MapReduce & YARN
Hands-on Lab Exercise 2

More MapReduce Programming

IBM

Contents

Lab 2   Additional MapReduce Programming

For this exercise, you must have completed the Lab setup instructions for the cloud.

In this exercise, you will work a more complex MapReduce program where you will add features as well as open up your environment to a range of other programs.

After completing this hands-on lab, you'll be able to:

- Compile and run a wider variety of MapReduce programs

Allow 45 minutes to complete this lab; more if you would like to explore further.

This version of the lab was designed using the cloud version of the IBM BigInsights 4.1 Quick Start Edition.

In Hands-on Lab Exercise 1 you explored the process of creating a MapReduce application. It was relatively easy since all the code was provided. We will continue that theme, but encourage you to branch out to find code that is more challenging and that provides opportunities to learn more about MapReduce. **You are encouraged to experiment further, depending on your skill in the Java language.**

You will build upon what you learned in Exercise 1 and we will presume that you have mastered the skills that you learned there. In this Exercise we will give less detail in the instructions in order to give you the opportunity to practice the skills and knowledge gained in Exercise 1, and to experiment more.

You will also learn where to find additional material to continue your experimentation.

2.1 Start your PuTTY

It is assumed that you have downloaded PuTTY, WinSCP and completed the setup and configuration in the Lab setup.

If you exited your PuTTY, re-start it and re-connect back to the cloud. If you need assistance on this task, consult the Lab setup.

The terminal window opens and will let you act with your environment.

2.2 Obtain MapReduce programs and data from the Internet

The classic MapReduce program is WordCount. This to Hadoop/MapReduce is equivalent to what "Hello, World" is to C & Java programming.

__1. Create a directory to hold the three Java files in your home directory that you will be making and make it accessible. The directory will be used to hold program artifacts and to separate it from the other things in the file system.
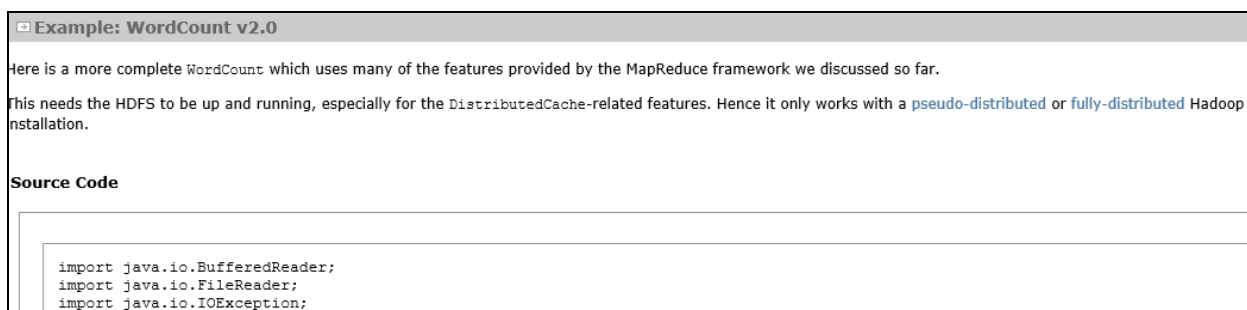
```
cd ~

mkdir wordcount

cd wordcount
```

__2. Open the vi editor (or whichever editor you choose) to create a file called WordCount2.java. Type:

```
vi WordCount2.java
```

*Note: Press **i** to put the vi editor into insert mode.*

__3. Open a Web browser and navigate to the following URL.

http://hadoop.apache.org/docs/current/hadoop-mapreduce-client/hadoop-mapreduce-client-core/MapReduceTutorial.html#Example:_WordCount_v2.0

__4. Scroll down and locate the section on the page titled **Example: WordCount v2.0**.

> ⊟ **Example: WordCount v2.0**
>
> Here is a more complete WordCount which uses many of the features provided by the MapReduce framework we discussed so far.
>
> This needs the HDFS to be up and running, especially for the DistributedCache-related features. Hence it only works with a pseudo-distributed or fully-distributed Hadoop installation.
>
> **Source Code**
>
> ```
> import java.io.BufferedReader;
> import java.io.FileReader;
> import java.io.IOException;
> ```

__5. Then, using an editor of your choice (vi editor or notepad with WinSCP), copy the source code into your WordCount2.java file.

__6. Before you quit the editor, scroll down to the main ( ) function and notice the line of code

```
if (!(remainingArgs.length != 2 | | remainingArgs.length != 4)) {
```

There is an extra space between the pipe symbols in the concatenate function. Remove that space or else you will get a compile error.

> Note: If using vi editor, to write and quit the editor by pressing **Esc**, typing **:wq**, and then pressing **Enter**. You can copy and paste into PuTTY using the right click as a paste.
>
> If using WinSCP, just drag the file to the correct location.

