CS553 Programming Assignment #2 (part A)

Sort on Single Shared Memory Node

Instructions:

- *Assigned date: Wednesday, 04/04/18*
- *Due date: 11:59PM on Wednesday, 04/11/18*
- *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 external data sort and multi-threaded programming.

2. Your Assignment

This programming assignment covers the external sort (see https://en.wikipedia.org/wiki/External_sorting) application implemented in a single node shared memory multi-threaded approach. You can use any Linux system to implement your application, but you must use the Neutron Cluster accessible at 216.47.142.38; each node has 4-cores, 8GB of memory, and 80GB of SSD storage. You will receive instructions on your account userid and password over Piazza. Your sorting application could read a large file and sort it in place (your program should be able to sort larger than memory files, also known as external sort). You can find two datasets to sort: 2GB (/input/data-2GB.in) and 20GB (/input/data-20GB.in). You can store intermediate and output data in /tmp, but be aware that after job completion the /tmp directory 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:

Shared-Memory External Sort: Implement the Shared-Memory TeraSort application in your favorite language (Java, C, C++); the next part of this assignment will involve you sorting data in Hadoop and Spark (which has best support for Java and Scala). You should make your Shared-Memory TeraSort multi-threaded to take advantage of multiple cores and SSD storage (which also requires multiple concurrent requests to achieve peak performance).

Performance: Compare the performance of your shared-memory external sort with that from Linux "sort" (more information at http://man7.org/linux/man-pages/man1/sort.1.html) on a single node of the Neutron Cluster with both the 2GB and 20GB 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.

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

Table 1: Performance evaluation of TeraSort

Experiment	Shared Memory (1VM 2GB)	Linux Sort (1VM 2GB)	Shared Memory (1VM 20GB)	Linux Sort (1VM 20GB)
Compute Time (sec)				
Data Read (GB)				
Data Write (GB)				
I/O Throughput (MB/sec)				

3. Grading

The grading will be done according to the rubric below:

- Shared memory sort implementation/scripts: 40 points
- Slurm scripts: 10 points
- Readme.txt: 5 points
- Performance evaluation, data, explanations, etc: 40 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 (pa2a_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 4 Slurm scripts, a build script, the source code for your implementation, the report, a readme file, and 4 log files. Here are the naming conventions for the required files:

- mysort2GB.slurm
- mysort20GB.slurm
- linsort2GB.slurm
- linsort20GB.slurm
- Makefile / build.xml (Ant) / pom.xml (Maven)
- MySort.java / mysort.c / MySort.cpp
- pa2a_report.pdf
- readme.txt
- mysort2GB.log
- mysort20GB.log
- linsort2GB.log
- linsort20GB.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@216.47.142.38/exports/git/userid/cs553-pa2a.git
cd cs553-pa2a/
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/