CS 553 - Cloud Computing

Programing Assignment 2 - Part A

Bhagyashree Bagwe (A20399761)

Problem statement :

"Perform external memory sort on single node with shared memory in a multi-threaded approach."

Methodology:

External memory sort is utilized for sorting larger than memory files. When data do not fit in main memory (RAM), external (or secondary) memory is used for storing chunks of the file temporarily.

In our experiment we are using two input data files one of size 2GB and another of size 20GB generated using gensort program, containing ASCII data.

I have developed two separate programs for sorting 2GB and 20GB data files. The flow of execution for both the programs goes as follows:

2GB external sort:

- Divide 2GB input file into 10 temporarily files of 200MB each and store them on disk temporarily
- Read one 200MB chunk at a time into the main memory
- Sort it and write it back to the disk
- Repeat above 2 steps for all temp input files
- When all the 10 files are sorted, merge all the sorted chunks in a multi threaded fashion by dividing chunks between threads

20GB external sort:

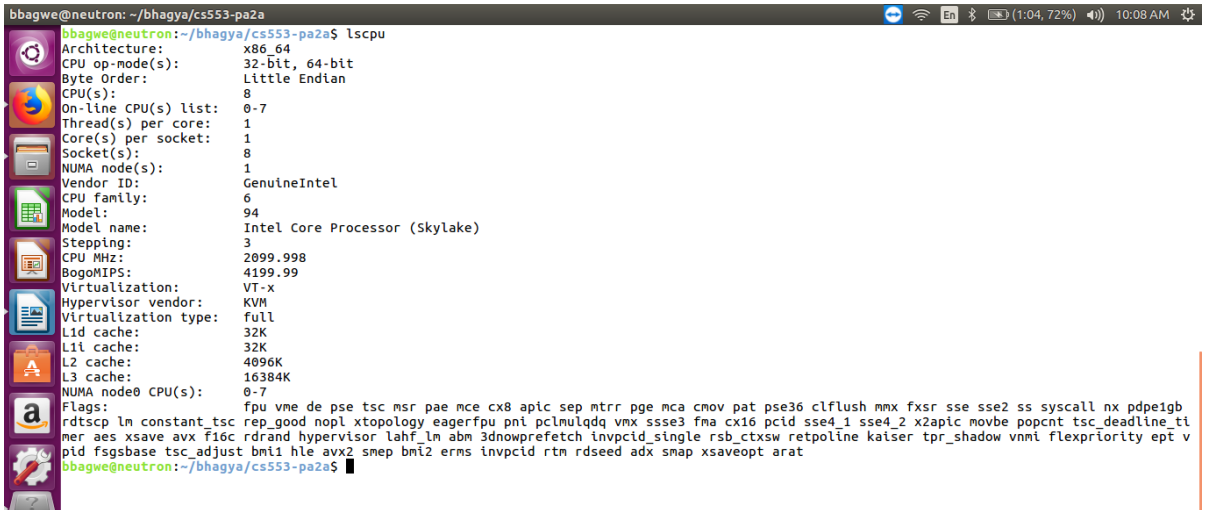
- Divide 20GB file into 10 chunks of 2GB each
- Pass each 2GB file as a input to above 2GB external sort program
- Merge all the sorted chunks in a multi threaded fashion by dividing chunks between threads for merging

Language of implementation :

Java

Runtime Environment:

Programs are executed on a Neutron cluster consisting of login and compute nodes connected via NFS.
The login nodes have capacity of 3GB RAM and compute nodes have capacity of 8GB RAM and 80GB of SSD storage which is essentially used for storing all the intermediary chunks. While executing the source code is placed on the login node and is executed on the compute nodes using the slurm job scheduling.



Performance Evaluation:

Experiment	Shared Memory (1VM 2 GB)	Linux Sort (1VM 2GB)	Shared Memory (1VM 20GB)	Linux Sort (1VM 20GB)
Compute Time (sec)	125	34.98	1141.25	439.024
Data Read (GB)	7.60	~4	116	~40
Data Write (GB)	8.80	~4	128	~40
I/O throughput (MB/sec)	130	~230	210	~180

Since access to disk drives is much slower than access to RAM, analysis of external-memory algorithms usually focuses on the number of disk accesses (I/O operations)