__7. Compile the program using the Java Compiler, and examining the classes created by your compile operation. Type:

```
javac –cp `hadoop classpath` WordCount2.java

ls
```

```
[wongk@iop-bi-master wordcount]$ javac -cp `hadoop classpath` WordCount2.java
[wongk@iop-bi-master wordcount]$ ls
WordCount2.class                 WordCount2$TokenizerMapper.class
WordCount2$IntSumReducer.class   WordCount2$TokenizerMapper$CountersEnum.class
WordCount2.java
[wongk@iop-bi-master wordcount]$
```

Notice the various classes that were created by the compile operation. You will run this later against MapReduce, but for now, create the appropriate jar file.

__8. Create the Java Archive File and list the manifest. Type:

```
jar cf WC2.jar *.class

jar tf WC2.jar
```

```
[wongk@iop-bi-master wordcount]$ jar cf WC2.jar *.class
[wongk@iop-bi-master wordcount]$ jar tf WC2.jar
META-INF/
META-INF/MANIFEST.MF
WordCount2.class
WordCount2$IntSumReducer.class
WordCount2$TokenizerMapper.class
WordCount2$TokenizerMapper$CountersEnum.class
[wongk@iop-bi-master wordcount]$
```

__9. The Java Archive File was created in the directory where the .java and .class files reside. But when we use Hadoop MapReduce to run the jar, Hadoop does not like to have the .class files in the same directory. Therefore you want to move the file to the parent directory, where we will run it in the next step:

```
cp *.jar ..

cd ..

ls -al
```

```
-rw-------    1 wongk wongk      29 Jan 18 09:05 myfile.txt
-rw-------    1 wongk wongk      41 Jan 18 12:23 myparams
-rw-------    1 wongk wongk     592 Jan 18 18:57 .mysql_history
drwx------    3 wongk wongk    4096 Jan 18 12:23 oozieWF
-rw-------    1 wongk wongk    3063 Jan 18 11:50 .pig_history
-rw-------    1 wongk wongk     158 Jan 18 12:33 pig.script
drwx------    2 wongk wongk    4096 Feb 25 11:30 sampledata
drwx------    2 wongk wongk    4096 Jan 18 09:48 .ssh
-rw-------    1 wongk wongk      47 Jan 28 12:04 test.txt
drwx------    2 wongk wongk    4096 Jan 28 11:35 Transfered
-rw-------    1 wongk wongk    4070 Feb 25 16:50 .viminfo
-rw-------    1 wongk wongk    5502 Feb 25 16:53 WC2.jar
drwx------    2 wongk wongk    4096 Feb 25 16:52 wordcount
[wongk@iop-bi-master ~]$
```

2.3　Obtain MapReduce data from the Internet

You will need some data. You can get text files of books from **www.Gutenberg.org**, from which you should download files in UTF-8 format (Unicode, i.e., text files). If you decide to edit any file to remove text prior to the title, do so with an editor such as TextPad (www.textpad.com) and save the file in UNIX/Linux format — Windows uses carriage return and line feed ("\r\n") as a line ending, which Unix uses just line feed ("\n").

Use the following for naming conventions:

- No spaces in the file name(s)
- Suffix of .txt on the file name(s) — not necessary, but convenient.

__1.　Open a Web browser, navigate to **http://www.gutenberg.org** , and then click the **Book Search Page** link in the site menu.

__2.　In the Search text box with the spyglass icon, type **Iliad** and press **Enter**.

__3.　In the search results, locate *The Iliad of Homer (1873) by Homer* item and click it.

__4.　On the Download page, locate and click the **Plain Text UTF-8** item that is 1.1MB in size.

__5.　Save the file from the browser to your shared folder. From the browser menu, click **File** -> **Save as**, and name it **Iliad.txt**.

__6.　Now open WinSCP (if it wasn't already open) and transfer the Iliad.txt file to your home directory by dragging and dropping it.
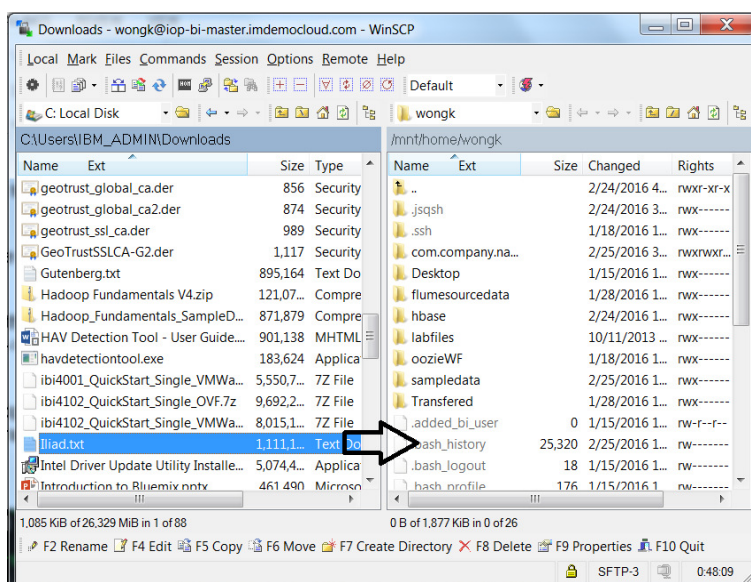
__7.   Upload the Iliad.txt file to the hadoop file system. Type:

```
cd ~

hdfs dfs -put Iliad.txt sampledata
```

