CS553 Homework #6

Sort on Single Shared Memory Node

Instructions:

- *Assigned date: Monday, 04/01/19*
- *Due date: 11:59PM on Monday, 04/08/19*
- *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 will automatically be collected through GIT after the deadline; email confirmation will be sent to your HAWK email address*
- *Late submission will be penalized at 10% per day; an email to cs553-s19-group@iit.edu with the subject "CS553: late homework submission" must be sent*

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 for your development, but you must use the Chameleon testbed [https://www.chameleoncloud.org]; more information about the hardware in this testbed can be found at https://www.chameleoncloud.org/about/hardware-description/, under Standard Cloud Units. Even more details can be found at https://www.chameleoncloud.org/user/discovery/, choose "Compute", then Click the "View" button. You have been created accounts on this virtual cluster for this assignment; if you have not received your accounts information, reach out to the TAs for this information. You will use two types of instances for this assignment: m1.medium (2-cores, 4GB of memory, and 40GB of SSD storage) and m1.xlarge (8-cores, 16GB of memory, and 160GB of SSD storage).

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 must generate your input data by gensort, which you can find more information about at http://www.ordinal.com/gensort.html. You will need three datasets of different sizes: 1GB, 10GB, and 40GB.

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 (choose between Java, C, C++, or Python); the next part of this assignment will involve you sorting data in Hadoop and Spark (which has best support for Java, Scala, and Python). 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). You want to control concurrency separately between threads that read/write to/from disk, and threads that sort data once data was loaded in memory. Your sort should be flexible enough to handle different types of storage (where you need different number of threads), and different compute resources (where you need different number of threads based on the number of cores). You may only use the PThread library in C/C++. In other languages, you can only use what is built in to these languages. You must implement your own I/O routines and sorting routines.

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 with all three 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 Single Node TeraSort

Experiment	**Shared Memory** (m1.medium **1GB)**	**Linux Sort** (m1.medium **1GB)**	**Shared Memory** (m1.medium **10GB)**	**Linux Sort** (m1.medium **10GB)**	**Shared Memory** (m1.xlarge **40GB)**	**Linux Sort** (m1.xlarge **40GB)**
Data Read (GB)						
Data Write (GB)						
Sort Time (sec)						
Overall I/O Throughput (MB/sec)						

3. Grading

The grading will be done according to the rubric below:

- Shared memory sort implementation/scripts: 50 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 (hw6_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 a build script, the source code for your implementation, benchmark scripts, the report, a readme file, and 6 log files. Here are the naming conventions for the required files:

- Makefile / build.xml (Ant) / pom.xml (Maven)
- MySort.java / mysort.c / MySort.cpp / mysort.py
- Hw6_report.pdf
- readme.txt
- mysort1GB.log
- mysort10GB.log
- mysort40GB.log
- linsort1GB.log
- linsort10GB.log
- linsort40GB.log

5 Where you will submit

You will have to submit your solution to a private git repository created for you at git@gitlab.com:cs553-2019/<student email id>.git. A directory structure for all the repositories will be created that looks like this: hw1/, hw2/, ... You will have to firstly clone the repository. Then you will have to add or update your source code, documentation and report. Make sure this assignment's source code can be found in your hw6 folder.

Your solution will be collected automatically after the deadline. If you want to submit your homework later, you will have to push your final version to your GIT repository and you will have let the TAs know of it through email cs553-s19-group@iit.edu. There is no need to submit anything on BB for this assignment. 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/

Submit code/report through GIT.

Grades for late programs will be lowered 10% per day late.