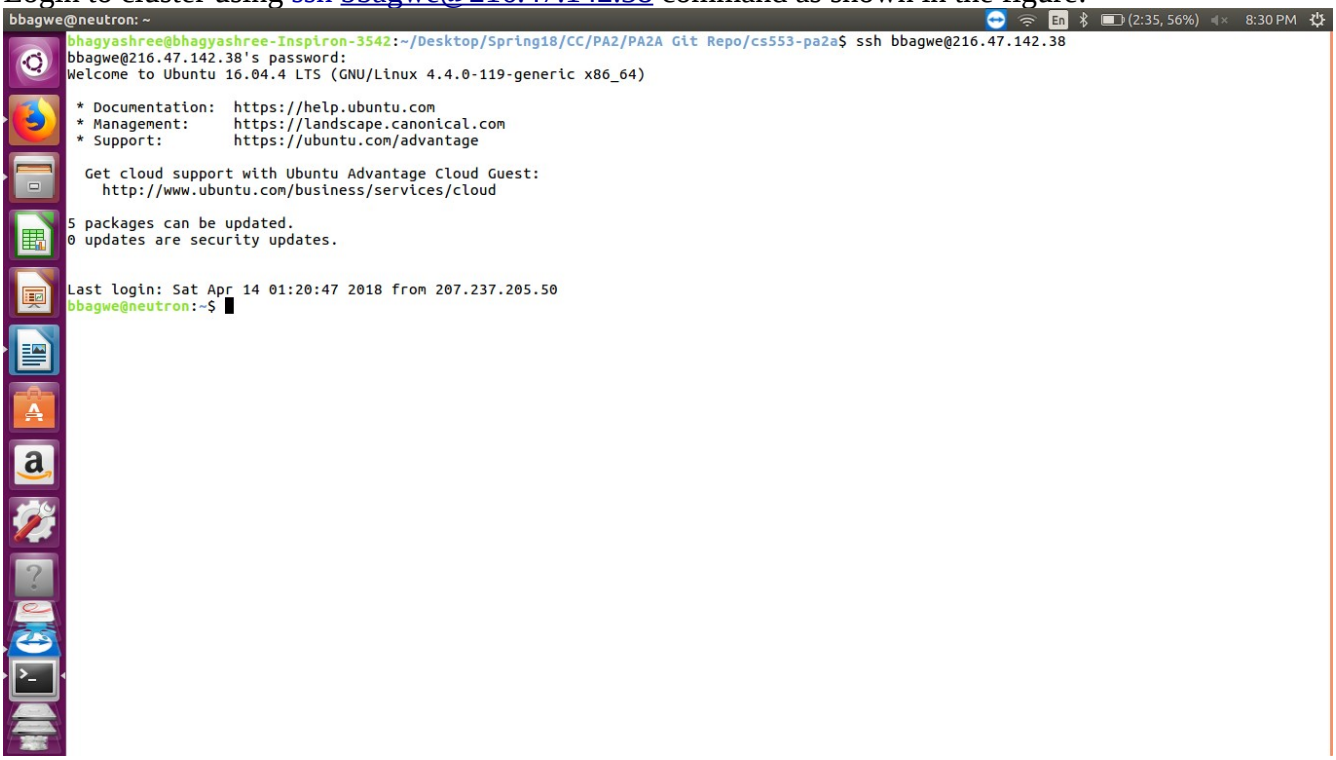
We are comparing comparing results of our experiments with UNIX sort command results.

For 2GB program, the throughput is less compared to linux sort. Since I have chosen a smaller chunk size for the available RAM, the program has to go back to disk more frequently and overall it ends up reading and writing more data from disk as compared to linux sort.

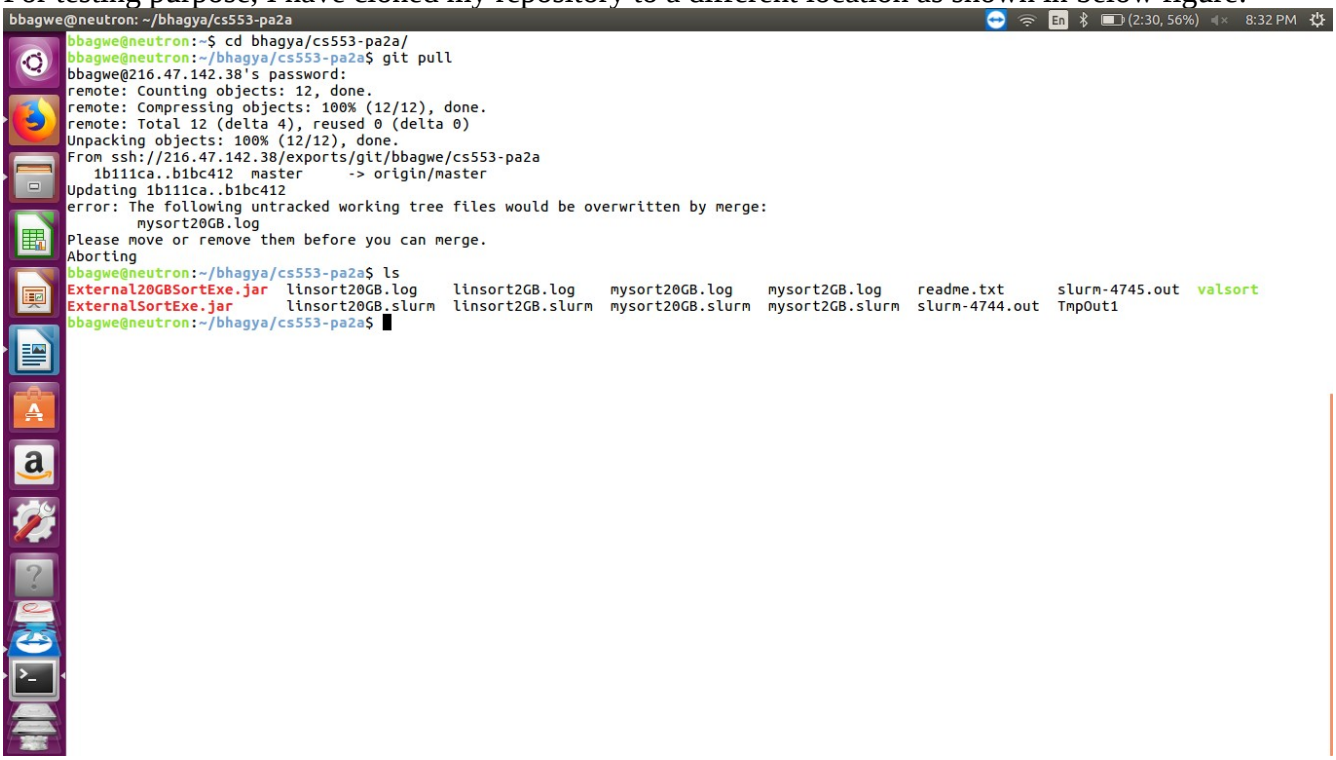
For 20GB program, since I have added an extra overhead of sending each chunk to 2GB program and merging them, even if it achieves more throughput as compared to linux sort, it compromises a lot on the response time.

