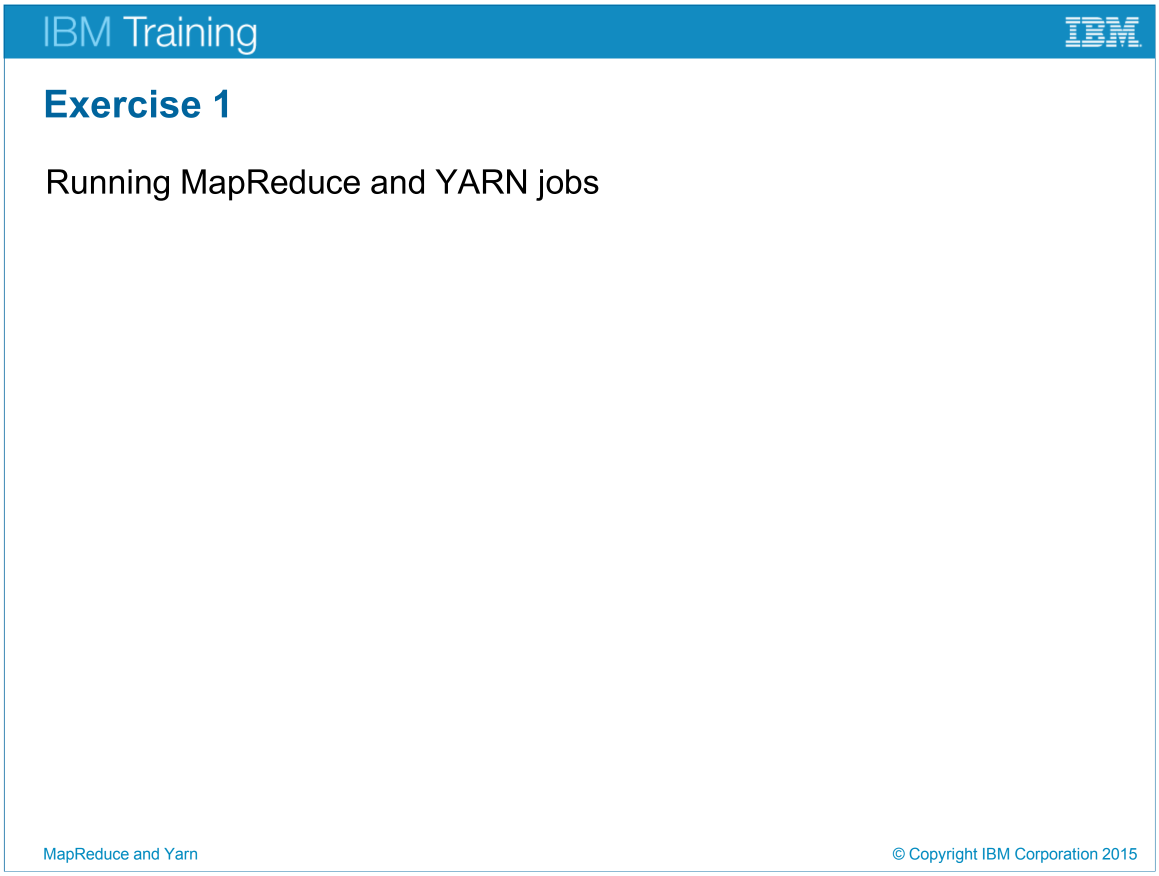


IBM Training

Exercise 1

Running MapReduce and YARN jobs

MapReduce and Yarn

© Copyright IBM Corporation 2015

Exercise 1: Running MapReduce and YARN jobs

Exercise 1: Running MapReduce and YARN jobs

Purpose:
You will run Java programs using Hadoop v2, YARN, and related technologies. You will not need to do any Java programming. The Hadoop community provides a number of standard example programs (akin to "Hello, World" for people learning to write C language programs).

In the real world, the sample/example programs provide opportunities to learn the relevant technology, and on a newly installed Hadoop cluster, to exercise the system and the operations environment.

You will start with an existing sample program, selected from among those available, and progress towards compiling a program a Java program from scratch and learning in what type of environment it runs.

The Java program that you will be compiling and running, WordCount2.java, resides in the Linux directory /home/labfiles.

VM Hostname:	**http://ibmclass.localdomain**
User/Password:	**biadmin / biadmin**
	root/dalvm3

Task 1. Run a simple MapReduce job from supplied example programs available with a Hadoop default installation.

The best references for writing MapReduce programs for MRv1, MRv2, and YARN are on the Hadoop website. The URLs relevant for Hadoop r2.7.0 are:
http://hadoop.apache.org/docs/current/hadoop-mapreduce-client/hadoop-mapreduce-client-core/MapReduceTutorial.html
http://hadoop.apache.org/docs/r2.7.0/hadoop-yarn/hadoop-yarn-site/WritingYarnApplications.html

Use the documentation that corresponds to your current release of Hadoop and HDFS (substitute the appropriate value for r2.7.0, if different).

1. Connect to and login to your lab environment with user **biadmin** and password **biadmin** credentials.
2. In a new terminal window, type `cd` to change to your home directory.

3. To locate all of the Java jar files in your system with **example** in the file name, type **`find / -name "*example*" 2> /dev/null | grep jar`**.

