Hadoop Lab Tutorial

1 Introduction

ECS has installed a small Hadoop cluster in the CO246 lab in the Cotton building to help students to try out some basic operations with a Hadoop Cluster. This tutorial is adapted from the hands-on tutorial created by the ECS cluster administrator.

2 Assumptions

It is assumed that you have a valid *ECS account* to access lab machines in CO246. If you experience any permission problems with your ECS account on the Hadoop cluster, please email me (*qi.chen@vuw.ac.nz*) for help.

Kerberos [1] is used for authentication in the Hadoop cluster. As long as you have logged into one computer of our Hadoop cluster (the list of computers is given in Section 4 below), you don't need to further provide your ECS crendentials while using the Hadoop functionalities.

3 Things to note

The text files and codes are written as though they belong to the user *fred* and assume that fred puts everything needed in his home directory /user/fred on the Hadoop Distributed File System (HDFS).

4 Cluster Setup

An 8-node Hadoop cluster was installed, which has:

1. co246a-1.ecs.vuw.ac.nz (DataNode)

[1] https://en.wikipedia.org/wiki/Kerberos_(protocol)

2. co246a-2.ecs.vuw.ac.nz (DataNode)
3. co246a-3.ecs.vuw.ac.nz (DataNode)
4. co246a-4.ecs.vuw.ac.nz (DataNode)
5. co246a-5.ecs.vuw.ac.nz (DataNode)
6. co246a-6.ecs.vuw.ac.nz (DataNode)
7. co246a-7.ecs.vuw.ac.nz (DataNode)
8. co246a-8.ecs.vuw.ac.nz (DataNode)
9. co246a-9.ecs.vuw.ac.nz (YARN resource manager)
10. co246a-a.ecs.vuw.ac.nz (NameNode)

NOTE: There may be some benefit, during the Tutorial, if you arrange between yourselves to login to different computers within the Cluster to access the Hadoop resource.

5 Accessing cluster nodes from home

You can use "ssh" to access the cluster nodes from home. It is recommended to access one of the data nodes, i.e., co246a-1, co246a-2, ..., co246a-8.

Firstly, you have to access your home directory in the ECS system. Notice: replace *fred* by your ECS username and enter your ECS password.

```
%ssh fred@barretts.ecs.vuw.ac.nz
```

From your home directory, you can access one of the cluster nodes (co246a-1 in the following command).

```
%ssh co246a-1
```

Now you are ready to set up environmental variables for Hadoop clusters.

6 Environmental variables

In order to access the Hadoop cluster from the command line, it will be useful to set a couple of environmental variables and add an extra component to the local machine's shell's default path.

```
%export HADOOP_VERSION=3.3.6
%export HADOOP_HOME=/local/Hadoop/hadoop-$HADOOP_VERSION
%export PATH=${PATH}:$HADOOP_HOME/bin
%export HADOOP_CONF_DIR=${HADOOP_HOME}/etc/hadoop
```

One way to achieve this is to copy the above three lines into a file, eg HadoopSetup.csh and then just source it before you start to call any Hadoop commands in the shell.

```
%source HadoopSetup.csh
```

You should check if the commands run properly by printing out these variables.

```
%echo $HADOOP_VERSION
%echo $PATH
```

You will also need to get the basic Java utilities onto your PATH by running:

```
%need java8
```

It is VERY IMPORTANT to call this command first before using any Hadoop functionalities.

7 Manipulating files within the HDFS

7.1 Browsing

To look at the files at the root of the HDFS, we type:

```
%hdfs dfs -ls /
```

or:

```
%hadoop fs -ls /
```

We can also look at the /user directory

```
%hdfs dfs -ls /user
```

What happend if you take a peek at someone else's directory?

```
%hdfs dfs -ls /user/otheruser
```

Let's try accessing the HDFS using the URI-syntax.

```
%hdfs dfs -ls hdfs://co246a-a.ecs.vuw.ac.nz:9000/user
```

Note that co246a-a.ecs.vuw.ac.nz is the NameNode of HDFS as mentioned earlier.

7.2 Moving Data Between HDFS and Local File System

First of all, let's create a couple of files that we'll run a program against later on.

```
%echo 'Hello World Bye World' > file01
%echo 'Hello Hadoop Goodbye Hadoop' > file02
```

and then make somewhere to put them, inside the HDFS.

```
%hdfs dfs -mkdir /user/fred/input
```

or, using the URI-style, as

```
%hdfs dfs -mkdir hdfs://co246a-a.ecs.vuw.ac.nz:9000/user/fred/input
```

Now we have somewhere to put the files, let's put the data in and see what's there

```
%hdfs dfs -put file01 input
%hdfs dfs -ls input
```

Note that the put has left the file on the OS filesystem in place. Recalling that put and get are the equivalents of cp across the OS/HDFS boundary. What about the equivalent of mv?

```
%hdfs dfs -moveFromLocal file02 input
%hdfs dfs -ls input
```

Sadly, the reciprocal command, moveToLocal, is not supported by our Hadoop cluster.

```
%hdfs dfs -moveToLocal input/file02 .
```

Option '-moveToLocal' is not implemented yet. So we need to do a move out of the HDFS using two commands:

```
%hdfs dfs -copyToLocal input/file02 .
%hdfs dfs -rm -skipTrash input/file02
```

One final thing to note:

```
%hadoop fs -put file02 input/file01
=> gives an error: put: Target input/file01 already exists
```

As you can see, Hadoop, being a framework for manipulating data, usually doesn't like you trying to overwrite its data.

