实验 4　Hadoop 安装问题解答、机制讲解、InvertedInverse 实验

本周需要完成 hadoop 集群上的 IndexInverse 实验。

一、Hadoop 安装问题

1）ssh 无密码登录问题

密钥成功传输却没有找到/密钥加进 authorized_keys 但是没办法无密码登陆。

解决办法：看一下是否是三台节点的登陆用户名不一致，如果是的话需要修改为一致或按照 http://blog.csdn.net/tragedyxd/article/details/46284949 修改远程连接配置。

2）无法启动多台节点：

a）clusterID 问题：因为多次格式化节点导致 namenode 的 clusterID 与 datanode 的 clusterID 不一致。

解决方法：修改/usr/local/hadoop/hdfs/data/current/VERSION 中的 clusterID，使其与 namenode 的 clusterID 一致。

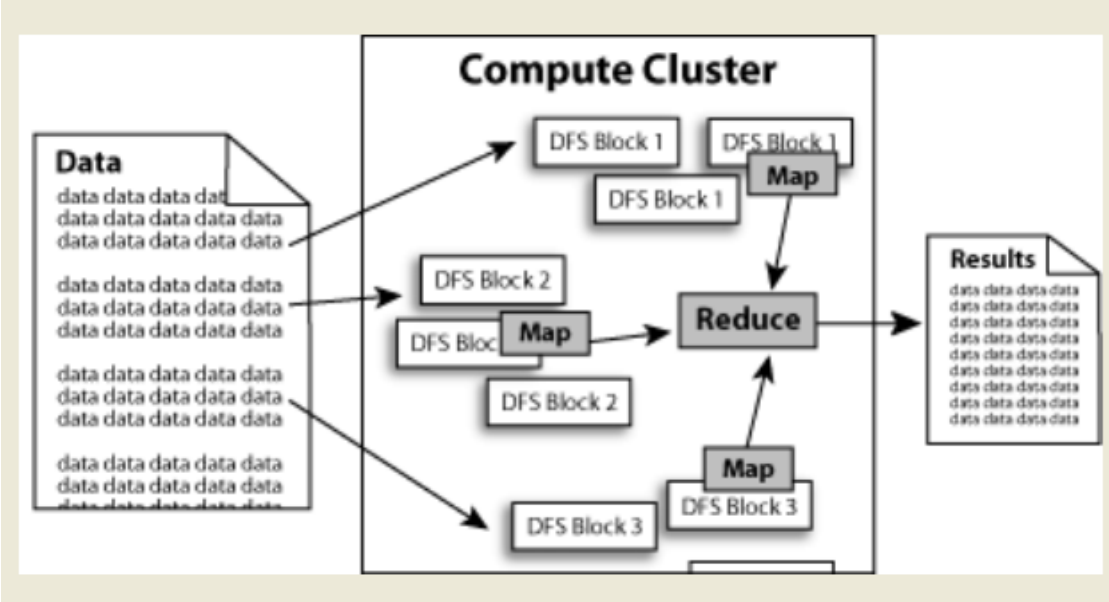
b）connection refused：去掉 127.0.1.1

c）permission denied：修改文件夹的属主（chown 操作）

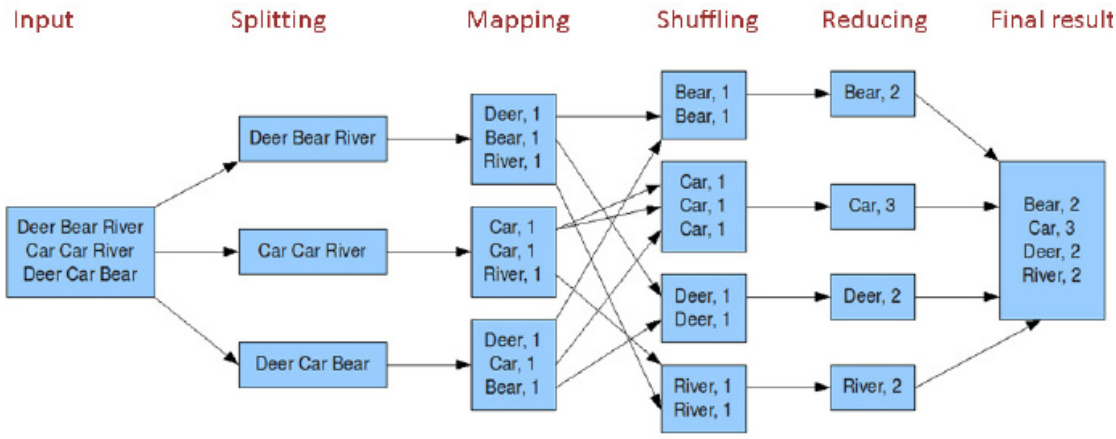
d）运行 wordcount 时显示文件夹已存在：删除 output 和 tmp 即可

