Run the WordCount program Instructions

Learning Goals

By the end of this activity, you will be able to:

- Explore MapReduce applications and run your first application.
- Execute the WordCount application on Shakespeare works.
- Copy the results from WordCount out of HDFS.

1. Start the cotainerized cluster. If you paused your containers in the previous Hands On module, go through the steps 3-5 of the previous module to start the containerized cluster.

2. Exploring MapReduce applications. Hadoop comes with several MapReduce applications These applications are located in *share/hadoop/mapreduce/hadoop-mapreduce-examples-3.3.6.jar.*

Run *yarn jar share/hadoop/mapreduce/hadoop-mapreduce-examples-3.3.6.jar* to display them.

```
yarn jar share/hadoop/mapreduce/hadoop-mapreduce-examples-3.3.6.jar
```

```
bash-4.2$ yarn jar share/hadoop/mapreduce/hadoop-mapreduce-examples-3.3.6.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.
```

We can learn how to run any application by examining its command-line arguments. Let's take for example pi

Run *yarn jar share/hadoop/mapreduce/hadoop-mapreduce-examples-3.3.6.jar pi*

```
yarn jar share/hadoop/mapreduce/hadoop-mapreduce-examples-3.3.6.jar pi
```

```
bash-4.2$ yarn jar share/hadoop/mapreduce/hadoop-mapreduce-examples-3.3.6.jar pi
Usage: org.apache.hadoop.examples.QuasiMonteCarlo <nMaps> <nSamples>
Generic options supported are:
-conf <configuration file>        specify an application configuration file
-D <property=value>               define a value for a given property
-fs <file:///|hdfs://namenode:port> specify default filesystem URL to use, overrides 'fs.defaultFS' property from configurations.
-jt <local|resourcemanager:port>  specify a ResourceManager
-files <file1,...>                specify a comma-separated list of files to be copied to the map reduce cluster
-libjars <jar1,...>               specify a comma-separated list of jar files to be included in the classpath
-archives <archive1,...>          specify a comma-separated list of archives to be unarchived on the compute machines

The general command line syntax is:
command [genericOptions] [commandOptions]
```

3. Running a first MapReduce example. We will start by running pi as our first example.

Run *yarn jar share/hadoop/mapreduce/hadoop-mapreduce-examples-3.3.6.jar pi 10 15*

```
yarn jar share/hadoop/mapreduce/hadoop-mapreduce-examples-3.3.6.jar pi 10 15
```

```
bash-4.2$ yarn jar share/hadoop/mapreduce/hadoop-mapreduce-examples-3.3.6.jar pi 10 15
Number of Maps  = 10
Samples per Map = 15
Wrote input for Map #0
Wrote input for Map #1
Wrote input for Map #2
Wrote input for Map #3
Wrote input for Map #4
Wrote input for Map #5
Wrote input for Map #6
Wrote input for Map #7
Wrote input for Map #8
Wrote input for Map #9
Starting Job
```

Once the job is completed, you should see a result like the following:

```
Job Finished in 20.468 seconds
Estimated value of Pi is 3.17333333333333333333
```

Furthermore, this process created the HDFS user folder automatically, so we don't need to go again through that step.

4. See WordCount command line arguments. Run *yarn jar share/hadoop/mapreduce/hadoop-mapreduce-examples-3.3.6.jar wordcount*

```
yarn jar share/hadoop/mapreduce/hadoop-mapreduce-examples-3.3.6.jar wordcount
```

```
bash-4.2$ yarn jar share/hadoop/mapreduce/hadoop-mapreduce-examples-3.3.6.jar wordcount
Usage: wordcount <in> [<in>...] <out>
```

5. **Run WordCount.** If you started a new session, make sure to place the words.txt file into the HDFS. You can either copy the file (*hdfs dfs -copyFromLocal words.txt*) or run *hdfs dfs -put words.txt* which will directly place the file into HDFS.

```
hdfs dfs -copyFromLocal words.txt
```

```
bash-4.2$ hdfs dfs -copyFromLocal words.txt
bash-4.2$ hdfs dfs -ls
Found 1 items
-rw-r--r--   1 hadoop supergroup    5458199 2024-04-03 17:56 words.txt
```

