**1. Getting started with Hadoop**

In this practical you will learn how to:

1. Add, manipulate and extract data from HDFS

2. Create and compile a Java program to run MapReduce

3. Run the MapReduce program and extract the results from HDFS

## Practical 3 - Get started with Hadoop

### 2. Accessing HDFS

Hadoop has two main components - HDFS and MapReduce.

In your practical work you will be using a virtual machine running on your computer - a virtual linux machine.  This virtual machine has its own file system.

You are running Hadoop on this virtual machine, and this creates a single node virtual cluster on your virtual machine.

Hadoop maintains the HDFS within your virtual linux machine.

To get data to HDFS you need to be able to move it from your linux virtual machine's file system to HDFS.  Typically this means you need to get it onto your linux virtual machine and then onto HDFS.

To access HDFS from within your linux virtual machine, use the Hadoop FS Shell commands.  The documentation on apache.org is at:

[Hadoop Shell Commands](http://hadoop.apache.org/docs/current/hadoop-project-dist/hadoop-common/FileSystemShell.html)

I will describe use of some of these commands as if you are logged into the Cloudera Hadoop virtual machine. This virtual machine has a both a Linux user and a hadoop user with name of "cloudera" and password "cloudera".

* on the Unix file system the user has a home directory of: **/home/cloudera**
* on the HDFS file system there is a home directory for the Hadoop user of: **/user/cloudera**
* **In case you installed hadoop according to the guidelines on prac2 then make a directory /prac3 in the most convenient location for you and modify the paths in the commands below accordingly.**

In the previous practical, I believe all of you have tried to play with some of the linux commands, now lets go to the Hadoop Shell commands. You keep working in the same terminal window but now you address Hadoop file system - HDFS.

**1. Create a subdirectory on your HDFS home.**

You are going to be working on different MapReduce programs, so you can create different areas on HDFS for each program.  For the first program you are going to run, create a subdirectory in your home directory on HDFS.  To do this:

hadoop fs -mkdir /user/cloudera/prac03

Note:  If you encounter error message “mkdir: Cannot create directory /your directory. Name node is in safe mode.”, this is because Cloudera 5.13 by default have put HDFS into safe mode. So input “sudo -u hdfs hdfs dfsadmin -safemode leave” to close safe mode of HDFS.

To store the input files for this first program, now use a similar command to create subdirectory called input of this directory  (so the directory is /user/cloudera/prac03/input)

Example above (as all following examples) assume that you work on Cloudera image and your login (or user name) is "cloudera". You have the same login/user name in Linux system and in HDFS, so in both cases you "home" directory is "cloudera".

Most examples will use full path to all files and folders, to make it easier to understand where you are at the moment - in Hadoop or in Linux. As mentioned above, if you are in Hadoop, then your home directory is "**user/cloudera**"; if you are in Linux, then home directory is "**home/cloudera**".

However, it is not compulsory to use full path, you can use relative path, so the above command might look like:

hadoop fs -mkdir prac03

Result will be the same. Please be careful, you can run this command only once, you can not create new directory with the same name. If you want to play more, then you have to set new name for the new directory or delete the old directory.

**2. Listing files in HDFS**

You can list files in directories on your HDFS file system.  Try:

hadoop fs -ls /user/cloudera/prac03

**3. Create subdirectories**

Create a mirrored set of subdirectories on your Linux file system - /home/cloudera/prac03 and /home/cloudera/prac03/input

We are going to need some more, so also create /home/cloudera/prac03/src and /home/cloudera/prac03/classes

**4. Copy files to HDFS.**

Download the file01 from the course website and place it in the /home/cloudera/prac03/input directory on the Unix file system.  We want to copy it to the HDFS file system to the location /user/cloudera/prac03/input.  To do this use:

hadoop fs -put /home/cloudera/prac03/input/\* /user/cloudera/prac03/input

Relative path version of this command is (assuming that you are inside your home folder on Linux):

hadoop fs -put prac03/input/\* prac03/input

... or even shorter if you are inside folder input on your local Linux system:

hadoop fs -put \* prac03/input

Compare these three versions of the same command. The first one is easy to understand but too long to type. The last one is easy to type, but you should carefully control location where you type this command.

Note: For students who wish to transfer files between your physical computer and the VM, you can use tools like FileZilla to achieve this goal. Follow the process described in the following link, “Recommended VirtualBox Configurations” section: https://softwaresanders.wordpress.com/2016/10/24/getting-started-with-the-cloudera-quickstart-vm/  to configure your VM.

**5. Getting files from HDFS back to the Unix file system.**

When you run a MapReduce program on Hadoop, you will specify an output director in HDFS and Hadoop will put the results there.  You will typically use a subdirectory output of your program working directory.  To get these files back to your Unix file system you can use:

hadoop fs -get /user/cloudera/prac03/output /home/cloudera/prac03/

**6.  Removing files and directories on HDFS.**

You have run your hadoop MapReduce program, and you want to remove the output directory.  You need to do this, or specify a different output directory if you want to run the MapReduce program again.

hadoop fs -rm -r /user/cloudera/prac03/output

This will remove the files in the directory plus the directory. You can use the rm command without the -r to remove a single file.

