

CS 553 - Cloud Computing, Spring 2018

PA 2 Part A - Performance Report

Sort on Single Shared Memory Node

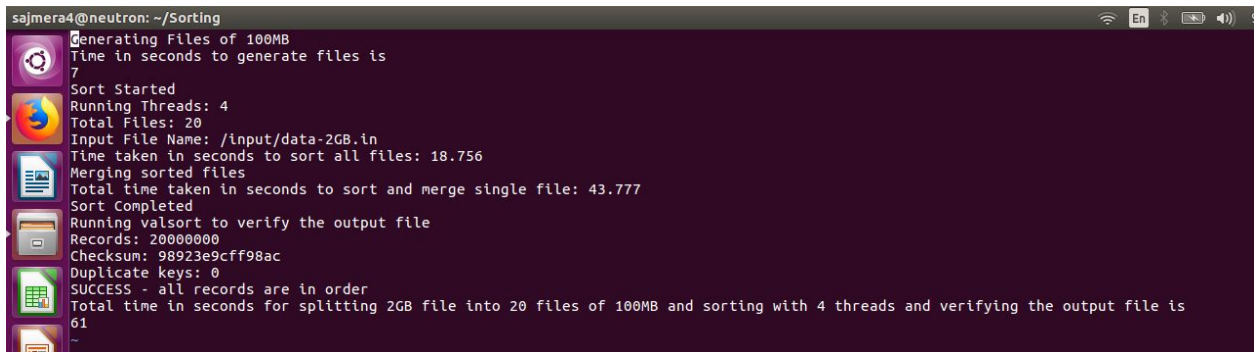
Saumil Pareshbhai Ajmera | A20397303 | sajmera4@hawk.iit.edu

1. Shared Memory Tera Sort for 2GB Data

I have implemented the code in Java, where input parameters are input file path, output file path, number of suffixes used in naming convention of small files, count of small files and threads.

Implementation-

- 1) I have divided entire file into 20 small files of 100 MB and used 4 threads in parallel to sort and finally merge the output into single file, this combination provides the best shortest time in sorting the data and achieve the best utilization of SSD storage.
- 2) By calling script *mysort2GB.slurm* will start executing mainly three tasks-
 - a) Split of files as per argument and giving file naming convention.
 - b) Calling Java executable with desired set of parameters.
 - c) Validating the output file with *valsort*.
- 3) The output log is generated by the file name *mysort2GB.log* which contains file splitting time, sorting time, sorting and merging time of entire code and entire script execution time.



```
sajmera4@neutron: ~/Sorting
Generating Files of 100MB
Time in seconds to generate files is
7
Sort Started
Running Threads: 4
Total Files: 20
Input File Name: /input/data-2GB.in
Time taken in seconds to sort all files: 18.756
Merging sorted files
Total time taken in seconds to sort and merge single file: 43.777
Sort Completed
Running valsort to verify the output file
Records: 20000000
Checksum: 98923e9cff98ac
Duplicate keys: 0
SUCCESS - all records are in order
Total time in seconds for splitting 2GB file into 20 files of 100MB and sorting with 4 threads and verifying the output file is
61
```

2. Shared Memory Tera Sort for 20GB Data

I have implemented the code in Java, where input parameters are input file path, output file path, number of suffixes used in naming convention of small files, count of small files and threads.

Implementation-

- 1) I have divided entire file into 200 small files of 100 MB and used 10 threads in parallel to sort and finally merge the output into single file, this combination provides the best shortest time in sorting the data and achieve the best utilization of SSD storage.

2) By calling script *mysort20GB.slurm* will start executing mainly three tasks-

- a) Split of files as per argument and giving file naming convention.
- b) Calling Java executable with desired set of parameters.
- c) Validating the output file with *valsort* for number of sorted records.

