CS553 Programming Assignment #2 (part B)

Sort on Hadoop/Spark

Instructions:

- *Assigned date: Thursday, 04/12/18*
- *Due date: 11:59PM on Thursday, 04/26/18*
- *Maximum Points: 100%*
- *This programming assignment must be done individually*
- *Please post your questions to the Piazza forum*
- *Only a softcopy submission is required; it must be submitted through GIT*
- *Late submission will be penalized at 20% per day (beyond the 4-day late pass).*

1. Introduction

The goal of this programming assignment is to enable you to gain experience programming with:

- The Hadoop framework (http://hadoop.apache.org/)
- The Spark framework (http://spark.apache.org/)

In PA2(a), you implemented an external sort and compared it to the Linux sort. You will now expand into implementing sort with Hadoop and with Spark.

2. Your Assignment

This programming assignment covers sort through Hadoop and Spark on multiple nodes. You can use any Linux system to implement your application, but you must use the Proton Cluster accessible at 216.47.142.37; each Hadoop group (cluster) has 4 nodes, each node having 4-cores, 8GB of memory, and 80GB of SSD storage. You can use the same accounts from PA2(a). Your Hadoop and Spark applications must read the data from HDFS. You can find three datasets to sort: 8GB (/input/data-8GB.in), 20GB (/input/data-20GB.in), and 80GB (/input/data-80GB.in). Be aware that after job completion any data you created in HDFS will be cleared, as well as the data found in /tmp directory on each node will be cleared. The data was generated by gensort, which you can find more information about at http://www.ordinal.com/gensort.html.

This assignment will be broken down into several parts, as outlined below:

Hadoop Sort: Implement the HadoopSort application (you must use Java). You must specify the number of reducers to ensure you use all the resources of the 4-node cluster. You must read the data from HDFS, sort it, and then store the sorted data in HDFS and validate it with valsort. Measure the time from reading from HDFS, sorting, and writing to HDFS; do not include the time to run valsort.

Spark Sort: Implement the SparkSort application (you must use Java). Make sure to use RDD to speed up the sort through in-memory computing. You must read the data from HDFS, make use of RDD, sort, and write the sorted data back to HDFS, and finally validate the sorted data with valsort. Measure the time from reading from HDFS, sorting, and writing to HDFS; do not include the time to run valsort.

Performance: Compare the performance of your shared-memory external sort, the Linux "sort" (more information at http://man7.org/linux/man-pages/man1/sort.1.html), Hadoop Sort, and Spark Sort on a 4-node cluster of the Proton Cluster with both 8GB, 20GB, and 80GB datasets. Fill in the table below, and then derive

new tables or figures (if needed) to explain the results. Your time should be reported in seconds. Some of the things that will be interesting to explain are: how many threads, mappers, reducers, you used in each experiment; how many times did you have to read and write the dataset for each experiment; what speedup and efficiency did you achieve?

What conclusions can you draw? Which seems to be best at 1 node scale? How about 4 nodes? Can you predict which would be best at 100 node scale? How about 1000 node scales? Compare your results with those from the Sort Benchmark (http://sortbenchmark.org), specifically the winners in 2013 and 2014 who used Hadoop and Spark. Also, what can you learn from the CloudSort benchmark, a report can be found at (http://sortbenchmark.org/2014_06_CloudSort_v_0_4.pdf). All of these questions must be addressed in your final report write-up for this assignment.

Complete Table 1 outlined below. Perform the experiments outlined above, and complete the following table:

Table 1: Performance evaluation of sort (weak scaling – small dataset)

Experiment	Shared Memory (1VM 2GB)	Linux Sort (1VM 2GB)	Hadoop Sort (4VM 8GB)	Spark Sort (4VM 8GB)
Computation Time (sec)				
Data Read (GB)				
Data Write (GB)				
I/O Throughput (MB/sec)				
Speedup				
Efficiency				

Table 2: Performance evaluation of sort (strong scaling – large dataset)

Experiment	Shared Memory (1VM 20GB)	Linux Sort (1VM 20GB)	Hadoop Sort (4VM 20GB)	Spark Sort (4VM 20GB)
Computation Time (sec)				
Data Read (GB)				
Data Write (GB)				
I/O Throughput (MB/sec)				
Speedup				
Efficiency				

Table 3: Performance evaluation of sort (weak scaling – large dataset)

Experiment	Shared Memory (1VM 20GB)	Linux Sort (1VM 20GB)	Hadoop Sort (4VM 80GB)	Spark Sort (4VM 80GB)
Computation Time (sec)				
Data Read (GB)				
Data Write (GB)				
I/O Throughput (MB/sec)				
Speedup				
Efficiency				

3. Grading

The grading will be done according to the rubric below:

- Hadoop sort implementation/scripts: 15 points
- Spark sort implementation/scripts: 25 points
- Readme.txt: 5 points
- Performance evaluation, data, explanations, etc: 50 points
- Followed instructions on deliverables: 5 points

The maximum score that will be allowed is 100 points.

4. Deliverables

You are to write a report (pa2b_report.pdf). Add a brief description of the problem, methodology, and runtime environment settings. You are to fill in the table on the previous page. Please explain your results, and explain the difference in performance? Include logs from your application as well as valsort (e.g. standard output) that clearly shows the completion of the sort invocations with clear timing information and experiment details; include separate logs for shared memory sort and Linux sort, for each dataset. Valsort can be found as part of the gensort suite (http://www.ordinal.com/gensort.html), and it is used to validate the sort. As part of your submission you need to upload to your private git repository 6 Slurm scripts, a build script, the source code for your implementation, the report, a readme file, and 6 log files. Here are the naming conventions for the required files:

- hadoopsort8GB.slurm
- hadoopsort20GB.slurm
- hadoopsort80GB.slurm
- sparksort8GB.slurm
- sparksort20GB.slurm
- sparksort80GB.slurm
- Makefile / build.xml (Ant) / pom.xml (Maven)
- HadoopSort.java
- SparkSort.java
- pa2b_report.pdf
- readme.txt
- HadoopSort8GB.log
- HadoopSort20GB.log
- HadoopSort80GB.log
- SparkSort8GB.log
- SparkSort20GB.log
- SparkSort80GB.log

5. Where you will submit

You will have to submit your solution to a private git repository created for you. You will have to firstly clone the repository. Then you will have to add or update your source code, documentation and report. Your solution will be collected automatically after the deadline grace period. If you want to submit your homework later, you will have to push your final version of the solution and you will have let the TAs know of it through email cs553-s18@datasys.cs.iit.edu. There is no need to submit anything on BB for this assignment. Here are

some examples on how to clone your private repository and how to add files to it (replace **userid** with your username):

```
git clone ssh://userid@216.47.142.37/exports/git/userid/cs553-pa2b.git
cd cs553-pa2b/
touch readme.txt
cat "Username A20*" > readme.txt
git add readme.txt
git commit -m "Added the readme file"
git push
```

If you cannot access your repository contact the TAs. You can find a git cheat sheet here: https://www.git-tower.com/blog/git-cheat-sheet/