个单独的进程启动。Reducer 任务运行时，它把输入切分成行并把每一行提供给可执行文件进程的标准输入。同时，Reducer 收集可执行文件进程标准输出的内容，并把每一行内容转化成键/值对，作为 Reducer 的输出。默认情况下，一行中第一个 tab 之前的部分作为 key，之后的(不包括 tab)作为 value。下面讨论如何自定义 key 和 value 的切分方式。

Hadoop Streaming 除了支持流命令选项(streaming command options)外，还支持 Hadoop 的通用命令选项(generic command options)。命令的使用规则如下：

```
>mapred streaming [genericOptions] [streamingOptions]
```

需要注意的是，在提交 Streaming 作业中使用到通用命令选项的时候，需要把通用命令选项设置在流命令选项之前，否则将会出现一些错误。

目前的 Hadoop streaming(Hadoop 3.0.0)支持的流命令选项如表 5.1 所示。

表 5.1 Hadoop Streaming 支持的流命令选项

参　数	是否可选	描　述
-input directoryname or filename	required	Mapper 的输入路径
-output directoryname	required	Reducer 输出路径
-mapper executable or JavaClassName	optional	Mapper 可执行程序或 Java 类名
-reducer executable or JavaClassName	optional	Reducer 可执行程序或 Java 类名
-file filename	optional	Mapper、Reducer 或 Combiner 依赖的文件
-inputformat JavaClassName	optional	键/值对输入格式，默认为 TextInputFormat
-outputformat JavaClassName	optional	键/值对输出格式，默认为 TextOutputformat
-partitioner JavaClassName	optional	该类确定将键发送到哪个 Reduce
-combiner streamingCommand or JavaClassName	optional	Map 输出结果执行 Combiner 的命令或者类名
-cmdenv name=value	optional	环境变量
-inputreader	optional	向后兼容，定义输入的 Reader 类，用于取代输出格式
-verbose	optional	输出日志
-lazyOutput	optional	延时输出

续表

参　数	是否可选	描　述
-numReduceTasks	optional	定义 Reduce 数量
-mapdebug	optional	Map 任务运行失败时，执行的脚本
-reducedebug	optional	Reduce 任务运行失败时，执行的脚本

1. 提交作业的时候打包文件

如上所述，可以指定任意的可执行文件作为 Mapper 或者 Reducer。在提交 Hadoop Streaming 作业的时候，Mapper 或者 Reducer 程序不需要事先部署在 Hadoop 集群的任意一台机器上，仅需要在提交 Streaming 作业的时候指定 -file 参数，这样 Hadoop 会自动将这些文件分发到集群。使用如下：

```
1. mapred streaming \
2.   -input myInputDirs \
3.   -output myOutputDir \
4.   -mapper myPythonScript.py \
5.   -reducer /usr/bin/wc \
6.   -file myPythonScript.py
```

上面命令中-file myPythonScript.py 会导致 Hadoop 将这个文件自动分发到集群。

除了可以指定可执行文件之外，还可以打包 Mapper 或者 Reducer 程序会用到的文件(包括目录、配置文件等)，例如：

```
1. mapred streaming \
2.   -input myInputDirs \
3.   -output myOutputDir \
4.   -mapper myPythonScript.py \
5.   -reducer /usr/bin/wc \
6.   -file myPythonScript.py \
7.   -file myDictionary.txt
```

2. 为作业指定其他插件

与正常的 Map / Reduce 作业一样，还可以为流式作业指定其他插件，选项如下：

- -inputformat JavaClassName
- -outputformat JavaClassName
- -partitioner JavaClassName
- -combiner streamingCommand or JavaClassName

为-inputformat 指定的 class 文件必须返回 Text 类型的键/值对。如果没有指定 InputFormat 类，默认使用 TextInputFormat 类。TextInputFormat 中 key 的返回类型是