Next, you want to see if your upload was successful.

__8.   To verify that your upload was successful, just type:

```
hdfs dfs -ls sampledata
```

```
[wongk@iop-bi-master ~]$ hdfs dfs -ls sampledata
Found 4 items
-rw-rw----+  3 wongk wongk    1092040 2016-02-25 17:02 sampledata/Iliad.txt
-rw-rw----+  3 wongk wongk     240900 2016-02-25 14:00 sampledata/SumnerCountyTe
mp.dat
drwxrwx---+  - wongk wongk          0 2016-02-25 14:01 sampledata/TempOut
drwxrwx---+  - wongk wongk          0 2016-02-25 14:11 sampledata/TempOut2
[wongk@iop-bi-master ~]$
```

Notice that Iliad.txt was successfully uploaded to the sampledata directory in the hadoop file system.

__9.   In your Web browser, navigate to the following link and read the directions on using the WordCount2:

●   http://hadoop.apache.org/docs/current/hadoop-mapreduce-client/hadoop-mapreduce-client-core/MapReduceTutorial.html#Example:_WordCount_v2.0

You will see that it can use a parameter file such as patternsToSkip.txt. You will use your host operating system to create such a file with the following contents. Use one entry per line:

```
\.
\,
\!
\'
\"
```

__10.   Now, create a text file and name it patternsToSkip.txt with the above contents in it. You may choose any editor you would like (vi editor or notepad with WinSCP).

Note: If you use Wordpad, select the **Unicode Text Document (.txt)** option in the Save as type box.

Next, you want to upload the skip patterns file into the hadoop file system.

__11.   Upload the patternsToSkip.txt file to the hadoop file system by going to your containing folder of patternsToSkip.txt and type:

```
hdfs dfs -put patternsToSkip.txt sampledata
```

__12.   To verify that your skip file was uploaded correctly. Type:

```
hdfs dfs -ls sampledata
```

```
[wongk@iop-bi-master ~]$ hdfs dfs -ls sampledata
Found 5 items
-rw-rw----+  3 wongk wongk     1092040 2016-02-25 17:02 sampledata/Iliad.txt
-rw-rw----+  3 wongk wongk      240900 2016-02-25 14:00 sampledata/SumnerCountyTe
mp.dat
drwxrwx---+  - wongk wongk           0 2016-02-25 14:01 sampledata/TempOut
drwxrwx---+  - wongk wongk           0 2016-02-25 14:11 sampledata/TempOut2
-rw-rw----+  3 wongk wongk          16 2016-02-25 17:07 sampledata/patternsToSkip
.txt
[wongk@iop-bi-master ~]$
```

Notice that patternsToSkip.txt was successfully uploaded to the sampledata directory in the hadoop file system.

With a little experience, you may wish to add your own entries and experiment with the skip patterns.

You are now ready to run WordCount2 against the text files that you uploaded to the /sampledata directory on HDFS.

__13.   As the hdfs user, navigate to the directory containing the .jar file you created earlier and list the files in that directory. Type:

```
cd ~
```

```
ls
```

```
[wongk@iop-bi-master ~]$ cd ~
[wongk@iop-bi-master ~]$ ls
BDU_MapReduce_and_YARN.tar  Iliad.txt    oozieWF                  Transfered
com.company.name            labfiles     patternsToSkip.txt  WC2.jar
Desktop                     MaxMT.jar    pig.script               wordcount
flumesourcedata             myfile.txt   sampledata
hbase                       myparams     test.txt
[wongk@iop-bi-master ~]$
```

