Instructions For Running Map Reduce Program

**How to log in by SSH terminal?**

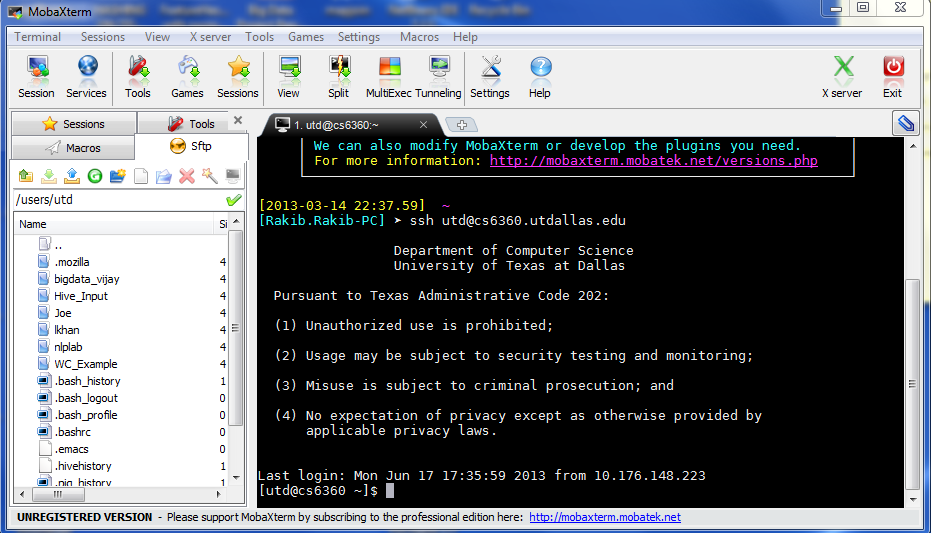
Please download an SSH client on your Windows laptops before you arrive to the workshop.  Here are the download URL's:  
  
      <http://mobaxterm.mobatek.net/MobaXterm_Setup_6.3.msi>  
      <http://www.hlt.utdallas.edu/MobaXterm_Setup_6.3.msi>

Log in **cs6360.utdallas.edu** with

Username: **utd**

Password: **hadoop.**

Run **MobaXterm** and type **utd@cs6360.utdallas.edu**

****

After log in verify your current directory by **pwd** command on ssh terminal. **pwd** shows your current directory **/users/utd**

Make sure that you have the **hadoop-core-1.0.4.jar** file in **/users/utd**. you can list all files and folder in **/users/utd** by **ls**command on terminal.

**Create a Folder by mkdir command with your name(e.g. Joe) inside /users/utd . Invoke your directory by issuing cd command on terminal .**

**In this document examples are shown using folder name 'Joe'. Please use your own name of folder in place of 'Joe' inside the commands.**

**How to Compile the .java file and build .jar file?**

You can compile java file and also build jar file by using any standard IDE like Eclipse / NetBeans or any stand java IDE.

On the other hand you can compile java file and then create jar file from command line on Putty terminal

Step 1:  **javac -classpath <hadoop-core-1.0.4.jar file path> <java file path>**

Step 2: **jar cvf <jar file> -C <manifest file directory> <class file directory>**

**Example: Word Count**

In Word Count example using Map Reduce,  counts the appearance of each word in a set of documents.

You will see **WC\_Example** in your parent directory **/users/utd/** and you will find files **file01**, **file02** inside **WC\_Example**. Your task is to count the frequency of each word for these two files.

1. Make input folder in HDFS and copy file01 and file02

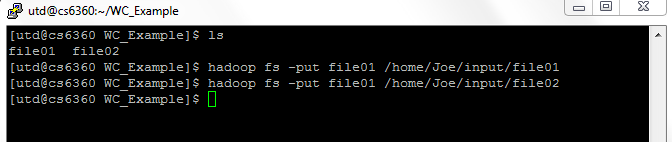
**hadoop fs -put file01 /home/<your\_name>/input/file01**

**hadoop fs -put file01 /home/<your\_name>/input/file02**

e.g.