3) The output log is generated by the file name *mysort20GB.log* various times in seconds for file splitting time, sorting time, sorting and merging time of entire code and entire script execution time.

```
sajmera4@neutron: ~/Sorting
Generating Files of 100MB
Time in seconds to generate files is
135
Sort Started
Running Threads: 10
Total Files: 200
Input File Name: /input/data-20GB.in
Time taken in seconds to sort all files: 287.408
Merging sorted files
Total time taken in seconds to sort and merge single file: 822.809
Sort Completed
Running Valsort to verify output file
Records: 200000000
Checksum: 5f5cc94518a4203
Duplicate keys: 0
SUCCESS - all records are in order
Total time for splitting 20GB file into 200 files of 100MB and sorting with 10 threads and verifying the output file is
1042
```

3. Linsort Shared Memory Tera Sort for 2GB Data

This is the implementation of linux in built command sort which sorts data of given input file in the script *linsort2GB.slurm*

Implementation-

- 1) The sort command takes in argument of input and output file, parallel to mention parallel threads.
- 2) The output log is generated by the file name *linsort2GB.log* which contains verified records with *valsort* and time in seconds for script execution.

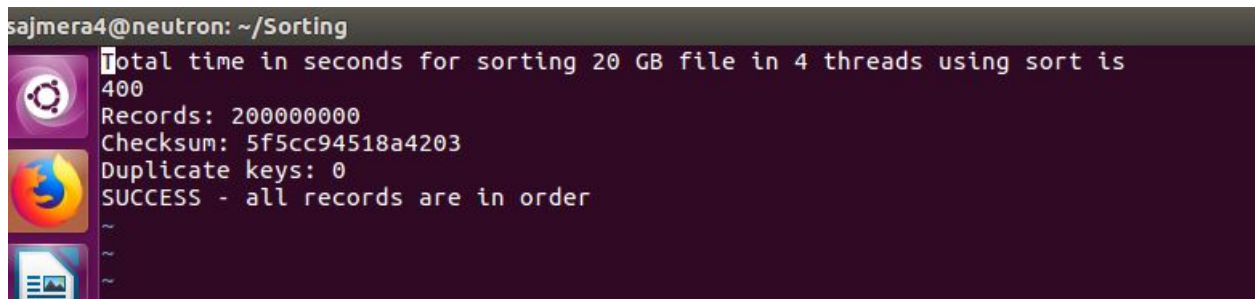
```
sajmera4@neutron: ~/Sorting
Total time in seconds for sorting 2 GB file in 4 threads using sort is
22
Records: 200000000
Checksum: 98923e9cff98ac
Duplicate keys: 0
SUCCESS - all records are in order
```

4. Linsort Shared Memory Tera Sort for 20GB Data

This is the implementation of linux in built command *sort* which sorts data of given input file in the script *linsort20GB.slurm*

Implementation-

- 1) The sort command takes in argument of input and output file, parallel to mention parallel threads.
- 2) The output log is generated by the file name *linsort20GB.log* which contains verified records with *valsort* and time in seconds for script execution.



```
sajmera4@neutron: ~/Sorting
Total time in seconds for sorting 20 GB file in 4 threads using sort is
400
Records: 200000000
Checksum: 5f5cc94518a4203
Duplicate keys: 0
SUCCESS - all records are in order
~
~
~
```

5. Comparison

Below table shows the comparison of timings and throughput of sorting done for Shared Memory and Linux Sort -