__14.   Run the JAR file. This is entered all in one line before pressing Enter. Type:

```
hadoop jar ./WC2.jar WordCount2 -Dwordcount.case.sensitive=false
sampledata/Iliad.txt sampledata/wcout -skip
sampledata/patternsToSkip.txt
```
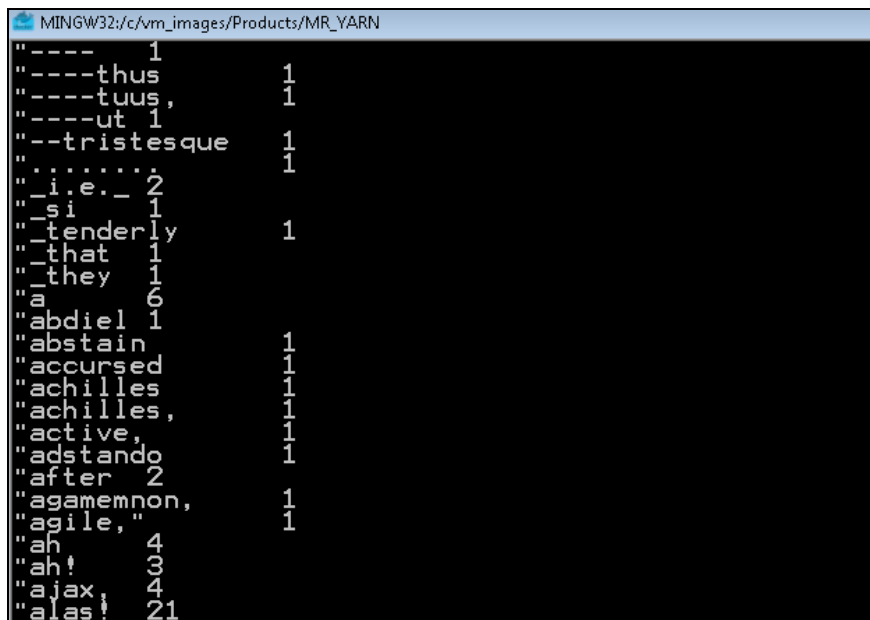
```
MINGW32:/c/vm_images/Products/MR_YARN
[hdfs@rvm ~]$ hadoop jar ./WC2.jar WordCount2 -Dwordcount.case.sensitive=false /
sampledata/Iliad.txt /sampledata/wcout -skip /sampledata/patternsToSkip.txt
15/09/17 17:43:34 INFO impl.TimelineClientImpl: Timeline service address: http:/
/0.0.0.0:8188/ws/v1/timeline/
15/09/17 17:43:36 INFO client.RMProxy: Connecting to ResourceManager at rvm.svl.
ibm.com/172.17.0.2:8050
15/09/17 17:43:43 INFO input.FileInputFormat: Total input paths to process : 1
15/09/17 17:43:43 INFO mapreduce.JobSubmitter: number of splits:1
15/09/17 17:43:45 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_14
38799705772_0016
15/09/17 17:43:47 INFO impl.YarnClientImpl: Submitted application application_14
38799705772_0016
15/09/17 17:43:48 INFO mapreduce.Job: The url to track the job: http://rvm.svl.i
bm.com:8088/proxy/application_1438799705772_0016/
15/09/17 17:43:48 INFO mapreduce.Job: Running job: job_1438799705772_0016
15/09/17 17:44:41 INFO mapreduce.Job: Job job_1438799705772_0016 running in uber
 mode : false
15/09/17 17:44:41 INFO mapreduce.Job:  map 0% reduce 0%
15/09/17 17:45:20 INFO mapreduce.Job:  map 67% reduce 0%
15/09/17 17:45:23 INFO mapreduce.Job:  map 100% reduce 0%
15/09/17 17:45:59 INFO mapreduce.Job:  map 100% reduce 100%
15/09/17 17:46:03 INFO mapreduce.Job: Job job_1438799705772_0016 completed succe
ssfully
15/09/17 17:46:05 INFO mapreduce.Job: Counters: 50
        File System Counters
                FILE: Number of bytes read=340342
                FILE: Number of bytes written=913451
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=1111306
                HDFS: Number of bytes written=250778
                HDFS: Number of read operations=6
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=2
        Job Counters
                Launched map tasks=1
                Launched reduce tasks=1
                Data-local map tasks=1
                Total time spent by all maps in occupied slots (ms)=35617
                Total time spent by all reduces in occupied slots (ms)=35525
                Total time spent by all map tasks (ms)=35617
                Total time spent by all reduce tasks (ms)=35525
                Total vcore-seconds taken by all map tasks=35617
                Total vcore-seconds taken by all reduce tasks=35525
                Total megabyte-seconds taken by all map tasks=18235904
                Total megabyte-seconds taken by all reduce tasks=18188800
        Map-Reduce Framework
                Map input records=19150
                Map output records=184794
                Map output bytes=1819171
                Map output materialized bytes=340342
                Input split bytes=113
                Combine input records=184794
                Combine output records=22918
                Reduce input groups=22918
                Reduce shuffle bytes=340342
                Reduce input records=22918
                Reduce output records=22918
                Spilled Records=45836
                Shuffled Maps =1
                Failed Shuffles=0
                Merged Map outputs=1
                GC time elapsed (ms)=502
                CPU time spent (ms)=23280
                Physical memory (bytes) snapshot=521297920
                Virtual memory (bytes) snapshot=2722750464
                Total committed heap usage (bytes)=495976448
        Shuffle Errors
                BAD_ID=0
                CONNECTION=0
                IO_ERROR=0
                WRONG_LENGTH=0
                WRONG_MAP=0
                WRONG_REDUCE=0
        WordCount2$TokenizerMapper$CountersEnum
                INPUT_WORDS=184794
        File Input Format Counters
                Bytes Read=1111193
        File Output Format Counters
                Bytes Written=250778
[hdfs@rvm ~]$
```