   There will be files and directories in your Linux file system that you, as biadmin user, will not have access to. The find command will give you errors. Those errors are sent to Linux errout and are eliminated from your console output by using: **`2> /dev/null`**

```
[hdfs@ibmclass ~]$ find / -name "*example*" 2> /dev/null | grep jar
/usr/share/java/rhino-examples-1.7.jar
/usr/share/java/rhino-examples.jar
/usr/iop/4.0.0.0/oozie/doc/oozie-4.1.0_IBM_5/examples/apps/java-main/lib/oozie-
examples-4.1.0_IBM_5.jar
/usr/iop/4.0.0.0/oozie/doc/oozie-4.1.0_IBM_5/examples/apps/datelist-java-
main/lib/oozie-examples-4.1.0_IBM_5.jar
/usr/iop/4.0.0.0/oozie/doc/oozie-4.1.0_IBM_5/examples/apps/hadoop-el/lib/oozie-
examples-4.1.0_IBM_5.jar
/usr/iop/4.0.0.0/oozie/doc/oozie-4.1.0_IBM_5/examples/apps/custom-main/lib/oozie-
examples-4.1.0_IBM_5.jar
/usr/iop/4.0.0.0/oozie/doc/oozie-4.1.0_IBM_5/examples/apps/demo/lib/oozie-examples-
4.1.0_IBM_5.jar
/usr/iop/4.0.0.0/oozie/doc/oozie-4.1.0_IBM_5/examples/apps/map-reduce/lib/oozie-
examples-4.1.0_IBM_5.jar
/usr/iop/4.0.0.0/oozie/doc/oozie-4.1.0_IBM_5/examples/apps/aggregator/lib/oozie-
examples-4.1.0_IBM_5.jar
/usr/iop/4.0.0.0/oozie/share/lib/hbase/hbase-examples-0.98.8_IBM_4-hadoop2.jar
/usr/iop/4.0.0.0/knox/samples/hadoop-examples.jar
/usr/iop/4.0.0.0/hadoop-mapreduce/hadoop-mapreduce-examples.jar
/usr/iop/4.0.0.0/hadoop-mapreduce/hadoop-mapreduce-examples-2.6.0-IBM-7.jar
/usr/iop/4.0.0.0/spark/lib/spark-examples.jar
/usr/iop/4.0.0.0/spark/lib/spark-examples-1.2.1_IBM_4-hadoop2.6.0-IBM-7.jar
/usr/iop/4.0.0.0/hbase/lib/hbase-examples.jar
/usr/iop/4.0.0.0/hbase/lib/hbase-examples-0.98.8_IBM_4-hadoop2.jar
[hdfs@ibmclass ~]$
```

   You can list the contents of a jar file using **`jar tf`** (t = table of contents, f = file name follows); you will apply that to one of the jar files that you found with your find-search.

4. To list the contents of the **hadoop-mapreduce-examples.jar** file, type **`jar tf /usr/iop/4.0.0.0/hadoop-mapreduce/hadoop-mapreduce-examples.jar`**.