二、Hadoop 详解

Hadoop 框架中最核心设计就是：HDFS 和 MapReduce。HDFS 提供了海量数据的存储，MapReduce 提供了对数据的计算。整个 Hadoop 处理的过程如下：



MapReduce:Hadoop 为每一个 input split 创建一个 task 调用 Map 计算，在此 task 中依次处理此 split 中的一个个记录(record)，map 会将结果以 key--value 的形式输出，hadoop 负责按 key 值将 map 的输出整理后作为 Reduce 的输入，Reduce Task 的输出为整个 job 的输出，保存在 HDFS 上. 实例见下图。



wordcount 源码如下

a） map 过程

```
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);
    }
  }
}
```

context is for return

b） reduce 过程

reduce 表累加

```
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, InterruptedException {
    int sum = 0;
    for (IntWritable val : values) {
      sum += val.get();
    }
    result.set(sum);
    context.write(key, result);
  }
}
```

c）执行过程

```
public static void main(String[] args) throws Exception {
  Configuration conf = new Configuration();
  String[] otherArgs = new GenericOptionsParser(conf, args).getRemain
  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);
}
```

三、本周实验 InvertedIndex

1）什么是 InvertedIndex：根据单次返回单次出现的文件及出现的次数。

比如 a.txt

```
Hello world
bye hadoop
```

b.txt

```
bye world
Hello hadoop
```

那么最终的输出应为

```
bye:<a.txt, 1>, <b.txt, 1>
hadoop: <a.txt, 1>, <b.txt, 1>
```

```
Hello:<a.txt,1>,<b.txt,1>
world: <a.txt,1>,<b.txt,1>
```

2）实验提示：

a)编写 map 函数,在 map 函数里面统计每一个单次的出现次数的同时要记录单次的文件名。

提示：用 FileSplit 获取文件所暑的切片信息及文件名

```
FileSplit split=(FileSplit)context.getInputSplit();
split.getPath().toString());
```

b）编写 reduce 函数，对统计的 values 进行累加

3）实验执行

本次实验使用的是 Java 代码（也可选择其他语言），编写好 IndexInverse.java 后，需要将其编程成 class 文件后打包成 jar 包

a） 环境修改:因为编译的代码需要 hadoop 的 jar 包,较为方便的做法就是把 hadoop 的 jar 包加入的环境变量中：

```
export HADOOP_HOME=/usr/local/hadoop
export CLASSPATH=$($HADOOP_HOME/bin/Hadoop classpath):$CLASSPATH
```

classpath 可以直接追加在原有的 classpath 之后，但要注意用:（冒号）隔开

```
export PATH=/home/hadoop/bin:/home/hadoop/.local/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/b
in:/usr/games:/usr/local/games:/snap/bin

export HADOOP_HOME=/usr/local/hadoop
export JAVA_HOME=/usr/local/jvm/jdk1.8.0_60
export JRE_HOME=${JAVA_HOME}/jre
export CLASSPATH=.:${JAVA_HOME}/lib:${JRE_HOME}/lib:$($HADOOP_HOME/bin/hadoop classpath):$CLASSPATH
export PATH=$PATH:${JAVA_HOME}/bin:$HADOOP_HOME/bin:$HADOOP_HOME/sbin:$PATH
```

b） 执行 source ~/setenv.sh，使修改内容生效。

c） 编译 InvertedIndex 代码,使用过时 API 的警告可忽略

```
javac InvertedIndex.java
```

```
hadoop@master:/InvertedIndex$ javac InvertedIndex.java
Note: InvertedIndex.java uses or overrides a deprecated API.
Note: Recompile with -Xlint:deprecation for details.
```

