实验五　MapReduce大数据编程模型实现单词计数

（一）实验目的

1. 理解MapReduce大数据编程思想；
2. 会编写MapReduce版本的WordCount；
3. 会执行该程序；
4. 自行分析执行过程。

（二）实验要求

1. 基于MapReduce模型编写WordCount程序；
2. 在大数据分析平台上分布式执行WordCount；
3. 将执行结果与Hadoop框架自带的WordCount案例进行比较。

（三）实验环境

1. 大数据分析实验系统（FSDP）；
2. CentOS 6.7；
3. Hadoop 2.7.1；
4. Java SE 10，Eclipse 4.7。

（四）实验步骤

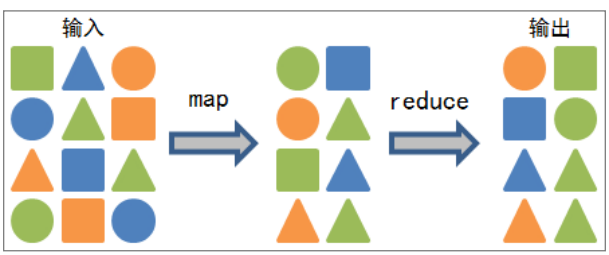
1. MapReduce预备知识；
2. 编写并在大数据分析平台上执行WordCount；
3. 执行结果分析与比较。

1、MapReduce预备知识

MapReduce是一种计算模型，简单地说就是将大批量的工作（数据）分解（MAP）执行，然后再将结果合并成最终结果（REDUCE）。这样做的好处是可以在任务被分解后，可以通过大量廉价机器进行并行计算，减少整个操作的时间。

适用范围：数据量大，数据种类少可以放入内存。

基本原理：将数据交给不同的机器去处理，数据划分，结果归约。



MapReduce模型的核心是Map函数和Reduce函数，二者都是以<key,value>作为输入，按一定的映射规则转换成另一个或一批<key,value>进行输入，如下表：

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$<k_1, v_1>$会输出一批计算的中间结果$<k_2, v_2>$。
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。

编写MapReduce程序的关键点在于掌握分布式的编程思想和方法，计算过程通常包括以下五个步骤：

(1) 迭代，遍历输入数据，并将之解析成key/value对；

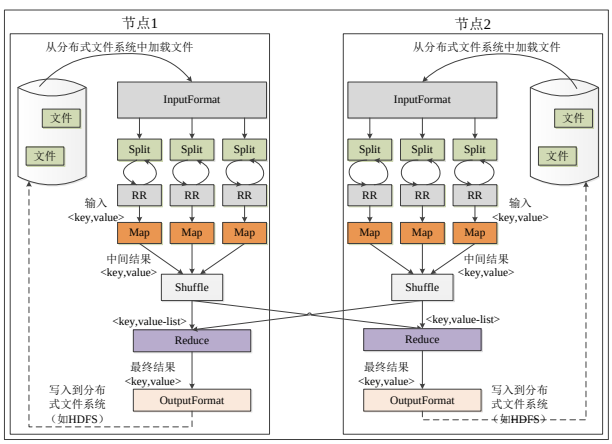
(2) 将输入key/value对映射（map）成另外一些key/value对；

(3) 依据key对中间数据进行分组（grouping）；

(4) 以组为单位对数据进行归约（reduce）；

(5) 迭代，将最终产生的key/value对保存到输出文件。

MapReduce工作流程中的各个执行阶段



2、编写并在大数据分析平台上执行WordCount

使用Java语言编写程序，主要编写Map和Reduce类，其中Map过程需要继承org.apache.hadoop.mapreduce包中的Mapper类，并重写其map方法；Reduce过程需要继承org.apache.hadoop.mapreduce包中的Reduce类，并重写其reduce方法，参考代码见WordCount.java。

实验五　MapReduce大数据编程模型实现单词计数

(1)将WordCount.java文件复制到Linux下的一个目录，这里复制到/root/example/hadoop_example。

```
[root@fsmanager hadoop_example]# ls
WordCount.java
```