**hadoop fs -put file01 /home/Joe/input/file01**

**hadoop fs -put file01 /home/Joe/input/file02**

****

2. Go to your directory (Joe) and create a folder **WC**.

3. Inside WC folder copy/upload **WordCount.java** by WinSCP

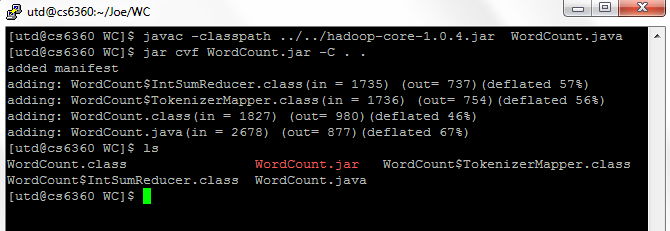
Source code of WordCount.java can be found later.

4. Run the following commands:

**javac -classpath ../../hadoop-core-1.0.4.jar WordCount.java**

**jar cvf WordCount.jar -C . .**

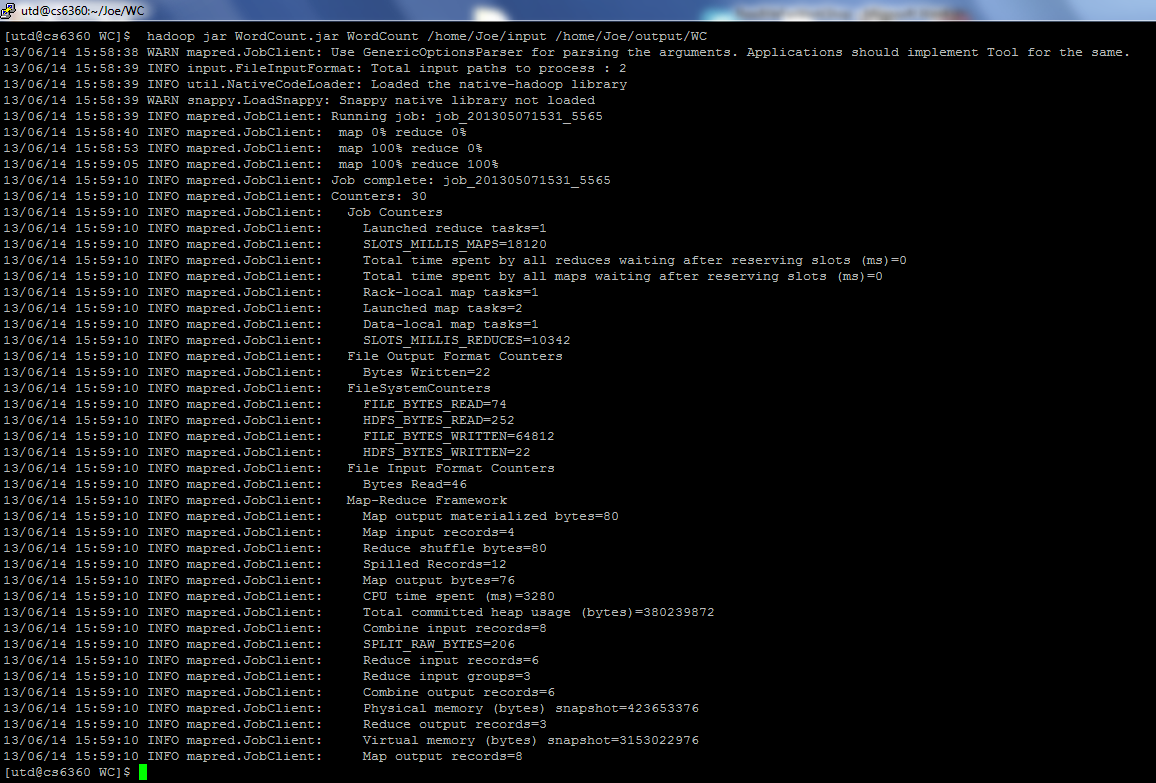
4. You will see a jar file named WordCount.jar is created in WC folder.