   The results will show something similar to:

```
[hdfs@ibmclass ~]$ jar -tf /usr/iop/4.0.0.0/hadoop-mapreduce/hadoop-mapreduce-
examples.jar
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/RandomWriter$RandomInputFormat$RandomRecordReader.class
org/apache/hadoop/examples/DBCountPageView$PageviewReducer.class
org/apache/hadoop/examples/DBCountPageView$PageviewRecord.class
org/apache/hadoop/examples/SecondarySort$FirstGroupingComparator.class
org/apache/hadoop/examples/MultiFileWordCount.class
org/apache/hadoop/examples/QuasiMonteCarlo.class
```

```
org/apache/hadoop/examples/BaileyBorweinPlouffe$BbpInputFormat.class
org/apache/hadoop/examples/SecondarySort$IntPair.class
org/apache/hadoop/examples/SecondarySort$Reduce.class
org/apache/hadoop/examples/BaileyBorweinPlouffe$BbpInputFormat$1.class
org/apache/hadoop/examples/BaileyBorweinPlouffe$Fraction.class
org/apache/hadoop/examples/DBCountPageView.class
org/apache/hadoop/examples/WordMean$WordMeanMapper.class
org/apache/hadoop/examples/SecondarySort.class
org/apache/hadoop/examples/RandomTextWriter.class
org/apache/hadoop/examples/QuasiMonteCarlo$HaltonSequence.class
org/apache/hadoop/examples/RandomWriter$RandomInputFormat.class
org/apache/hadoop/examples/Sort.class
org/apache/hadoop/examples/WordStandardDeviation.class
org/apache/hadoop/examples/BaileyBorweinPlouffe$1.class
org/apache/hadoop/examples/dancing/Sudoku$ColumnConstraint.class
org/apache/hadoop/examples/dancing/OneSidedPentomino.class
org/apache/hadoop/examples/dancing/Sudoku$SolutionPrinter.class
org/apache/hadoop/examples/dancing/Pentomino$Point.class
org/apache/hadoop/examples/dancing/Sudoku$RowConstraint.class
org/apache/hadoop/examples/dancing/DistributedPentomino$PentMap.class
org/apache/hadoop/examples/dancing/DistributedPentomino$PentMap$SolutionCatcher.class
org/apache/hadoop/examples/dancing/DancingLinks$Node.class
org/apache/hadoop/examples/dancing/Sudoku$CellConstraint.class
org/apache/hadoop/examples/dancing/DancingLinks$ColumnHeader.class
org/apache/hadoop/examples/dancing/Pentomino$ColumnName.class
org/apache/hadoop/examples/dancing/DancingLinks$SolutionAcceptor.class
org/apache/hadoop/examples/dancing/Pentomino.class
org/apache/hadoop/examples/dancing/DancingLinks.class
org/apache/hadoop/examples/dancing/Sudoku.class
org/apache/hadoop/examples/dancing/Pentomino$SolutionPrinter.class
org/apache/hadoop/examples/dancing/DistributedPentomino.class
org/apache/hadoop/examples/dancing/Sudoku$SquareConstraint.class
org/apache/hadoop/examples/dancing/Pentomino$SolutionCategory.class
org/apache/hadoop/examples/dancing/Sudoku$ColumnName.class
org/apache/hadoop/examples/dancing/Pentomino$Piece.class
org/apache/hadoop/examples/AggregateWordHistogram.class
org/apache/hadoop/examples/DBCountPageView$PageviewMapper.class
org/apache/hadoop/examples/WordMedian$WordMedianReducer.class
org/apache/hadoop/examples/RandomTextWriter$RandomTextMapper.class
org/apache/hadoop/examples/BaileyBorweinPlouffe.class
org/apache/hadoop/examples/WordMean$WordMeanReducer.class
org/apache/hadoop/examples/RandomWriter$Counters.class
org/apache/hadoop/examples/BaileyBorweinPlouffe$BbpSplit.class
org/apache/hadoop/examples/BaileyBorweinPlouffe$BbpReducer$1.class
org/apache/hadoop/examples/WordCount.class
org/apache/hadoop/examples/Join.class
org/apache/hadoop/examples/SecondarySort$IntPair$Comparator.class
org/apache/hadoop/examples/MultiFileWordCount$MapClass.class
org/apache/hadoop/examples/AggregateWordCount$WordCountPlugInClass.class
org/apache/hadoop/examples/WordStandardDeviation$WordStandardDeviationReducer.class
org/apache/hadoop/examples/RandomTextWriter$Counters.class
org/apache/hadoop/examples/MultiFileWordCount$CombineFileLineRecordReader.class
org/apache/hadoop/examples/WordCount$TokenizerMapper.class
org/apache/hadoop/examples/BaileyBorweinPlouffe$BbpMapper.class
org/apache/hadoop/examples/QuasiMonteCarlo$QmcReducer.class
org/apache/hadoop/examples/MultiFileWordCount$WordOffset.class
org/apache/hadoop/examples/pi/DistSum$Machine$AbstractInputFormat.class
org/apache/hadoop/examples/pi/DistSum$MapSide$PartitionInputFormat.class
org/apache/hadoop/examples/pi/DistSum$ReduceSide$SummationInputFormat.class
org/apache/hadoop/examples/pi/Parser.class
org/apache/hadoop/examples/pi/DistSum.class
org/apache/hadoop/examples/pi/DistSum$Machine.class
org/apache/hadoop/examples/pi/SummationWritable.class
org/apache/hadoop/examples/pi/Util$Timer.class
org/apache/hadoop/examples/pi/DistSum$Parameters.class
org/apache/hadoop/examples/pi/DistSum$ReduceSide$PartitionMapper.class
```

```
org/apache/hadoop/examples/pi/DistBbp.class
org/apache/hadoop/examples/pi/DistSum$1.class
org/apache/hadoop/examples/pi/DistSum$ReduceSide$SummingReducer.class
org/apache/hadoop/examples/pi/DistSum$Machine$AbstractInputFormat$1.class