### . Creating and compiling a MapReduce program

**1. Create the Java source files.**

You now have a unix file system directory /home/cloudera/prac03/src  (or a variation of this name depending on which virtual machine you are using).

In this directory create the files below.

**Note**, as copying the files into web based editor used to create this page did a few small changes to the text (to handle characters that are part of the syntax of HTML), copying the text below does not work well, and I have provided the files for you to download (see this Practical download files or links below).  You can do this easily from a browser running inside of your virtual machine.

1. [AggregateJob.java](https://lo.unisa.edu.au/pluginfile.php/2365569/mod_book/chapter/235512/AggregateJob.java)

import org.apache.hadoop.conf.Configured;

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.io.LongWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Job;

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

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

//import org.apache.hadoop.mapreduce.lib.reduce.LongSumReducer;

import org.apache.hadoop.util.Tool;

import org.apache.hadoop.util.ToolRunner;

public class AggregateJob extends Configured implements Tool {

@Override

public int run(String[] args) throws Exception {

Job job = new Job(getConf());

job.setJarByClass(getClass());

job.setJobName(getClass().getSimpleName());

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

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

job.setMapperClass(ProjectionMapper.class);

//job.setCombinerClass(LongSumReducer.class);

//job.setReducerClass(LongSumReducer.class);

job.setReducerClass(SumReducer.class);

job.setOutputKeyClass(Text.class);

job.setOutputValueClass(LongWritable.class);

return (job.waitForCompletion(true) ? 0 : 1);

}

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

int rc = ToolRunner.run(new AggregateJob(), args);

System.exit(rc);

}

}

2. [ProjectionMapper.java](https://lo.unisa.edu.au/pluginfile.php/2365569/mod_book/chapter/235512/ProjectionMapper.java)

import org.apache.hadoop.io.LongWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Mapper;

import java.io.IOException;

public class ProjectionMapper extends Mapper<LongWritable, Text, Text, LongWritable> {

private Text word = new Text();

private LongWritable count = new LongWritable();

@Override

protected void map(LongWritable key, Text value,  
 Context context)throws IOException, InterruptedException {

// value tab separated: word, year, occurrences, #books, #pages

// output (word, occurrences) so we can sum over all years

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

word.set(split[0]);

if (split.length > 2) { try {

count.set(Long.parseLong(split[2]));

context.write(word, count);

} catch (NumberFormatException e) {

// cannot parse – put out error msg for debugging

word.set("<<problem with number>>");

count.set(1);

context.write(word, count);

       }

     }else{

// another debugging line

word.set("<<problem parsing line>>");

count.set(1);

context.write(word, count);

}

}

}

3. [SumReducer.java](https://lo.unisa.edu.au/pluginfile.php/2365569/mod_book/chapter/235512/SumReducer.java)

import java.io.IOException;

import org.apache.hadoop.io.LongWritable;

import org.apache.hadoop.mapreduce.Reducer;import org.apache.hadoop.io.Text

public class SumReducer extends Reducer<Text, LongWritable, Text, LongWritable> {

private LongWritable result = new LongWritable();

public void reduce(Text key, Iterable<LongWritable> values,  
 Context context) throws IOException, InterruptedException {

long sum = 0;

for (LongWritable val : values) {

sum += val.get();

}

result.set(sum);

context.write(key, result);

}

}

**2. Compile the Java source files**

While you are inside the /home/cloudera/prac03 directory use the command:

javac -cp /usr/lib/hadoop/\*:/usr/lib/hadoop/client-0.20/\* -d classes src/\*.java

This will compile the three Java source files and create three .class files in the classes subdirectory that you created earlier.

It is easiest now to create a jar file that packages the three class files together:

jar -cvf prac03.jar -C classes/ .

Don't forget the '.' at the end - it is telling the jar command to put the jar file in the current directory.  You should now have prac03.jar in the current directory.

If you have problems getting the files compiled (if there are errors), or if  you change the files and recompile, delete the jar file before you re-create it.

You now have a MapReduce program ready to run.

### 4. Running the MapReduce program.

You have set up the data file on HDFS, and created and compiled the Java program on the Unix file system, creating a jar file from the compiled classes.

Now try to run the program:

hadoop jar prac03.jar AggregateJob /user/cloudera/prac03/input /user/cloudera/prac03/output

You are setting the MapReduce jar file (prac03.jar), saying that the class AggregateJob inside this jar file is the one to start executing, that the HDFS input is in /user/cloudera/prac03/input and that the MapReduce should write the output to /user/cloudera/prac03/output.

### 5. Getting the results back from HDFS

You have run the hadoop MapReduce program, and it has written the results to the directory:

/user/cloudera/prac03/output

The directory should contain a files

part-r-00000  
\_SUCCESS

and a directory

\_logs

The latest version of Cloudera image might not have \_log directory. That is OK, we don't need it anyway.

The actual output of the MapReduce is i n the **part-r-00000** file.

You can view it within HDFS by:

hadoop fs -cat /user/cloudera/prac03/output/part-r-00000

It should contain:

dobbs 42

doctor 7612391

You can copy this back to the Unix file system using:

hadoop fs -get /user/cloudera/prac03/output .

This copies the whole directory tree.  You can just copy file by first creating the output subdirectory and then copying just the part\* files from HDFS to your local output directory.