__15. The output is going to the directory wcout. You can check what is in that directory by using commands such as:

```
hdfs dfs –ls sampledata/wcout
hdfs dfs –cat sampledata/wcout/* | more
```

A portion of your results will appear as follows:

```
MINGW32:/c/vm_images/Products/MR_YARN
"----    1
"----thus       1
"----tuus,      1
"----ut 1
"--tristesque   1
".......,       1
"_i.e._ 2
"_si    1
"_tenderly      1
"_that  1
"_they  1
"a      6
"abdiel 1
"abstain        1
"accursed       1
"achilles       1
"achilles,      1
"active,        1
"adstando       1
"after  2
"agamemnon,     1
"agile,"        1
"ah     4
"ah!    3
"ajax,  4
"alas!  21
```

__16. Press the spacebar to page down your output.

Examining this output will give you a better idea of what text patterns you wish to include in your patterns-to-skip file. Notice extraneous symbols counted as words, or part of words that you may wish to skip. For example, you may wish to skip patterns such as quotation marks, parenthetical units, and page numbers: ", (, 469, 470, 471, and so forth.

__17. Press **Ctrl+c** to break out of the piped output.

__18. If you want to run this jar file again, or another similar program, note that Hadoop expects that the output directory is empty (and it will create it, if necessary). To remove the output files and the directory itself, use a recursive remove statement:

```
hdfs dfs –rm –R sampledata/wcout
```

__19. The program that we ran with MapReduce can also be run with YARN. The library interface is the same. Use instead:

```
yarn jar ./WC2.jar WordCount2 –Dwordcount.case.sensitive=false
sampledata/Iliad.txt sampledata/ywcout –skip
sampledata/patternsToSkip.txt
```

In each case here, you should use the URL included in the listing produced by the run in the terminal window to view your job run, either while it is running, or afterwards. Also, you should review the listing produced by the run in the terminal window, as well as the output file(s) produced on the hadoop file system.

2.4   Follow up: Additional MapReduce programs to run

There are a number of additional MapReduce programs available in your VM Image, as in any distribution. You can find example programs by using the find command.

__1.   Locate the example programs by using *find*. At the prompt, type:

```
find / -name "*examples*" 2> /dev/null
```

__2.   The standard set of available sample MapReduce programs ready to run can be found. Type (as one line):

```
jar tf /usr/iop/4.1.0.0/hadoop-mapreduce/hadoop-mapreduce-examples-
2.7.1-IBM-8.jar
```

The output is not shown here.

__3.   The source code for these can be found as:

```
cd ~
mkdir java_source_files
cd java_source_files
tar xvf /usr/iop/4.1.0.0/hadoop/mapreduce.tar.gz
jar tf
~/java_source_files/hadoop/share/hadoop/mapreduce/sources/hadoop-
mapreduce-examples-2.7.1-IBM-8-sources.jar
```

The results are:
META-INF/
META-INF/MANIFEST.MF
org/
org/apache/
org/apache/hadoop/
org/apache/hadoop/examples/
org/apache/hadoop/examples/dancing/
org/apache/hadoop/examples/pi/
org/apache/hadoop/examples/pi/math/
org/apache/hadoop/examples/terasort/
org/apache/hadoop/examples/terasort/2009-write-up/
org/apache/hadoop/examples/WordMedian.java
org/apache/hadoop/examples/RandomWriter.java
org/apache/hadoop/examples/AggregateWordHistogram.java