(2)编译WordCount.java文件，其中-classpath选项表示要引用hadoop官方的包，-d选项表示要将编译后的class文件生成的目标目录。

```
[root@fsmanagerhadoop_example]#javac -
classpath .:/usr/hdp/2.3.2.0-
2950/hadoop/hadoop-annotations-2.7.1.2.3.2.0-
2950.jar:/usr/hdp/2.3.2.0-2950/hadoop/hadoop-
common-2.7.1.2.3.2.0-2950.jar:/usr/hdp/2.3.2.0-
2950/hadoop/hadoop/share/hadoop/mapreduce/hadoo
p-mapreduce-client-core-2.7.1.2.3.2.0-
2950.jar:/usr/hdp/2.3.2.0-
2950/hadoop/client/commons-cli.jar -d  .
WordCount.java
```

(3)将编译后的class文件打包。

```
[root@fsmanager hadoop_example]# jar -cvf
wordcount.jar -c wordcount
added manifest
adding: -c(in = 3474) (out= 3001)(deflated 13%)
adding: wordcount/(in = 0) (out= 0)(stored 0%)
adding: wordcount/WordCount.class(in = 1947)
(out= 1045)(deflated 46%)
adding:
wordcount/WordCount$IntSumReducer.class(in =
1759) (out= 746)(deflated 57%)
adding:
wordcount/WordCount$TokenizerMapper.class(in =
1756) (out= 760)(deflated 56%)
[root@fsmanager hadoop_example]# ls
-c  wordcount  wordcount.jar  WordCount.java
```

(4)在本地用echo生成2个文件f1.txt和f2.txt，用于输入数据：

```
[root@fsmanager hadoop_example]# echo "hello
world, hello hadoop" > f1.txt
[root@fsmanager hadoop_example]# echo "weclcome
to hadoop world" > f2.txt
[root@fsmanager hadoop_example]# cat  f1.txt
f2.txt
hello world, hello hadoop
weclcome to hadoop world
```

(5)在hadoop上建立一个输入文件目录。

```
[root@fsmanager hadoop_example]# hadoop fs -
mkdir hadoop_example
[root@fsmanager hadoop_example]# hadoop fs -ls
Found 1 items
drwxr-xr-x   - root hdfs 0 2016-12-12 16:33
hadoop_example
```

(6)将文件f1.txt、f2.txt上传到hadoop上的hadoop_example目录。

```
[root@fsmanager hadoop_example]# hadoop fs -put
f1.txt f2.txt hadoop_example
[root@fsmanager hadoop_example]# hadoop fs -ls
hadoop_example
Found 2 items
-rw-r--r--   2 root hdfs 26 2016-12-12 16:36
hadoop_example/f1.txt
-rw-r--r--   2 root hdfs 25 2016-12-12 16:36
hadoop_example/f2.txt
```

(7)以建立的Input目录作为输入参数，运行jar。

```
[root@fsmanager hadoop_example]#hadoop jar
wordcount.jar wordcount.WordCount
hadoop_example output
```

3、执行结果分析比较

(1)查看output目录是否有结果。

```
[root@fsmanager hadoop_example]# hadoop fs -ls
output
-rw-r--r--   2 root hdfs 0 2016-12-14 09:12
output/_SUCCESS
-rw-r--r--   2 root hdfs 25 2016-12-14 09:12
output/part-r-00000
```

(2)将该目录下所有文本文件合并后下载到本地。

```
(Hadoop fs -text output/part-r-00000)
[root@fsmanager hadoop_example]# hadoop fs -
getmerge output wordcount_result
[root@fsmanager hadoop_example]# ls
-c  f1.txt  f2.txt  wordcount  wordcount.jar
WordCount.java  wordcount_result
```

(3)查看下载下来的计算结果。

```
[root@fsmanager hadoop_example]# more
wordcount_result
hadoop  1
hello  2
world  1
```

执行hadoop框架自带的wordcount案例，并与的执行结果进行比较。

```
Hadoop jar  /usr/hdp/2.3.2.0-2950/hadoop/
hadoop/share/hadoop/mapreduce/hadoop-mapreduce-
examples-2.7.1.2.3.2.0-2950.jar wordcount
Hadoop_input Hadoop_output
```