Lab 4. Running Mapreduce

# Exercise 1. Install mapreduce

Choose a node to install the relevant packages (For this lab, it’s probably going to be the same node as your name node).

First we install the relevant packages.

|  |
| --- |
| yum install –y hadoop-mapreduce hadoop-doc |

Verify that the sample jar file now works and shows the example programs.

|  |
| --- |
| yarn jar /usr/share/doc/hadoop-0.20-mapreduce/examples/hadoop-examples.jar |

Replace any existing content in the file /etc/hadoop/conf/mapred-site.xml with the following:

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?>  <configuration>  <property>  <name>mapreduce.framework.name</name>  <value>yarn</value>  </property>  </configuration> |

Next, we wil configure YARN to load an auxiliary service that can perform the shuffle phase of the mapreduce algorithm by adding the following to the existing yarn-site.xml config:

|  |
| --- |
| <property>  <name>yarn.nodemanager.aux-services</name>  <value>mapreduce\_shuffle</value>  </property>  <property>  <name>yarn.nodemanager.aux-services.mapreduce\_shuffle.class</name>  <value>org.apache.hadoop.mapred.ShuffleHandler</value>  </property> |

Now we will configure hadoop application managers to store their temporary files within user directories in HDFS by adding the following lines to the existing yarn-site.xml config file.

|  |
| --- |
| <property>  <name>yarn.app.mapreduce.am.staging-dir</name>  <value>/user</value>  </property> |

Make sure that this temporary directory exists in hdfs.

|  |
| --- |
| su -c ‘hdfs dfs -mkdir /tmp’ hdfs  su -c ‘hdfs dfs -chmod -R 1777 /tmp’ hdfs |

Next restart :YARN to allow the changes to take effect.

|  |
| --- |
| service hadoop-yarn-resourcemanager restart  service hadoop-yarn-nodemanager restart  ps -u yarn w |

Now let’s create some sample data for our mapreduce job.

|  |
| --- |
| find /etc -exec strings {} \+ > /tmp/etc.words  find /usr/share/doc -exec strings {} \+ > /tmp/doc.words  ls -lh /tmp\*.words |

Let’s upload this file to hdfs.

|  |
| --- |
| su – student  hdfs dfs –put /tmp/etc.words /upload/  hdfs dfs –put /tmp/doc.words /upload/ |

Now we’ll run the mapreduce job, placing the output in the users home directory.

|  |
| --- |
| # yarn jar /usr/lib/hadoop-0.20-mapreduce/hadoop-examples.jar wordcount /upload/etc.words |

Look at the temporary directories that were created by the mapreduce job.

|  |
| --- |
| hdfs dfs –ls /home/<username> |

Look at the yarn logs.

|  |
| --- |
| su –c ‘ls –R /yarn/logs/application\_\*0001’ root |

Now let’s get the results from the mapreduce job.

|  |
| --- |
| hdfs dfs –get /home/<username>/result1  ls –l result1/ |

Let’s look, for example, for all words that occurred between 500 and 600 times.

|  |
| --- |
| awk ‘{if ($2 > 500 && $2 < 600) {print}}’ result1/part-r-00000 |

Now we’ll add the node managers on the additional nodes in our cluster.

First, let’s see how long it takes to run the job with only one node.

|  |
| --- |
| time yarn jar /usr/lib/hadoop-0.20-mapreduce/hadoop-examples.jar wordcount \ /upload/doc.words result2 |

Now let’s push our MapReduce/YARN configs to the other nodes.

|  |
| --- |
| cd /etc/hadoop.conf  for n in 1 2 4; do scp mapred-site.xml yarn-site.xml node$n:/etc/hadoop/conf; done |

Start the nodemanager daemon on the other nodes.

|  |
| --- |
| allnodes service hadoop-yarn-nodemanager start  yarn node list | grep Nodes |

Now, as the student user run the mapreduce job again and time it to see what the difference is.

|  |
| --- |
| time yarn jar /usr/lib/hadoop-0.20-mapreduce/hadoop-examples.jar wordcount /upload/doc.words result3 |

Now run the same job with multiple reducers.

|  |
| --- |
| yarn jar /usr/lib/hadoop-0.20-mapreduce/hadoop-examples.jar \  wordcount mapreduce.job.reduces=3 /upload/doc.words results\_multi  hdfs dfs –ls results\_multi  hdfs dfs –get results\_multi  grep ^watch part-r-0000”\* |

Note that the data has been partitioned across multiple files We can merge them back by using standard Linux commands.

|  |
| --- |
| sort part-r-0000\* -o merged  grep ^watch merged |

However we can do the same thing more easily by using the –getmerge option to hdfs dfs

|  |
| --- |
| hdfs dfs –getmerge results\_multi merged |