Before you begin, first do

module load hadoop-paths

```
################################################
# Exercise 1:
#  Hadoop File system commands
#
################################################
#list directory in hdfs
hadoop fs -ls /
#create a data directory in the default home directory in hdfs
hadoop fs -mkdir data
# copy file from local file system to hdfs
hadoop fs -put test_text.txt data
hadoop fs -put stories data
# check stat of a file
hadoop fs -stat data/test_text.txt
# output text file to stdout
hadoop fs -cat data/test_text.txt
#Practice suggestions
1. how to move, rename a file within hdfs.
2. how to change the replication factor of a particular file
3. how to change the block size of a particular file.
4. what's the difference between storing directory  stories and test_text.txt
################################################
# Exercise 2:
#  Run hadoop application from the example jar
#
################################################
# running word count in example jar
hadoop jar /usr/lib/hadoop-mapreduce/hadoop-mapreduce-examples.jar wordcount -D
mapred.map.tasks=96 -D mapred.reduce.tasks=24 data/test_text.txt test_text_wc
#check the result files
hadoop fs -ls test_text_wc
hadoop fs -cat test_text_wc/part-r-00000
#Practice suggestions
1. What happens when not specify map.tasks and reduce.tasks
```

2. What happens when changing the value of map.tasks and reduce.tasks

3. Try some other command in the example jar using

hadoop jar /usr/lib/hadoop-mapreduce/hadoop-mapreduce-examples.jar

to see available programs e.g.

teragen: create a large file for

hadoop jar /usr/lib/hadoop-mapreduce/hadoop-mapreduce-examples.jar teragen -D mapred.map.tasks=96 100000000 TS-10GB

terasort:

hadoop jar /usr/lib/hadoop-mapreduce/hadoop-mapreduce-examples.jar terasort -D mapred.map.tasks=96 -D mapred.reduce.tasks=24 TS-10GB TS-10GB-sort

calculate pi?

```
################################################
# Exercise 3:
#  compile and run a simple java
#  implementation of wordcount
################################################
# compile WordCount.java
export JAVA_HOME=/usr/lib/jvm/java-1.7.0/
export HADOOP_CLASSPATH=$JAVA_HOME/lib/tools.jar
hadoop com.sun.tools.javac.Main WordCount.java
jar cf wc.jar WordCount*.class
#run compiled jar file
hadoop jar wc.jar WordCount /tmp/data/20news-all/alt.atheism/54564 wc_output
#check the result file
hadoop fs -ls wc_output
hadoop fs -cat wc_output/part-r-00000
#practice suggestions
1. Any improvements to the code?
2. Just count word length longer than 3?
3. try program something useful.
################################################
# Exercise 4:
#  Run hadoop streaming job with bash scripts
#  implementation of wordcount
################################################
hadoop jar /usr/lib/hadoop-mapreduce/hadoop-streaming.jar \
    -D mapred.map.tasks=96 -D mapred.reduce.tasks=24 -D stream.num.map.output.key.fields=1 \
```

```
    -output wc_bash \
    -mapper ./mapwc.sh -reducer ./reducewc.sh \
    -file ./mapwc.sh -file ./reducewc.sh \
    -input /tmp/data/20news-all/alt.atheism \
#Practice suggestion:
1. The bash scripts actually could work without Hadoop
  but just bash/linux command. How?
2. It won't work (too slow) with large file.
  Can you write your own map reduce routines with
  your favorite programming languages that
  work with Hadoop streaming?
3. There is also implementation using python and Rscript. Give it a try.
##################################################
# Exercise 5:
#  Running K-means example with Mahout
#
##################################################
cp -r /work/00791/xwj/hadoop-training/reuters-sgm ~/reuters-sgm
#step 1
mahout org.apache.lucene.benchmark.utils.ExtractReuters ~/reuters-sgm reuters-sgm-extract
hadoop fs -put reuters-sgm-extract
#step 2
mahout seqdirectory -i reuters-sgm-extract -o reuters-seqdir -c UTF-8 -chunk 5
#step 3
mahout seq2sparse -i reuters-seqdir/ -o reuters-seqdir-vectors
#step 4
mahout kmeans -i reuters-seqdir-vectors/tfidf-vectors/ -c reuters-kmeans-clusters -o reuters-kmeans -x 10 -
k 20
#step 5
mahout clusterdump -i reuters-kmeans/clusters-* -d reuters-seqdir-vectors/dictionary.file-0 -dt sequencefile
-b 100 -n 20 -o ./cluster-output.txt
#practice suggestion:
1. Try mahout to see other programs.
##################################################
# Exercise 6:
#  Spark using JAVA basic and how to run examples
#
```

```
################################################
1.word count example
cd /work/00791/xwj/DMS/spark/word_count
spark-submit --class JavaWordCount --master yarn-client --num-executors 16  target/wordcount-1.0-SNAPSHOT.jar file:/work/00791/xwj/DMS/spark/book.txt
2. KMeans
/usr/lib/spark/bin/run-example ml.JavaKMeansExample file:/work/00791/xwj/DMS/spark/kmeans_data.txt 3
3. Run SparkPi example
> /usr/lib/spark/bin/run-example SparkPi 10
or
>spark-submit --class org.apache.spark.examples.SparkPi /usr/lib/spark/examples/lib/spark-examples-1.5.0-cdh5.5.1-hadoop2.6.0-cdh5.5.1.jar 10
################################################
# Exercise 7:
#  Spark using R - SparkR: Word Count, housing stat and logistic regression
#
################################################
0. Go to the exercise directory
cd /work/00791/xwj/DMS/R-training/RBigData/sparkr
ml big-data-r
1. Upload data
source upload-data.sh
2. word count example
cd /work/00791/xwj/DMS/R-training/RBigData/sparkr/wordcount/
source wordcount_sparkR.sh
hadoop fs -cat /user/$USER/output/spark-1/*|less
3. housing example(http://tessera.io/docs-RHIPE/#the-data)
cd /work/00791/xwj/DMS/R-training/RBigData/sparkr/housing/
source housing.sh
hadoop fs -cat /user/$USER/output/spark/housing-seq/*|less
4. logistic regression
cd /work/00791/xwj/DMS/R-training/RBigData/sparkr/logisticRegression/
source logisticRegression.sh
################################################
# Exercise 8:
#  Spark using Python - : Word Count example
#
```

```
################################################
cd /work/00791/xwj/DMS/spark
source wordcount.sh
hadoop fs -cat output/wc_py/*|less
################################################
# Exercise 9:
# SparkShell  with Scala basics
#
################################################
#start spark-shell with following:
>spark-shell --master=yarn-client
#Once spark-shell started type
:help           Show spark-shell commands help
:sh <command>        Run a shell command from within spark shell
#type in following for an word count using scala.
val f = sc.textFile("/tmp/data/book.txt")
val words = f.flatMap(_.split(" "))
val wc = words.map(w => (w, 1)).reduceByKey(_ + _)
wc.saveAsTextFile("SS-counts")
```