org/apache/hadoop/examples/pi/DistSum$MapSide$SummingMapper.class
org/apache/hadoop/examples/pi/Combinable.class
org/apache/hadoop/examples/pi/Container.class
org/apache/hadoop/examples/pi/DistSum$Machine$SummationSplit.class
org/apache/hadoop/examples/pi/DistSum$MapSide.class
org/apache/hadoop/examples/pi/DistSum$MixMachine.class
org/apache/hadoop/examples/pi/Util.class
org/apache/hadoop/examples/pi/DistSum$Computation.class
org/apache/hadoop/examples/pi/DistSum$ReduceSide$IndexPartitioner.class
org/apache/hadoop/examples/pi/math/Bellard$Sum$Tail.class
org/apache/hadoop/examples/pi/math/Bellard$Parameter.class
org/apache/hadoop/examples/pi/math/ArithmeticProgression.class
org/apache/hadoop/examples/pi/math/Modular.class
org/apache/hadoop/examples/pi/math/LongLong.class
org/apache/hadoop/examples/pi/math/Summation.class
org/apache/hadoop/examples/pi/math/Montgomery.class
org/apache/hadoop/examples/pi/math/Bellard$Sum$1.class
org/apache/hadoop/examples/pi/math/Bellard$Sum.class
org/apache/hadoop/examples/pi/math/Montgomery$Product.class
org/apache/hadoop/examples/pi/math/Bellard$1.class
org/apache/hadoop/examples/pi/math/Bellard.class
org/apache/hadoop/examples/pi/DistSum$ReduceSide.class
org/apache/hadoop/examples/pi/SummationWritable$ArithmeticProgressionWritable.class
org/apache/hadoop/examples/pi/TaskResult.class
org/apache/hadoop/examples/RandomWriter$RandomMapper.class
org/apache/hadoop/examples/terasort/TeraInputFormat$TeraRecordReader.class
org/apache/hadoop/examples/terasort/TeraChecksum.class
org/apache/hadoop/examples/terasort/TeraOutputFormat$TeraRecordWriter.class
org/apache/hadoop/examples/terasort/TeraValidate$ValidateReducer.class
org/apache/hadoop/examples/terasort/GenSort.class
org/apache/hadoop/examples/terasort/Random16$RandomConstant.class
org/apache/hadoop/examples/terasort/TeraGen$SortGenMapper.class
org/apache/hadoop/examples/terasort/TeraInputFormat$1.class
org/apache/hadoop/examples/terasort/TeraInputFormat$SamplerThreadGroup.class
org/apache/hadoop/examples/terasort/TeraGen$RangeInputFormat$RangeRecordReader.class
org/apache/hadoop/examples/terasort/TeraChecksum$ChecksumReducer.class
org/apache/hadoop/examples/terasort/TeraSort.class
org/apache/hadoop/examples/terasort/Random16.class
org/apache/hadoop/examples/terasort/TeraSort$TotalOrderPartitioner$InnerTrieNode.class
org/apache/hadoop/examples/terasort/TeraSort$SimplePartitioner.class
org/apache/hadoop/examples/terasort/TeraGen.class
org/apache/hadoop/examples/terasort/TeraInputFormat$TextSampler.class
org/apache/hadoop/examples/terasort/TeraGen$Counters.class
org/apache/hadoop/examples/terasort/TeraSort$TotalOrderPartitioner.class
org/apache/hadoop/examples/terasort/TeraOutputFormat.class
org/apache/hadoop/examples/terasort/TeraScheduler$Split.class
org/apache/hadoop/examples/terasort/TeraSort$TotalOrderPartitioner$TrieNode.class
org/apache/hadoop/examples/terasort/TeraValidate$ValidateMapper.class
org/apache/hadoop/examples/terasort/TeraSort$TotalOrderPartitioner$LeafTrieNode.class
org/apache/hadoop/examples/terasort/TeraGen$RangeInputFormat.class
org/apache/hadoop/examples/terasort/TeraScheduler.class
org/apache/hadoop/examples/terasort/TeraChecksum$ChecksumMapper.class
org/apache/hadoop/examples/terasort/TeraScheduler$Host.class
org/apache/hadoop/examples/terasort/TeraValidate.class
org/apache/hadoop/examples/terasort/Unsigned16.class
org/apache/hadoop/examples/terasort/TeraInputFormat.class
org/apache/hadoop/examples/terasort/TeraGen$RangeInputFormat$RangeInputSplit.class
org/apache/hadoop/examples/AggregateWordHistogram$AggregateWordHistogramPlugin.class
org/apache/hadoop/examples/AggregateWordCount.class
org/apache/hadoop/examples/Grep.class
org/apache/hadoop/examples/WordCount$IntSumReducer.class
org/apache/hadoop/examples/WordMean.class
```

```
org/apache/hadoop/examples/WordStandardDeviation$WordStandardDeviationMapper.class
org/apache/hadoop/examples/SecondarySort$MapClass.class
org/apache/hadoop/examples/BaileyBorweinPlouffe$BbpReducer.class
org/apache/hadoop/examples/RandomWriter.class
org/apache/hadoop/examples/SecondarySort$FirstPartitioner.class
org/apache/hadoop/examples/ExampleDriver.class
org/apache/hadoop/examples/WordMedian$WordMedianMapper.class
org/apache/hadoop/examples/MultiFileWordCount$MyInputFormat.class
org/apache/hadoop/examples/QuasiMonteCarlo$QmcMapper.class
org/apache/hadoop/examples/DBCountPageView$AccessRecord.class
org/apache/hadoop/examples/WordMedian.class
META-INF/maven/
META-INF/maven/org.apache.hadoop/
META-INF/maven/org.apache.hadoop/hadoop-mapreduce-examples/
META-INF/maven/org.apache.hadoop/hadoop-mapreduce-examples/pom.xml
META-INF/maven/org.apache.hadoop/hadoop-mapreduce-examples/pom.properties
[hdfs@ibmclass ~]$
```