5. Run map reduce program

hadoop jar WordCount.jar WordCount /home/Joe/input /home/Joe/output/WC

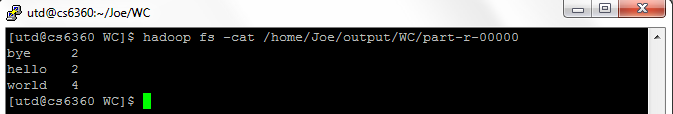
This WordCount program takes the input files location (/home/Joe/input) and output file location (/home/Joe/output/WC).



6. You will see the **part-r-00000** file in **/home/Joe/output/WC** directory. The following command will give you the output.

**hadoop fs -cat /home/Joe/output/WC/part-r-00000**

This will give the following output.



**Hands on Exercise**

In this exercise you will learn how to solve problems using Map Reduce.

Please apply Hadoop mapreduce to derive some statistics from White House Visitor Log. There are currently 2.9 million records available at

<http://www.whitehouse.gov/briefing-room/disclosures/visitor-records>

Data is available as web only spreadsheet view and downloadable raw format in CSV (Comma Separated Value). In CSV format each column is separated by a comma “,” in each line. The first line represents the heading for the corresponding columns in other lines. We are going to use this raw data for our mapreduce operation.

Download the 4 CSV files (zipped) from the site (at the bottom of the page), unzip, and copy to the UTD’s HDFS directory - “/home/<your\_utd\_netid>/whitehouse/input”. Use the *put* or *copyFromLocal* HDFS shell command to copy those files. Also produce the output to “/home/<your\_name>/output” HDFS directory. For more information about HDFS shell, see here -

<http://hadoop.apache.org/docs/stable/file_system_shell.html>

You are required to write efficient Hadoop ***MapReduce programs in Java*** to find the following information:

**N.B.** The customized dataset will be found at **/WH/input** and you should use this as input.

You are required to write efficient Hadoop ***MapReduce programs in Java*** to find the following information:

*(i) The 10 most frequent visitors (NAMELAST, NAMEFIRST) to the White House.*

*(ii) The 10 most frequently visited people (visitee\_namelast, visitee\_namefirst) in the White House.*

*(iii) The 10 most frequent visitor-visitee combinations.*

Consider mapreduce chaining for efficiency. See the possible ways for chaining in Yahoo’s tutorial and the stackoverflow discussion

<http://developer.yahoo.com/hadoop/tutorial/module4.html#chaining>

<http://stackoverflow.com/questions/3059736/map-reduce-chainmapper-and-chainreducer>

**(i) The 10 most frequent visitors (NAMELAST, NAMEFIRST) to the White House.**

1. Create a folder WH inside folder Joe(folder created earlier).

**mkdir WH**

2. Inside WH folder copy/upload WhiteHouse.java by WinSCP

3. Run the following commands:

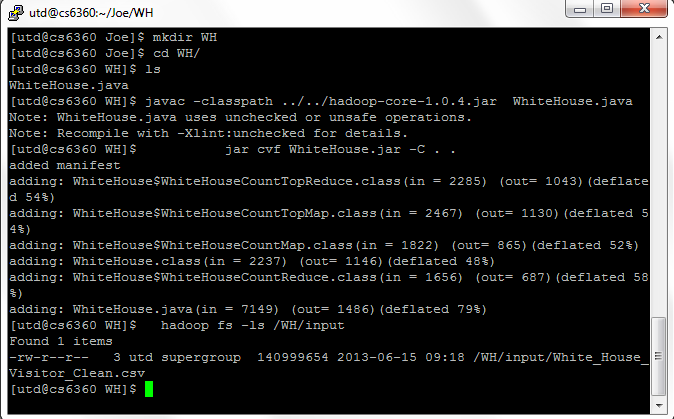
**javac -classpath ../../hadoop-core-1.0.4.jar WhiteHouse.java**