org/apache/hadoop/examples/ExampleDriver.java
org/apache/hadoop/examples/package.html
org/apache/hadoop/examples/BaileyBorweinPlouffe.java
org/apache/hadoop/examples/dancing/DancingLinks.java
org/apache/hadoop/examples/dancing/Pentomino.java
org/apache/hadoop/examples/dancing/package.html
org/apache/hadoop/examples/dancing/OneSidedPentomino.java
org/apache/hadoop/examples/dancing/Sudoku.java
org/apache/hadoop/examples/dancing/DistributedPentomino.java
org/apache/hadoop/examples/dancing/puzzle1.dta
org/apache/hadoop/examples/Grep.java
org/apache/hadoop/examples/WordCount.java
org/apache/hadoop/examples/SecondarySort.java
org/apache/hadoop/examples/DBCountPageView.java
org/apache/hadoop/examples/pi/Combinable.java
org/apache/hadoop/examples/pi/TaskResult.java
org/apache/hadoop/examples/pi/SummationWritable.java
org/apache/hadoop/examples/pi/package.html
org/apache/hadoop/examples/pi/DistBbp.java
org/apache/hadoop/examples/pi/Parser.java
org/apache/hadoop/examples/pi/DistSum.java
org/apache/hadoop/examples/pi/Util.java
org/apache/hadoop/examples/pi/math/LongLong.java
org/apache/hadoop/examples/pi/math/Summation.java
org/apache/hadoop/examples/pi/math/package.html
org/apache/hadoop/examples/pi/math/Montgomery.java
org/apache/hadoop/examples/pi/math/ArithmeticProgression.java
org/apache/hadoop/examples/pi/math/Bellard.java
org/apache/hadoop/examples/pi/math/Modular.java
org/apache/hadoop/examples/pi/Container.java
org/apache/hadoop/examples/Sort.java
org/apache/hadoop/examples/QuasiMonteCarlo.java
org/apache/hadoop/examples/WordMean.java
org/apache/hadoop/examples/terasort/TeraGen.java
org/apache/hadoop/examples/terasort/job_history_summary.py
org/apache/hadoop/examples/terasort/package.html
org/apache/hadoop/examples/terasort/TeraOutputFormat.java
org/apache/hadoop/examples/terasort/TeraChecksum.java
org/apache/hadoop/examples/terasort/2009-write-up/500GBTaskTime.png
org/apache/hadoop/examples/terasort/2009-write-up/tera.bib
org/apache/hadoop/examples/terasort/2009-write-up/1TBTaskTime.png
org/apache/hadoop/examples/terasort/2009-write-up/1PBTaskTime.png
org/apache/hadoop/examples/terasort/2009-write-up/Yahoo2009.tex
org/apache/hadoop/examples/terasort/2009-write-up/.gitignore
org/apache/hadoop/examples/terasort/2009-write-up/100TBTaskTime.png
org/apache/hadoop/examples/terasort/TeraInputFormat.java

> **org/apache/hadoop/examples/terasort/TeraScheduler.java**
> **org/apache/hadoop/examples/terasort/Random16.java**
> **org/apache/hadoop/examples/terasort/TeraValidate.java**
> **org/apache/hadoop/examples/terasort/GenSort.java**
> **org/apache/hadoop/examples/terasort/TeraSort.java**
> **org/apache/hadoop/examples/terasort/Unsigned16.java**
> **org/apache/hadoop/examples/RandomTextWriter.java**
> **org/apache/hadoop/examples/WordStandardDeviation.java**
> **org/apache/hadoop/examples/Join.java**
> **org/apache/hadoop/examples/AggregateWordCount.java**
> **org/apache/hadoop/examples/MultiFileWordCount.java**

Next you want to extract the source files into a directory that you create.

__4. Navigate to your directory where you will operate.

```
cd ~
```

__5. Extract the source files into a directory that you create (java-examples). Type:

```
mkdir java-examples
```

```
cd java-examples
```

(Type the following statement as one line)

```
jar xf
~/java_source_files/hadoop/share/hadoop/mapreduce/sources/hadoop-
mapreduce-examples-2.7.1-IBM-8-sources.jar
```

```
ls -l
```

```
[wongk@iop-bi-master ~]$ mkdir java-examples
[wongk@iop-bi-master ~]$ cd java-examples
[wongk@iop-bi-master java-examples]$ jar xf /usr/iop/4.1.0.0/hadoop-mapreduce/ha
doop-mapreduce-examples-2.7.1-IBM-8.jar
[wongk@iop-bi-master java-examples]$ ls -l
total 8
drwx------ 3 wongk wongk 4096 Aug 21  2015 META-INF
drwx------ 3 wongk wongk 4096 Aug 21  2015 org
[wongk@iop-bi-master java-examples]$
```