This is a long list. You are going to use the WordCount example from this file, but what is the file called? Note also that you can get dates and times for the various classes and programs in this jar file by using **`jar tvf`** (v = verbose).

We can find what executable Java programs are included in the manifest by attempting to run this jar with **`hadoop jar`**, but not providing any additional and perhaps necessary parameters.

5. Type **`hadoop jar /usr/iop/4.0.0.0/hadoop-mapreduce/hadoop-mapreduce-examples.jar`**.

```
[hdfs@ibmclass ~]$ hadoop jar /usr/iop/4.0.0.0/hadoop-mapreduce/hadoop-mapreduce-
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 words in the
input files.
  aggregatewordhist: An Aggregate based map/reduce program that computes the histogram
of the words in the input files.
  bbp: A map/reduce program that uses Bailey-Borwein-Plouffe to compute exact digits 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 bits 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 problems.
  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@ibmclass ~]$
```

The program that you will use is wordcount which is a map/reduce program that counts the words in the input files.

6. To run the program with appropriate parameters, type **`hadoop jar /usr/iop/4.0.0.0/hadoop-mapreduce/hadoop-mapreduce-examples.jar wordcount Gutenberg/Frankenstein.txt wcout`**.

```
[biadmin@ibmclass ~]$ hadoop jar /usr/iop/4.0.0.0/hadoop-mapreduce/hadoop-mapreduce-
examples.jar wordcount Gutenberg/Frankenstein.txt wcout

15/06/04 14:48:45 INFO impl.TimelineClientImpl: Timeline service address:
http://ibmclass.localdomain:8188/ws/v1/timeline/
15/06/04 14:48:45 INFO client.RMProxy: Connecting to ResourceManager at
ibmclass.localdomain/192.168.244.140:8050
15/06/04 14:48:46 INFO input.FileInputFormat: Total input paths to process : 1
15/06/04 14:48:46 INFO mapreduce.JobSubmitter: number of splits:1
15/06/04 14:48:47 INFO mapreduce.JobSubmitter: Submitting tokens for job:
job_1433282779965_0002
15/06/04 14:48:48 INFO impl.YarnClientImpl: Submitted application
application_1433282779965_0002
15/06/04 14:48:48 INFO mapreduce.Job: The url to track the job:
http://ibmclass.localdomain:8088/proxy/application_1433282779965_0002/
15/06/04 14:48:48 INFO mapreduce.Job: Running job: job_1433282779965_0002
15/06/04 14:49:21 INFO mapreduce.Job: Job job_1433282779965_0002 running in uber mode
: false
15/06/04 14:49:21 INFO mapreduce.Job:  map 0% reduce 0%
15/06/04 14:49:57 INFO mapreduce.Job:  map 100% reduce 0%
15/06/04 14:50:11 INFO mapreduce.Job:  map 100% reduce 100%
15/06/04 14:50:12 INFO mapreduce.Job: Job job_1433282779965_0002 completed
successfully
15/06/04 14:50:12 INFO mapreduce.Job: Counters: 49
   File System Counters
          FILE: Number of bytes read=167616
          FILE: Number of bytes written=564573
          FILE: Number of read operations=0
          FILE: Number of large read operations=0
          FILE: Number of write operations=0
          HDFS: Number of bytes read=421641
          HDFS: Number of bytes written=122090
          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)=33447
          Total time spent by all reduces in occupied slots (ms)=11038
          Total time spent by all map tasks (ms)=33447
          Total time spent by all reduce tasks (ms)=11038
          Total vcore-seconds taken by all map tasks=33447
          Total vcore-seconds taken by all reduce tasks=11038
          Total megabyte-seconds taken by all map tasks=34249728
          Total megabyte-seconds taken by all reduce tasks=11302912
   Map-Reduce Framework
          Map input records=7244
          Map output records=74952
          Map output bytes=717818
          Map output materialized bytes=167616
          Input split bytes=137
          Combine input records=74952
          Combine output records=11603
          Reduce input groups=11603
          Reduce shuffle bytes=167616
```

```
                Reduce input records=11603
                Reduce output records=11603
                Spilled Records=23206
                Shuffled Maps =1
                Failed Shuffles=0
                Merged Map outputs=1
                GC time elapsed (ms)=291
                CPU time spent (ms)=5950
                Physical memory (bytes) snapshot=799457280
                Virtual memory (bytes) snapshot=3353075712
                Total committed heap usage (bytes)=859832320
        Shuffle Errors
                BAD_ID=0
                CONNECTION=0
                IO_ERROR=0
                WRONG_LENGTH=0
                WRONG_MAP=0
                WRONG_REDUCE=0
        File Input Format Counters
                Bytes Read=421504
        File Output Format Counters
                Bytes Written=122090
[biadmin@ibmclass ~]$
```

There was just one Reduce task:

```
                Launched map tasks=1
                Launched reduce tasks=1
```

and hence there would be only one output file.

7. To find what file(s) were produced, type **`hadoop fs -ls -R`**.

```
[biadmin@ibmclass ~]$ hadoop fs -ls -R
drwx---   - biadmin          0 2015-06-04 14:50 .staging
drwxr-xr-x   - biadmin biadmin          0 2015-06-04 11:01 Gutenberg
-rw-r-r-   3 biadmin biadmin     421504 2015-06-04 11:01 Gutenberg/Frankenstein.txt
-rw-r-r-   3 biadmin biadmin     697802 2015-06-04 11:01
Gutenberg/Pride_and_Prejudice.txt
-rw-r-r-   3 biadmin biadmin     757223 2015-06-04 11:01
Gutenberg/Tale_of_Two_Cities.txt
-rw-r-r-   3 biadmin biadmin     281398 2015-06-04 11:01 Gutenberg/The_Prince.txt
drwxr-xr-x   - biadmin biadmin          0 2015-06-04 14:50 wcout
-rw-r-r-   3 biadmin biadmin          0 2015-06-04 14:50 wcout/_SUCCESS
-rw-r-r-   3 biadmin biadmin     122090 2015-06-04 14:50 wcout/part-r-00000
[biadmin@ibmclass ~]$
```

The directory **`wcout`** (**`/user/biadmin/wcout`**) in HDFS was created and there are two files in it: **`_SUCCESS`** and **`part-r-00000`**.