LongWritable，这个并不是输入数据的一部分，所以 key 部分将会被忽略，而仅返回 value 部分。

为-outputformat 指定的 class 文件接收的数据类型是 Text 类型的键/值对。如果没有指定 OutputFormat 类，默认使用 TextOutputFormat 类。

可以在提交 Streaming 作业的时候设置环境变量，如下：

```
-cmdenv EXAMPLE_DIR=/home/example/dictionaries/
```

在提交流作业的时候，可支持的通用命令选项如表 5.2 所示。

表 5.2 通用命令选项

参 数	是否可选	描 述
-conf configuration_file	optional	定义应用的配置文件
-D property=value	optional	定义参数
-fs host：port or local	optional	定义 NameNode 地址
-files	optional	定义需要复制到 Map/Reduce 集群的文件，多个文件以逗号分隔
-libjars	optional	定义需要引入 classpath 的 jar 文件，多个文件以逗号分隔
-archives	optional	定义需要解压到计算节点的压缩文件，多个文件以逗号分隔

3. 通过-D 选项指定配置变量

可以通过-D ＜property＞＝＜value＞的方式指定额外的配置变量（configuration variables）。为了改变默认的本地临时目录，可以使用下面的命令：

```
-D dfs.data.dir=/tmp
```

增加额外的本地临时目录可以使用下面的命令：

```
-D mapred.local.dir=/tmp/local
-D mapred.system.dir=/tmp/system
-D mapred.temp.dir=/tmp/temp
```

4. 设置只有 Map 的作业

有时候仅想跑只有 Map 的 Hadoop 作业，只需要将 mapreduce.job.reduces 设置为 0 即可实现。这会导致 MapReduce 框架不会启动 Reduce 类型的 Task。MapTask 的输出就是作业的最终结果输出，设置如下：

```
-D mapreduce.job.reduces=0
```

为了向后兼容，Hadoop Streaming 还支持-reducer NONE 选项，其含义等同于-D mapreduce.job.reduces＝0。

5. 设置 Reduce 的个数

下面例子中将程序的 reduce 个数设置为 2：

```
1. mapred streaming \
2.   -D mapreduce.job.reduces=2 \
3.   -input myInputDirs \
4.   -output myOutputDir \
5.   -mapper /bin/cat \
6.   -reducer /usr/bin/wc
```

6. 自定义行数据如何拆分成键/值对

本文开头介绍过，当 MapReduce 框架从 stdout 读取行数据的时候，它会把一行数据拆分成一个键/值对。默认情况下，tab 制表符分隔的前一部分数据作为 key，后一部分数据作为 value。当然，可以自定义行数据的分隔符：

```
1. mapred streaming \
2.   -D stream.map.output.field.separator=. \
3.   -D stream.num.map.output.key.fields=4 \
4.   -input myInputDirs \
5.   -output myOutputDir \
6.   -mapper /bin/cat \
7.   -reducer /bin/cat
```

在上面例子中，stream. map. output. field. separator 指定“.”为 key 和 value 的分隔符。

7. 使用大文件或归档文件

可以使用-files 和 -archives 选项分别指定文件或者归档文件(archives)，这些文件可以被 Tasks 使用。使用这个选项时，需要把这些文件或者归档文件上传到 HDFS。这些文件在作业执行的时候会被缓存到所有的 Jobs 中。

-files 选项会在当前 Tasks 的工作目录(current working directory)下创建一个符号链接(symlink)，这个链接指定的就是从 HDFS 复制文件的副本。下面例子中，指定了 HDFS 上的 testfile. txt 文件，在使用-files 选项之后，其会在 Tasks 的当前工作目录下创建名为 testfile. txt 的符号链接。

```
-files hdfs://host:fs_port/user/testfile.txt
```

当然，也可以自己通过 # 设置符号链接的名字：

```
-files hdfs://host:fs_port/user/testfile.txt#testfile
```

如果需要指定多个文件，使用如下：