Run *yarn jar share/hadoop/mapreduce/hadoop-mapreduce-examples-3.3.6.jar wordcount words.txt out*

```
jar share/hadoop/mapreduce/hadoop-mapreduce-examples-3.3.6.jar wordcount words.txt out
```

```
bash-4.2$ yarn jar share/hadoop/mapreduce/hadoop-mapreduce-examples-3.3.6.jar wordcount words.txt out
2024-04-03 19:08:07 INFO  DefaultNoHARMFailoverProxyProvider:64 - Connecting to ResourceManager at resourcemanager/172.27.0.5:8032
2024-04-03 19:08:07 INFO  JobResourceUploader:907 - Disabling Erasure Coding for path: /tmp/hadoop-yarn/staging/hadoop/.staging/job_1712166917019_0001
2024-04-03 19:08:07 INFO  FileInputFormat:300 - Total input files to process : 1
2024-04-03 19:08:07 INFO  JobSubmitter:202 - number of splits:1
2024-04-03 19:08:07 INFO  JobSubmitter:298 - Submitting tokens for job: job_1712166917019_0001
2024-04-03 19:08:07 INFO  JobSubmitter:299 - Executing with tokens: []
2024-04-03 19:08:07 INFO  Configuration:2854 - resource-types.xml not found
2024-04-03 19:08:07 INFO  ResourceUtils:476 - Unable to find 'resource-types.xml'.
2024-04-03 19:08:08 INFO  YarnClientImpl:338 - Submitted application application_1712166917019_0001
2024-04-03 19:08:08 INFO  Job:1682 - The url to track the job: http://resourcemanager:8088/proxy/application_1712166917019_0001/
2024-04-03 19:08:08 INFO  Job:1727 - Running job: job_1712166917019_0001
2024-04-03 19:08:14 INFO  Job:1748 - Job job_1712166917019_0001 running in uber mode : false
2024-04-03 19:08:14 INFO  Job:1755 -  map 0% reduce 0%
2024-04-03 19:08:20 INFO  Job:1755 -  map 100% reduce 0%
2024-04-03 19:08:24 INFO  Job:1755 -  map 100% reduce 100%
2024-04-03 19:08:24 INFO  Job:1766 - Job job_1712166917019_0001 completed successfully
2024-04-03 19:08:24 INFO  Job:1773 - Counters: 54
```

6. **See WordCount output directory.** Once WordCount is finished, let's verify the output was created. First, let's see that the output directory, *out,* was created in HDFS by running *hdfs dfs –ls*

```
hdfs dfs -ls
```

```
bash-4.2$ hdfs dfs -ls
Found 2 items
drwxr-xr-x   - hadoop supergroup          0 2024-04-03 19:08 out
-rw-r--r--   1 hadoop supergroup    5458199 2024-04-03 17:56 words.txt
```

We can see there are now two items in HDFS: *words.txt* is the text file that we previously created, and *out* is the directory created by WordCount.

7. **Look inside output directory.** The directory created by WordCount contains several files. Look inside the directory by running *hdfs dfs -ls out*

```
hdfs dfs -ls out
```

```
bash-4.2$ hdfs dfs -ls out
Found 2 items
-rw-r--r--   1 hadoop supergroup          0 2024-04-03 19:08 out/_SUCCESS
-rw-r--r--   1 hadoop supergroup     717768 2024-04-03 19:08 out/part-r-00000
```

The file *part-r-00000* contains the results from WordCount. The file _SUCCESS means WordCount executed successfully.

8. **Copy WordCount results to local file system.** Copy *part-r-00000* to the namenode-1 container file system by running *hdfs dfs -copyToLocal out/part-r-00000 local.txt*

```
hdfs dfs -copyToLocal out/part-r-00000 local.txt
```

```
bash-4.2$ hdfs dfs -copyToLocal out/part-r-00000 local.txt
bash-4.2$
```

9. Exit the container. Run *exit* to return to your local system terminal shell.

```
exit
```

```
bash-4.2$ exit
exit
PS C:\Users\[redacted]\Desktop\Docker Containers for Coursera\coursera\big-data-1\hadoop>
```

9. View the WordCount results. Run *docker cp hadoop-namenode-1:/opt/hadoop/local.txt ./local.txt* to copy the text file into your local system.

```
docker cp hadoop-namenode-1:/opt/hadoop/local.txt ./local.txt
```

```
PS C:\Users\[redacted]\Desktop\coursera\big-data-1\hadoop> docker cp hadoop-namenode-1:/opt/hadoop/local.txt ./local.txt
Successfully copied 719kB to C:\Users\[redacted]\Desktop\coursera\big-data-1\hadoop\local.txt
```

Go to your *big-data-1/hadoop* directory and open the text file to view the results using your system's text editor (Notepad for Windows or TextEdit for macOS).

```
local
File   Edit   View

"       241
"'Tis   1
"A      4
"AS-IS".        1
"Air,"  1
"Alas,  1
"Amen"  2
"Amen"? 1
"Amen," 1
"And    1
"Aroint 1
"B      1
"Black  1
"Break  1
"Brutus"        1
"Brutus,        2
"C      1
"Caesar"?       1
"Caesar,        1
"Caesar."       2
"Certes,"       1
```

Each line of the results file shows the number of occurrences for a word in the input file. For example, *Amen* appears twice in the input, but *Aroint* appears only once. Keep in mind that we did not pre process the words, so that's why you see some repeated words, where the application identifies them a different given an additional character (like "Brutus" vs "Brutus,").

10. Delete the container. If you're planning to come back to the course later, run *docker compose down* to delete all your containers. If you're planning to go to the Quiz from here, do not delete the containers and go directly to the Quiz.

```
PS C:\Users\[redacted]\Desktop\coursera\big-data-1\hadoop> docker compose down
[+] Running 5/5
 ✔Container hadoop-datanode-1          Removed                                  0.6s
 ✔Container hadoop-resourcemanager-1   Removed                                  1.2s
 ✔Container hadoop-nodemanager-1       Removed                                  0.9s
 ✔Container hadoop-namenode-1          Removed                                  1.1s
 ✔Network hadoop_default               Removed                                  0.2s
```