**jar cvf WhiteHouse.jar -C . .**

4. You will see a jar file named WhiteHouse.jar is created in WH folder

5. Make sure that input files are inserted proper directory.

**hadoop fs -ls /WH/input**

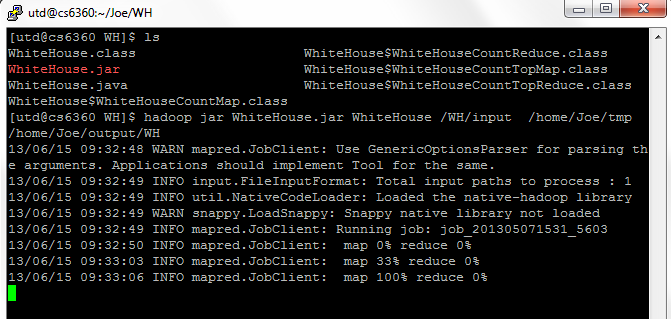
****

6. Run map reduce program

**hadoop jar WhiteHouse.jar WhiteHouse /WH/input /home/Joe/tmp /home/Joe/output/WH**

(For chaining purpose the java program requires three file locations. First is input file location, the second is the intermediate file location and the third one is the final output file location.)

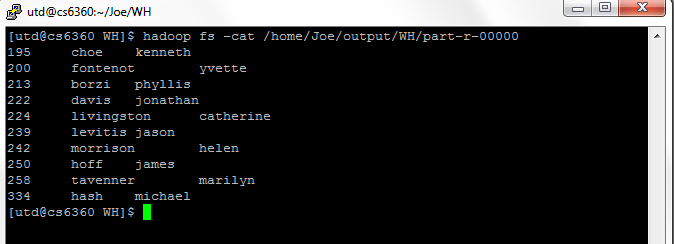
This will gives you following which indicates that map and reduce job status

****

7. You will see the **part-r-00000** file in **/home/Joe/output/WH** directory. The following command will give you the output.

**hadoop fs -cat /home/Joe/output/WH/part-r-00000**

This will give the following output.



**WordCount.java**

import java.io.IOException;

import java.util.StringTokenizer;

import org.apache.hadoop.fs.FileSystem;

import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.io.IntWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Job;

import org.apache.hadoop.mapreduce.Mapper;

import org.apache.hadoop.mapreduce.Reducer;

import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;

import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

import org.apache.hadoop.util.GenericOptionsParser;

public class WordCount {

public static class TokenizerMapper

extends Mapper<Object, Text, Text, IntWritable>{

private final static IntWritable one = new IntWritable(1);

private Text word = new Text();

@Override

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);

}

}

}

public static class IntSumReducer

extends Reducer<Text,IntWritable,Text,IntWritable> {

private IntWritable result = new IntWritable();

/\*\*

\*

\* @param key

\* @param values

\* @param context

\* @throws IOException

\* @throws InterruptedException

\*/

@Override

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);

}

}

public static void main(String[] args) throws Exception {

Configuration conf = new Configuration();

if (args.length != 2) {

System.err.println("Usage: WordCount <in> <out>");

System.exit(2);

}

FileSystem fs = FileSystem.get(conf);

fs.delete(new Path(args[1]), true);

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(args[0]));

FileOutputFormat.setOutputPath(job, new Path(args[1]));

System.exit(job.waitForCompletion(true) ? 0 : 1);

}

}

**WhiteHouse.java**

import java.io.IOException;

import java.util.TreeMap;

import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.fs.FileSystem;

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.io.IntWritable;

import org.apache.hadoop.io.NullWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Job;

import org.apache.hadoop.mapreduce.Mapper;

import org.apache.hadoop.mapreduce.Reducer;

import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;

import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

/\*\*

\*

\* @author Rakib

\*/

public class WhiteHouse {

public static class WhiteHouseCountMap extends Mapper<Object, Text, Text, IntWritable>

{

private final static IntWritable one = new IntWritable(1);

private Text word = new Text();

@Override

public void map(Object key, Text value, org.apache.hadoop.mapreduce.Mapper.Context context) throws IOException, InterruptedException {

String[] container = value.toString().split(",");

//StringTokenizer tokenizer = new StringTokenizer(line, ",");

if (container.length > 0) {

String firstName = container[1].trim();

String lastName = container[0].trim();

if (!"NAMELAST".equals(lastName) && !"NAMEFIRST".equals(firstName)) {

String name = lastName + "\t" + firstName;

word.set(name);

context.write(word, one);