8. To review the **part-r-0000** file, type **`hadoop fs -cat wcout/part-r-00000 | more`**.

   Scroll through the part-r-0000 file and look at least a dozen or more pages of output into more (press Enter to see more than one page of output).

   Here is a sample from several pages down:

```
Concerning      1
Constantinople  1
Constantinople, 1
Continue  1
Continuing      1
Copet.    1
Cornelius 6
Could     6
Coupar,   1
Cousin,   1
Covered   1
Creator.  1
Creator;  1
Cumberland      3
```

   While the job was running, you could have watched its progress by using the suggest URL. For the job listing above, this was:

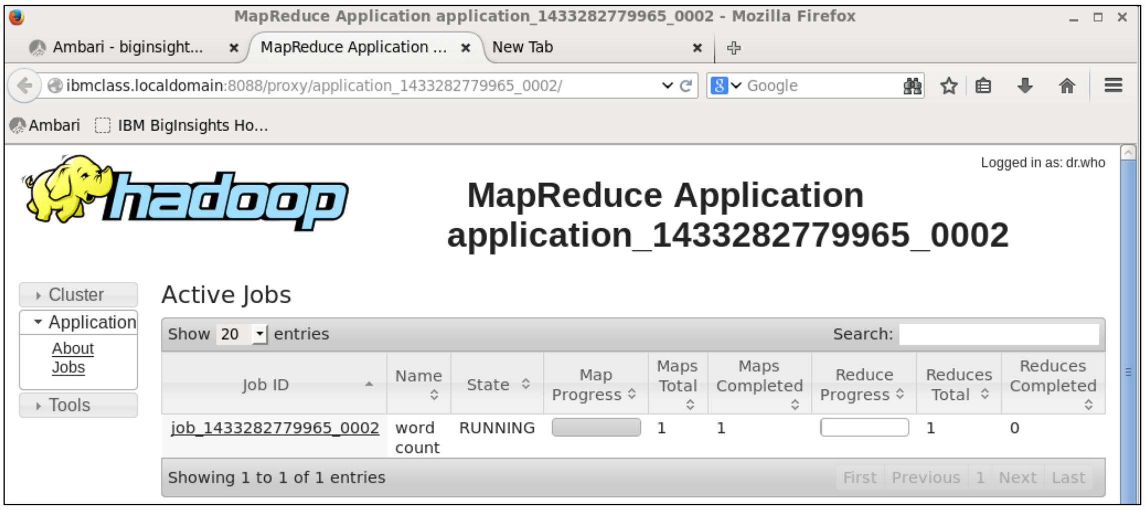
```
15/06/04 14:48:48 INFO mapreduce.Job: The url to track the job:
http://ibmclass.localdomain:8088/proxy/application_1433282779965_0002/
```

9. Type **`q`** to exit from more, and then to remove the directory where your output file was stored (wcout), type **`hadoop fs -rm -R wcout`**.

   It is necessary to remove this directory and all files in it to use the command again without changes. Alternatively you could run the command again, but with a different output directory such as wcout2.

10. You will now run the wordcount program again and, immediately after it starts, you should copy the URL for that run by selecting it, right-clicking, and clicking **Copy**. **Paste** the URL into a Firefox browser within your lab environment.

    Depending on how fast you do this, you will see something like the following, if your MapReduce job is still running:


MapReduce Application application_1433282779965_0002 - Mozilla Firefox
Ambari - biginsight...
MapReduce Application ...
New Tab
ibmclass.localdomain:8088/proxy/application_1433282779965_0002/
Google
Ambari
IBM BigInsights Ho...
Logged in as: dr.who
Cluster
Application
About
Jobs
Tools

