Deployment of Hadoop, Spark, and implementation of PageRank in a 3-nodes cluster

Group 16: You Hu 12036234, Wenchen Lai 12482625

For assignment 2.2, we installed Hadoop and Spark on a 3-nodes cluster and performed a PageRank application with the specified requirements. The cluster was created on our laptop by VMware as we could have more flexibility and privilege.

There is one thing should be mentioned is that we created VMs manually one by one. The original idea was to install Hadoop and its dependency on one VM and clone it into a 3-nodes cluster. However, there was one issue that the MAC addresses in three nodes were the same, and there would be a conflict which led to no Internet access. After the creation of VMs, we set one node as the master node and others as worker nodes.

On the master node, the NameNode and ResourceManager services will be running while on worker nodes there will be DataNode and NodeMananeger services running. After installing Hadoop on 3 nodes, we started HDFS and YARN. And then we uploaded the data to the HDFS and installed Spark at the Master node. For the configuration of Spark, we set yarn as spark master, therefore, the application would be running on the Hadoop cluster instead of standalone mode.

For the PageRank application, we reused the example code provided by Spark and modified it to show only the top 10 URLs. We configured the iteration number as 10 and performed it multiple times on the data with different parameters for Spark. The result is shown on the next page as well as the running time of each parameter setting.

According to the results, we can see that the running times for each parameter setting range from 9 mins to 13 mins. However, from what we observed, the running times of each setting are very close to 10 mins, as for those lasted for over 12mins, there were about 3 mins on the “accepting” status, which indicated the YARN was scheduling resources however we did not submit multiple jobs at same time. Therefore, the fact is that the running times are close to each other in different settings. We found that according to the log messages, the tasks were only executed on worker1. So, we guess it is because the data is not big enough and it would lead to too much communication overhead if we place data on two nodes, and Spark just automatically optimized the processing to set data locality higher priority. Even if we forced the dataset to be parallelized or increased executor number(we only assign 1 processer to one VM) the tasks were still running on worker1. So, when we shut down one worker, the performance is still the same. Besides, the memory will only be effective when it is lower than 1G(default).

Results: Top 10 URL IDs and their ranks
272919 has rank: 6531.3246237524645.
438238 has rank: 4335.323158564439.
571448 has rank: 2383.8976074118896.
601656 has rank: 2195.394075596729.
316792 has rank: 1855.690875790152.
319209 has rank: 1632.8193684975702.
184094 has rank: 1532.2842374483398.
571447 has rank: 1492.9301630938794.
401873 has rank: 1436.160093346929.
66244 has rank: 1261.578395867334

time spark-submit spark/examples/src/main/python/pagerank.py ./web-BerkStan.txt 10
real 12m23.545s
user 0m16.003s
sys 0m3.229s
.
time spark-submit --master yarn spark/examples/src/main/python/pagerank.py
./web-BerkStan.txt 10
real 9m36.003s
user 0m15.582s
sys 0m2.835s

time spark-submit --master yarn --num-executors 1
spark/examples/src/main/python/pagerank.py ./web-BerkStan.txt 10
real 12m15.193s
user 0m16.150s
sys 0m3.076s

time spark-submit --executor-memory 2G --master yarn
spark/examples/src/main/python/pagerank.py ./web-BerkStan.txt 10
real 13m28.136s
user 0m16.326s
sys 0m2.735s

time spark-submit --master yarn --num-executors 2 --executor-memory 512m
spark/examples/src/main/python/pagerank.py ./web-BerkStan.txt 10
real 16m10.892s
user 0m23.398s
sys 0m5.784s