查看编译后的内容，新增了四个 class 文件

```
hadoop@master:/InvertedIndex$ ls
InvertedIndex.class                       InvertedIndex$InvertedIndexMapper.class  InvertedIndex.jar
InvertedIndex$InvertedIndexCombiner.class  InvertedIndex$InvertedIndexReduce.class  InvertedIndex.java
```

d） 用编译后的 class 文件生成 jar 包

```
jar -cvf InvertedIndex.jar InvertIndex*.class
```

```
hadoop@master:/InvertedIndex$ jar -cvf InvertedIndex.jar InvertedIndex*.class
added manifest
adding: InvertedIndex.class(in = 1932) (out= 994)(deflated 48%)
adding: InvertedIndex$InvertedIndexCombiner.class(in = 2137) (out= 929)(deflated 56%)
adding: InvertedIndex$InvertedIndexMapper.class(in = 2085) (out= 909)(deflated 56%)
adding: InvertedIndex$InvertedIndexReduce.class(in = 1893) (out= 812)(deflated 57%)
```

e） 运行 InvertedIndex 程序：

代码与上周一致，注意把 example.jar 换成自己路径下的 InvertedIndex.jar 包，把 WordCount 函数换为 InvertedIndex 函数即可。

```
/usr/local/hadoop/bin/hadoop jar InvertedIndex.jar InvertedIndex /input /output
/usr/local/hadoop/bin/hdfs dfs -cat /ouput/*
```

```
environmental   hdfs://master:9000/inputII/b.txt:1;
figuring        hdfs://master:9000/inputII/a.txt:1;
find    hdfs://master:9000/inputII/a.txt:1;
for     hdfs://master:9000/inputII/b.txt:2;
getting hdfs://master:9000/inputII/b.txt:1;
hard    hdfs://master:9000/inputII/a.txt:1;
has     hdfs://master:9000/inputII/a.txt:1;
in      hdfs://master:9000/inputII/b.txt:1;
it      hdfs://master:9000/inputII/a.txt:1;
jps     hdfs://master:9000/inputII/b.txt:1;
know    hdfs://master:9000/inputII/a.txt:1;
make    hdfs://master:9000/inputII/a.txt:1;
multiple        hdfs://master:9000/inputII/a.txt:1;
new     hdfs://master:9000/inputII/a.txt:1;
other   hdfs://master:9000/inputII/b.txt:1;
out     hdfs://master:9000/inputII/a.txt:1;
passes. hdfs://master:9000/inputII/b.txt:1;
question        hdfs://master:9000/inputII/a.txt:1;
running hdfs://master:9000/inputII/a.txt:1;
set     hdfs://master:9000/inputII/b.txt:1;
shown   hdfs://master:9000/inputII/b.txt:1;
still   hdfs://master:9000/inputII/a.txt:1;
the     hdfs://master:9000/inputII/a.txt:2;hdfs://master:9000/inputII/b.txt:3;
think   hdfs://master:9000/inputII/b.txt:1;
this    hdfs://master:9000/inputII/a.txt:1;
times.  hdfs://master:9000/inputII/a.txt:1;
to      hdfs://master:9000/inputII/a.txt:3;
variables       hdfs://master:9000/inputII/b.txt:1;
version.        hdfs://master:9000/inputII/a.txt:1;
vi      hdfs://master:9000/inputII/b.txt:1;
with    hdfs://master:9000/inputII/b.txt:1;
wrong.  hdfs://master:9000/inputII/b.txt:1;
```

Tips：

1、 打包成 jar 包时不要使用 tar -xzvf 打包，不然运行 jar 包时会出现 zip file 的错误；

2、 如果你的 java 文件里面有 package 的定义，那么在输入函数名时要加上 package.InvertedIndex；

3、 输出格式不限，只要能有单词、文件名、数目即可。