MapReduce Application application_1433282779965_0002

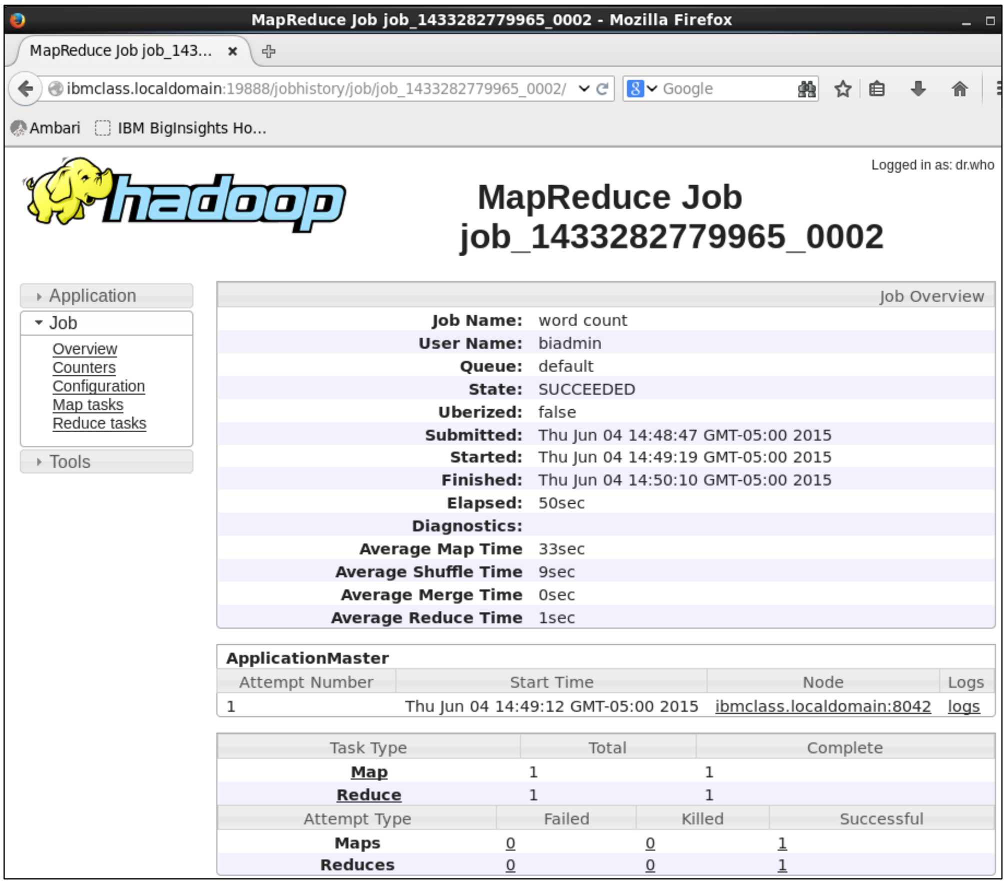
Active Jobs

Show 20 entries Search:

Job ID	Name	State	Map Progress	Maps Total	Maps Completed	Reduce Progress	Reduces Total	Reduces Completed
job_1433282779965_0002	word count	RUNNING		1	1		1	0

Showing 1 to 1 of 1 entries First Previous 1 Next Last

    Or the following, if your MapReduce job has already completed:



MapReduce Job job_1433282779965_0002

Job Overview

Job Name:	word count
User Name:	biadmin
Queue:	default
State:	SUCCEEDED
Uberized:	false
Submitted:	Thu Jun 04 14:48:47 GMT-05:00 2015
Started:	Thu Jun 04 14:49:19 GMT-05:00 2015
Finished:	Thu Jun 04 14:50:10 GMT-05:00 2015
Elapsed:	50sec
Diagnostics:	
Average Map Time	33sec
Average Shuffle Time	9sec
Average Merge Time	0sec
Average Reduce Time	1sec

ApplicationMaster

Attempt Number	Start Time	Node	Logs
1	Thu Jun 04 14:49:12 GMT-05:00 2015	ibmclass.localdomain:8042	logs

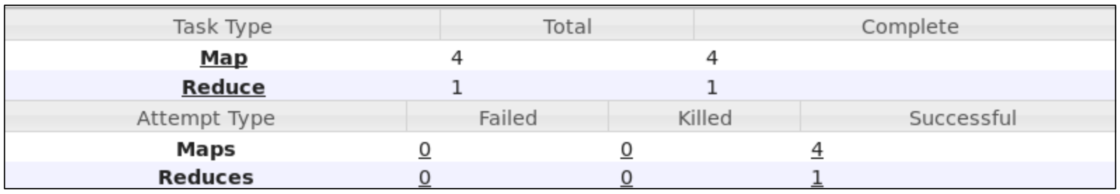
Task Type	Total	Complete
Map	1	1
Reduce	1	1

Attempt Type	Failed	Killed	Successful
Maps	0	0	1
Reduces	0	0	1

11. The same job can be run (and monitored) as a YARN job. To do this, type **`yarn jar /usr/iop/4.0.0.0/hadoop-mapreduce/hadoop-mapreduce-examples.jar wordcount Gutenberg/Frankenstein.txt wcout2.`**
12. You can also run the job against all four files in the Gutenberg directory. To do this, type **`hadoop jar /usr/iop/4.0.0.0/hadoop-mapreduce/hadoop-mapreduce-examples.jar wordcount Gutenberg/*.txt wcout2.`**