Demo of execution

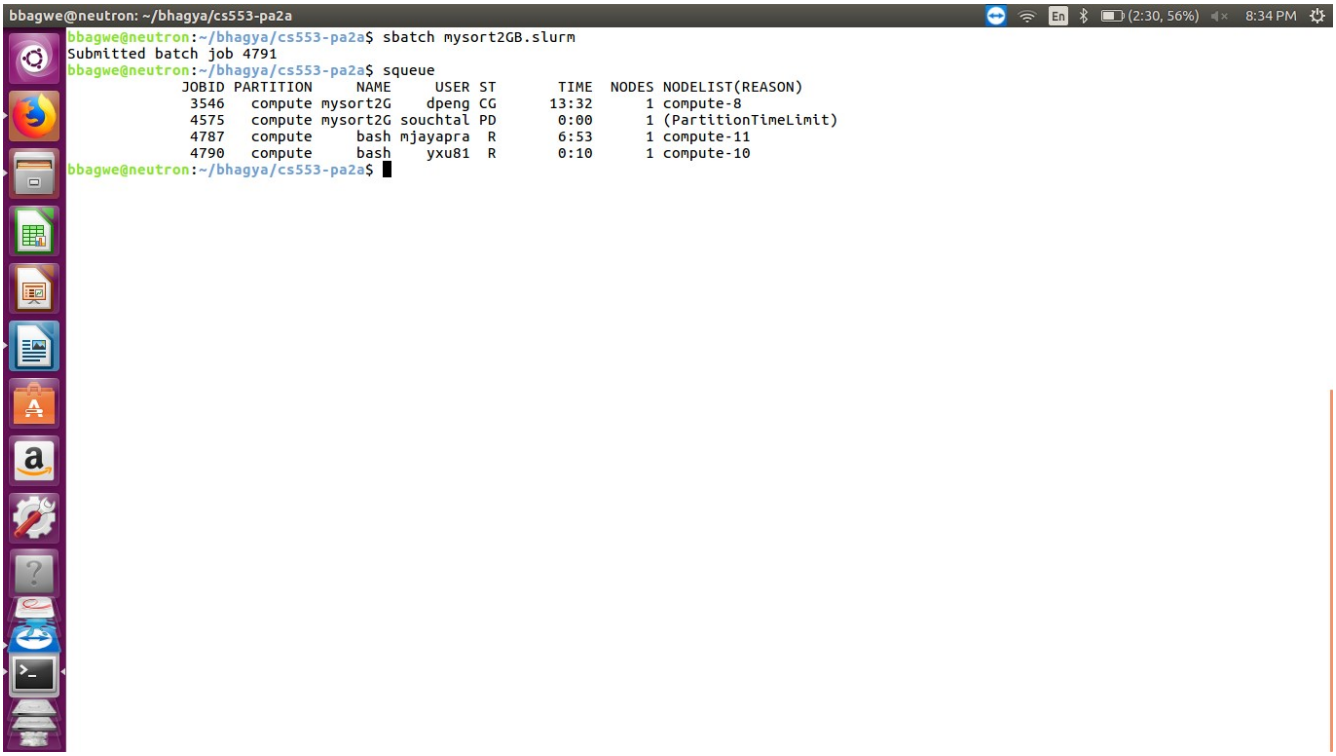
Login to cluster using ssh bbagwe@216.47.142.38 command as shown in the figure:



For testing purpose, I have cloned my repository to a different location as shown in below figure:



To execute say 2GB program, submit a slurm job using command sbatch mysort2GB.slurm. We can check status of our job using squeue command after submission as shown in the figure:



Once the job completes execution, a log file named mysort2GB.log will be generated in the same path as the executable. We use cat command to view its content.

References:

http://vkundeti.blogspot.com/2008/03/tech-algorithmic-details-of-unix-sort.html
https://pdfs.semanticscholar.org/9d8d/14d9e7cc3f05e943934e8e473461e4b89477.pdf