8 Compile and Run a MapReduce Program

Below are the shell commands for compiling our Java program and building the JAR executable.

8.1 Setting Environment Variables and CLASSPATH For Compilation

```
%export
    LD_LIBRARY_PATH=$HADOOP_PREFIX/lib/native:$JAVA_HOME/jre/lib/amd64/server
```

and, since Hadoop is written in Java, you also need to set a CLASSPATH environmental variable. A simple way to see what CLASSPATH a Hadoop installtion requires is to type:

```
%hadoop classpath --glob
```

and then recoil in horror at the size of it.

It might be obvious to some of you that an easy way to use the CLASSPATH info above is to pipe the output into a file; edit it so as to add an setenv CLASSPATH at the start of the one massively long line in the file, and then source that.

```
%hadoop classpath --glob > SetupHadoopClasspath.csh
%vim SetupHadoopClasspath.csh
```

You need to change the one line in that file:

- Add to the beginning of the line the words: setenv CLASSPATH
- Remove the last component of the line, i.e. :/contrib/capacity-scheduler/*.jar

Then source it:

```
%source SetupHadoopClasspath.csh
```

8.2 Compile and Run WordCount.java Program

Given the program *WordCount.java* is in the current folder. Let's compile it:

```
%mkdir wordcount_classes
%javac -d wordcount_classes WordCount.java
%jar cvf wordcount.jar -C wordcount_classes/ .
%ls
%ls -R wordcount_classes
```

Recalling that we put two files file01 and file02 into the HDFS folder named *input*, it might be tempting to do the following:

```
%hadoop jar wordcount.jar MapReduceExample.WordCount input .
```

which tells Hadoop to use our home directory as the output directory, but, once again, Hadoop will not be happy about this, and we need to allow Hadoop to create a specific output directory just for this job.

```
%hadoop jar wordcount.jar MapReduceExample.WordCount input output
```

What do we see in the HDFS

```
 %hdfs dfs -ls output
```

We will see the output of the program in:

```
%hdfs dfs -cat output/part-r-00000
```

Let's analyse the given program *WordCount2.java* and run on the previous input files. What are the differences in the output?

9 Running a Streaming MapReduce from the shell

Hadoop streaming is a utility that comes with the Hadoop distribution. The utility allows you to create and run Map/Reduce jobs with any executable or script as the mapper and/or the reducer. For example, given an input file containing tab separated text as follows.

```
alice        50
bruce        70
charlie      80
dan          75
```

Running the following command will return the second column of the content.

```
%hadoop jar
    $HADOOP_PREFIX/share/hadoop/tools/lib/hadoop-streaming-3.3.6.jar
    -input input -output output -mapper 'cut -f2' -reducer 'cat'
```

where the LINUX commands (cut and cat) are used as mapper and reducer for the job.

10 Running MapReduce on a Bigger Data Set

The NBER data[2] is a big data set, approximatley 250MB with about 16.5 million rows. It is part of a freely available set of files, made available by the US National Bureau of Economic Research, http://www.nber.org/patents/ and serves to describe US patent citations covering a period of pattens issued from 1970s to the 1990s.

The data set we will use, *cite75_99.txt*, has been loaded into the HDFS:

```
%hdfs dfs -ls /tmp/patentin
```

Let's explore the data set:

```
%head cite75_99.txt
%wc cite75_99.txt
```

The given program *PatentCitation.java* takes each pair of values in the original file, and constructs, for each patent, a list of which patents it cites. You might wish to think of this in terms of the header line of the original data

```
'CITING','CITED'
```

being inverted into

```
'CITED','CITING','CITING','CITING',...
```

You can build the program:

```
%mkdir patent_classes
%javac -d patent_classes PatentCitation.java
%jar cvf patent.jar -C patent_classes/ .
```

and run it following the usual conventions. However, note that the output directory needs to exist (i.e. you must create one if it doesn't exist).

```
%hadoop jar patent.jar MapReduceExample.PatentCitation /tmp/patentin
    /user/fred/patentout1
```

After it has run, you should see the output in /user/fred/patentout1 directory.

[2]The NBER Patent Citation Data File: Lessons, Insights and Methodological Tools. Hall, B. H., A. B. Jaffe, and M. Trajtenberg (2001). NBER Working Paper 8498.