__6. List the filenames. Type:

```
ls -l -R * | more
```

```
[hdfs@rvm java-examples]$ ls -l -R * | more
META-INF:
total 4
-rw-r--r-- 1 hdfs hadoop 126 Mar 27 21:41 MANIFEST.MF

org:
total 4
drwxr-xr-x 3 hdfs hadoop 4096 Mar 27 21:33 apache

org/apache:
total 4
drwxr-xr-x 3 hdfs hadoop 4096 Mar 27 21:33 hadoop

org/apache/hadoop:
total 4
drwxr-xr-x 5 hdfs hadoop 4096 Mar 27 21:33 examples

org/apache/hadoop/examples:
total 204
-rw-r--r-- 1 hdfs hadoop  2897 Mar 27 21:33 AggregateWordCount.java
-rw-r--r-- 1 hdfs hadoop  3016 Mar 27 21:33 AggregateWordHistogram.java
-rw-r--r-- 1 hdfs hadoop 21254 Mar 27 21:33 BaileyBorweinPlouffe.java
drwxr-xr-x 2 hdfs hadoop  4096 Mar 27 21:33 dancing
-rw-r--r-- 1 hdfs hadoop 13495 Mar 27 21:33 DBCountPageView.java
-rw-r--r-- 1 hdfs hadoop  4301 Mar 27 21:33 ExampleDriver.java
-rw-r--r-- 1 hdfs hadoop  3730 Mar 27 21:33 Grep.java
-rw-r--r-- 1 hdfs hadoop  7033 Mar 27 21:33 Join.java
-rw-r--r-- 1 hdfs hadoop  8111 Mar 27 21:33 MultiFileWordCount.java
-rw-r--r-- 1 hdfs hadoop   853 Mar 27 21:33 package.html
drwxr-xr-x 3 hdfs hadoop  4096 Mar 27 21:33 pi
-rw-r--r-- 1 hdfs hadoop 12628 Mar 27 21:33 QuasiMonteCarlo.java
-rw-r--r-- 1 hdfs hadoop 40575 Mar 27 21:33 RandomTextWriter.java
-rw-r--r-- 1 hdfs hadoop 10573 Mar 27 21:33 RandomWriter.java
-rw-r--r-- 1 hdfs hadoop  7809 Mar 27 21:33 SecondarySort.java
-rw-r--r-- 1 hdfs hadoop  8167 Mar 27 21:33 Sort.java
drwxr-xr-x 3 hdfs hadoop  4096 Mar 27 21:33 terasort
-rw-r--r-- 1 hdfs hadoop  3297 Mar 27 21:33 WordCount.java
-rw-r--r-- 1 hdfs hadoop  6327 Mar 27 21:33 WordMean.java
-rw-r--r-- 1 hdfs hadoop  7084 Mar 27 21:33 WordMedian.java
-rw-r--r-- 1 hdfs hadoop  7253 Mar 27 21:33 WordStandardDeviation.java

org/apache/hadoop/examples/dancing:
total 68
-rw-r--r-- 1 hdfs hadoop 12675 Mar 27 21:33 DancingLinks.java
-rw-r--r-- 1 hdfs hadoop  8677 Mar 27 21:33 DistributedPentomino.java
-rw-r--r-- 1 hdfs hadoop  2921 Mar 27 21:33 OneSidedPentomino.java
-rw-r--r-- 1 hdfs hadoop  2806 Mar 27 21:33 package.html
-rw-r--r-- 1 hdfs hadoop 14342 Mar 27 21:33 Pentomino.java
-rw-r--r-- 1 hdfs hadoop   162 Mar 27 21:33 puzzle1.dta
-rw-r--r-- 1 hdfs hadoop  9369 Mar 27 21:33 Sudoku.java

org/apache/hadoop/examples/pi:
total 88
-rw-r--r-- 1 hdfs hadoop  1155 Mar 27 21:33 Combinable.java
-rw-r--r-- 1 hdfs hadoop  1047 Mar 27 21:33 Container.java
-rw-r--r-- 1 hdfs hadoop  6423 Mar 27 21:33 DistBbp.java
-rw-r--r-- 1 hdfs hadoop 22234 Mar 27 21:33 DistSum.java
drwxr-xr-x 2 hdfs hadoop  4096 Mar 27 21:33 math
-rw-r--r-- 1 hdfs hadoop  8210 Mar 27 21:33 package.html
-rw-r--r-- 1 hdfs hadoop  6531 Mar 27 21:33 Parser.java
-rw-r--r-- 1 hdfs hadoop  4343 Mar 27 21:33 SummationWritable.java
-rw-r--r-- 1 hdfs hadoop  3480 Mar 27 21:33 TaskResult.java
-rw-r--r-- 1 hdfs hadoop 11343 Mar 27 21:33 Util.java

org/apache/hadoop/examples/pi/math:
total 44
-rw-r--r-- 1 hdfs hadoop  4250 Mar 27 21:33 ArithmeticProgression.java
-rw-r--r-- 1 hdfs hadoop 10282 Mar 27 21:33 Bellard.java
-rw-r--r-- 1 hdfs hadoop  3047 Mar 27 21:33 LongLong.java
-rw-r--r-- 1 hdfs hadoop  2908 Mar 27 21:33 Modular.java
-rw-r--r-- 1 hdfs hadoop  2496 Mar 27 21:33 Montgomery.java
-rw-r--r-- 1 hdfs hadoop   917 Mar 27 21:33 package.html
-rw-r--r-- 1 hdfs hadoop  8094 Mar 27 21:33 Summation.java

org/apache/hadoop/examples/terasort:
total 120
drwxr-xr-x 2 hdfs hadoop  4096 Mar 27 21:33 2009-write-up
-rw-r--r-- 1 hdfs hadoop  8479 Mar 27 21:33 GenSort.java
-rw-r--r-- 1 hdfs hadoop  3444 Mar 27 21:33 job_history_summary.py
-rw-r--r-- 1 hdfs hadoop  4514 Mar 27 21:33 package.html
-rw-r--r-- 1 hdfs hadoop 22566 Mar 27 21:33 Random16.java
-rw-r--r-- 1 hdfs hadoop  3644 Mar 27 21:33 TeraChecksum.java
-rw-r--r-- 1 hdfs hadoop  9960 Mar 27 21:33 TeraGen.java
-rw-r--r-- 1 hdfs hadoop 10729 Mar 27 21:33 TeraInputFormat.java
-rw-r--r-- 1 hdfs hadoop  4001 Mar 27 21:33 TeraOutputFormat.java
-rw-r--r-- 1 hdfs hadoop  8097 Mar 27 21:33 TeraScheduler.java
-rw-r--r-- 1 hdfs hadoop 11048 Mar 27 21:33 TeraSort.java
-rw-r--r-- 1 hdfs hadoop  6787 Mar 27 21:33 TeraValidate.java
-rw-r--r-- 1 hdfs hadoop  7526 Mar 27 21:33 Unsigned16.java

org/apache/hadoop/examples/terasort/2009-write-up:
total 608
-rw-r--r-- 1 hdfs hadoop 138496 Mar 27 21:33 100TBTaskTime.png
-rw-r--r-- 1 hdfs hadoop 186897 Mar 27 21:33 1PBTaskTime.png
-rw-r--r-- 1 hdfs hadoop 136810 Mar 27 21:33 1TBTaskTime.png
-rw-r--r-- 1 hdfs hadoop 128816 Mar 27 21:33 500GBTaskTime.png
-rw-r--r-- 1 hdfs hadoop   1371 Mar 27 21:33 tera.bib
-rw-r--r-- 1 hdfs hadoop  17951 Mar 27 21:33 Yahoo2009.tex
[hdfs@rvm java-examples]$
```