}

}

}

}

public static class WhiteHouseCountReduce extends Reducer<Text, IntWritable, Text, IntWritable> {

@Override

public void reduce(Text key, Iterable<IntWritable> values, Context context

) throws IOException, InterruptedException {

int sum = 0;

for (IntWritable val : values) {

sum += val.get();

}

context.write(key, new IntWritable(sum));

}

}

public static class WhiteHouseCountTopMap extends Mapper<Object, Text, NullWritable, Text> {

private TreeMap<Integer, Text> visitorToRecordMap = new TreeMap<Integer, Text>();

@Override

public void map(Object key, Text value, org.apache.hadoop.mapreduce.Mapper.Context context) throws IOException, InterruptedException {

String[] container = value.toString().split("\t");

if (container.length > 0)

{

String containerValue = container[2].trim() + "\t" + container[0].trim() + "\t" + container[1].trim();

visitorToRecordMap.put(Integer.parseInt(container[2].trim()), new Text(containerValue));

if (visitorToRecordMap.size() > 10)

{

visitorToRecordMap.remove(visitorToRecordMap.firstKey());

}

}

}

@Override

protected void cleanup(org.apache.hadoop.mapreduce.Mapper.Context context) throws IOException,

InterruptedException {

// Output our ten records to the reducers with a null key

for (Text t : visitorToRecordMap.values()) {

context.write(NullWritable.get(), t);

}

}

}

public static class WhiteHouseCountTopReduce extends Reducer<NullWritable, Text, NullWritable, Text> {

private TreeMap<Integer, Text> visitorToRecordMap = new TreeMap<Integer, Text>();

public void reduce(NullWritable key, Iterable<Text> values, Reducer.Context context

) throws IOException, InterruptedException {

for (Text val : values)

{

String value = val.toString();

String[] container = value.split("\t");

//StringTokenizer tokenizer = new StringTokenizer(line, ",");

if (container.length > 0)

{

String count = container[0].trim();

visitorToRecordMap.put(Integer.parseInt(count), new Text(value));

if (visitorToRecordMap.size() > 10)

{

visitorToRecordMap.remove(visitorToRecordMap.firstKey());

}

}

}

for (Text t : visitorToRecordMap.values()) {

//System.out.println(t.toString());

context.write(NullWritable.get(), t);

}

}

}

public static void main(String[] args) throws Exception {

Configuration conf = new Configuration();

if (args.length != 3) {

System.err.println("Usage: whitehouse <in> <temp> <out>");

System.exit(2);

}

FileSystem fs = FileSystem.get(conf);

fs.delete(new Path(args[1]), true);

fs.delete(new Path(args[2]), true);

Job countingJob = new Job(conf, "whitehouse\_count");

countingJob.setJarByClass(WhiteHouse.class);

countingJob.setMapperClass(WhiteHouseCountMap.class);

countingJob.setCombinerClass(WhiteHouseCountReduce.class);

countingJob.setReducerClass(WhiteHouseCountReduce.class);

countingJob.setOutputKeyClass(Text.class);

countingJob.setOutputValueClass(IntWritable.class);

FileInputFormat.addInputPath(countingJob, new Path(args[0]));

FileOutputFormat.setOutputPath(countingJob, new Path(args[1]));

if (countingJob.waitForCompletion(true))

{

Job countingTopJob = new Job(conf, "whitehouse\_top");

countingTopJob.setJarByClass(WhiteHouse.class);

countingTopJob.setMapperClass(WhiteHouseCountTopMap.class);

countingTopJob.setCombinerClass(WhiteHouseCountTopReduce.class);

countingTopJob.setReducerClass(WhiteHouseCountTopReduce.class);

countingTopJob.setOutputKeyClass(NullWritable.class);

countingTopJob.setOutputValueClass(Text.class);

FileInputFormat.addInputPath(countingTopJob, new Path(args[1]));

FileOutputFormat.setOutputPath(countingTopJob, new Path(args[2]));

System.exit(countingTopJob.waitForCompletion(true) ? 0 : 1);

}

}

}