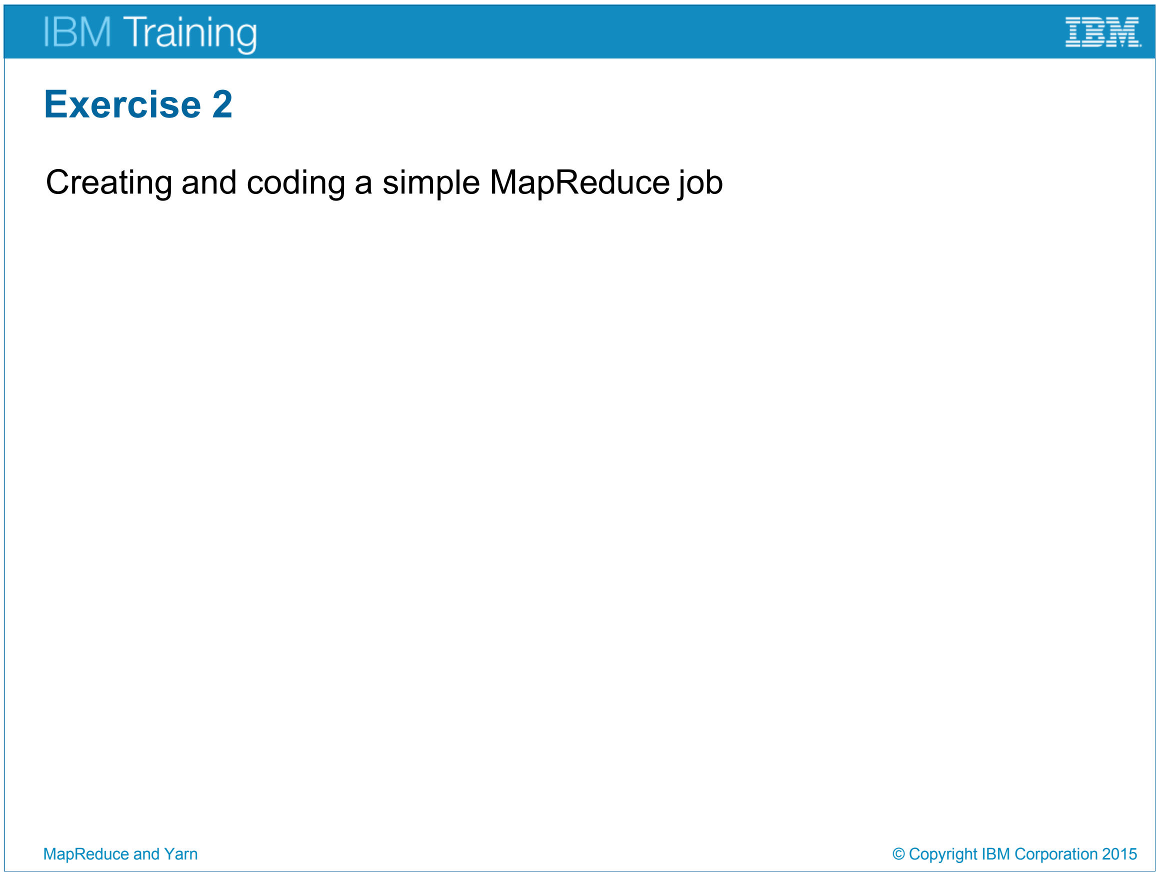
Question 1: How many Mappers were used for your run?

Question 2: How many Reducers were used for your run?



Task Type	Total	Complete	
Map	4	4	
Reduce	1	1	
Attempt Type	**Failed**	**Killed**	**Successful**
Maps	0	0	4
Reduces	0	0	1

Results:
You ran Java programs using Hadoop v2, YARN, and related technologies. You started with an existing sample program, selected from among those available, and progressed towards compiling a program a Java program from scratch and learned in what type of environment it runs.



IBM Training

Exercise 2

Creating and coding a simple MapReduce job

MapReduce and Yarn

© Copyright IBM Corporation 2015

Exercise 2: Creating and coding a simple MapReduce job

Exercise 2: Creating and coding a simple MapReduce job

Purpose:
You will compile and run a more complete version of WordCount that has been written specifically for MapReduce2.

Task 1. Compile and run a more complete version of WordCount that has been written specifically for MapReduce2.

The program that you will use - `WordCount2.java` - has been taken from http://hadoop.apache.org/docs/current/hadoop-mapreduce-client/hadoop-mapreduce-client-core/MapReduceTutorial.html#Example:_WordCount_v2.0

There is a small error in the Java code ("| |" with a space rather than "||" without a space. This has been corrected in the version in /home/labfiles.

This version 2 of WordCount.java is more sophisticated than the one that you have already run, since it allows for splitting on text in addition to and other than whitespace, and also allows you to specify anything that you want to skip when counting words (such as "to", "the", etc.).

There are some limitations; if you are a competent Java programmer, you may want to experiment later with other features. For instance, are all words lowercased when they are tokenized?

Since you are now more familiar with the process of running MR/YARN jobs, the directions provided here will concentrate on the compilation only.

1. To copy the files that you will need to your **biadmin** home directory, type the following commands:

```
cd /home/biadmin/labfiles
cp patternsToSkip run-wc WordCount2.java .
```

2. To move the file patternsToSkip to HDFS, type `hadoop fs -put patternsToSkip .`.
3. Type `hadoop classpath`, which provides you with the CLASSPATH environmental variables that you need for compilation.

   You will not likely want to type that information manually.

4. Compile WordCount.java with the Hadoop2 API and this CLASSPATH:
   ```
   javac -cp `hadoop classpath` WordCount2.java
   ```
   Note here, the use of the back-quote character to substitute the value of the CLASSPATH set into the compilation as the classpath (-cp) needed for the compilation.
5. Create a Jar file that can be run in the Hadoop2/YARN environment:
   ```
   jar cf WC2.jar *.class
   jar tf WC2.jar
   ```
6. To run your Jar file successfully, you will need to remove all the class files from your biadmin Linux directory:
   ```
   rm *.class
   ```
7. You are now ready to run your compiled program with appropriate parameters (see the program logic and the use of these additional parameters on the website listed above). Type the following command on one line:
   ```
   hadoop jar WC2.jar WordCount2 -D wordcount.case.sensitive=false
   ./Gutenberg/*.txt ./wc2out -skip ./patternsToSkip
   ```
   This can also be run with "yarn" and the MR job can be monitored as previously.

Question 1: How many Mappers were used for your run?

Question 2: How many Reducers were used for your run?

8. Look at your output (in wc2out) and see if it differs from what you generated previously.

Results:
You compiled and ran a more complete version of WordCount that had been written specifically for MapReduce2.