Next, you want to find out what these sample programs do.

__7. To find out what the sample programs do, type as one line:

```
yarn jar /usr/iop/4.1.0.0/hadoop-mapreduce/hadoop-mapreduce-examples.jar
```

```
[hdfs@rvm java-examples]$ yarn jar /usr/iop/4.0.0.0/hadoop-mapr
educe-examples.jar
An example program must be given as the first argument.
Valid program names are:
  aggregatewordcount: An Aggregate based map/reduce program that counts the word
s in the input files.
  aggregatewordhist: An Aggregate based map/reduce program that computes the his
togram of the words in the input files.
  bbp: A map/reduce program that uses Bailey-Borwein-Plouffe to compute exact di
gits of Pi.
  dbcount: An example job that count the pageview counts from a database.
  distbbp: A map/reduce program that uses a BBP-type formula to compute exact bi
ts of Pi.
  grep: A map/reduce program that counts the matches of a regex in the input.
  join: A job that effects a join over sorted, equally partitioned datasets
  multifilewc: A job that counts words from several files.
  pentomino: A map/reduce tile laying program to find solutions to pentomino pro
blems.
  pi: A map/reduce program that estimates Pi using a quasi-Monte Carlo method.
  randomtextwriter: A map/reduce program that writes 10GB of random textual data
 per node.
  randomwriter: A map/reduce program that writes 10GB of random data per node.
  secondarysort: An example defining a secondary sort to the reduce.
  sort: A map/reduce program that sorts the data written by the random writer.
  sudoku: A sudoku solver.
  teragen: Generate data for the terasort
  terasort: Run the terasort
  teravalidate: Checking results of terasort
  wordcount: A map/reduce program that counts the words in the input files.
  wordmean: A map/reduce program that counts the average length of the words in
the input files.
  wordmedian: A map/reduce program that counts the median length of the words in
 the input files.
  wordstandarddeviation: A map/reduce program that counts the standard deviation
 of the length of the words in the input files.
[hdfs@rvm java-examples]$
```

Source code for other examples is also available on the internet at:

- http://svn.apache.org/viewvc/hadoop/common/trunk/hadoop-mapreduce-project/hadoop-mapreduce-examples/src/main/java/org/apache/hadoop/examples/.

This location also provides some explanation on how to run, etc.

There are quite a few very good explanations of MapReduce available on the internet, including:

- https://highlyscalable.wordpress.com/2012/02/01/mapreduce-patterns/

End of exercise

NOTES

NOTES

IBM

IBM Software