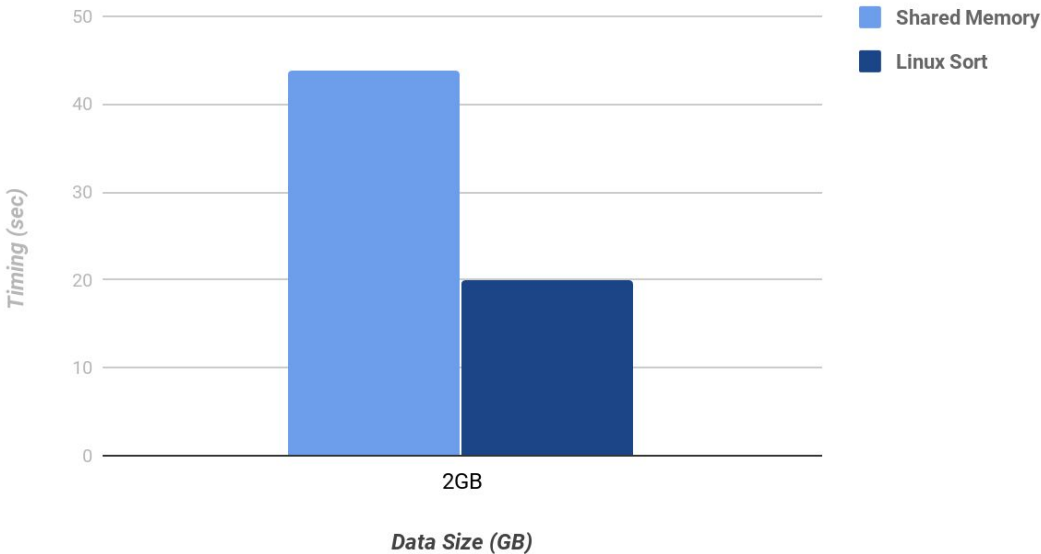
As Linux sort is written in C and its Coreutils provided by linux so it does it efficiently by handling memory and space management.

While for Shared Memory, I have written code in java so, I have tried my level best to optimize space and memory usage.

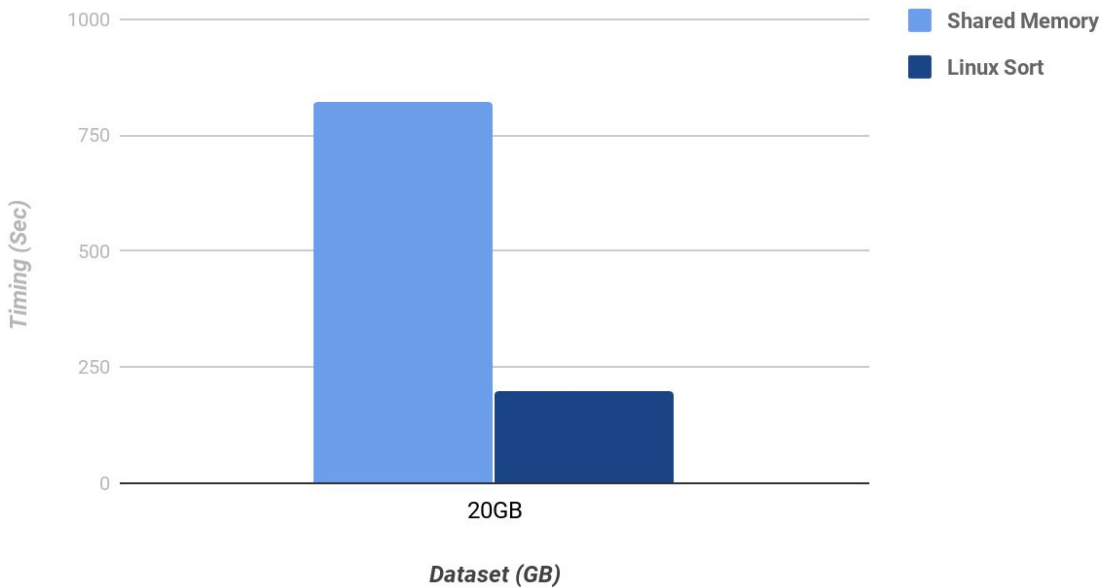
Correctness can be verified by valsort which checks for checksum and ordering which can be verified in my screenshots as both produces the same values.

Experiment	Shared Memory (1VM 2GB)	Linux Sort (1VM 2GB)	Shared Memory (1VM 20GB)	Linux Sort (1VM 20GB)
Compute Time (sec)	43.77	20	822.809	401.34
Data Read (GB)	6	2	60	40
Data Write (GB)	6	2	60	40
I/O Throughput (MB/sec)	274.160	200	145.842	199.332

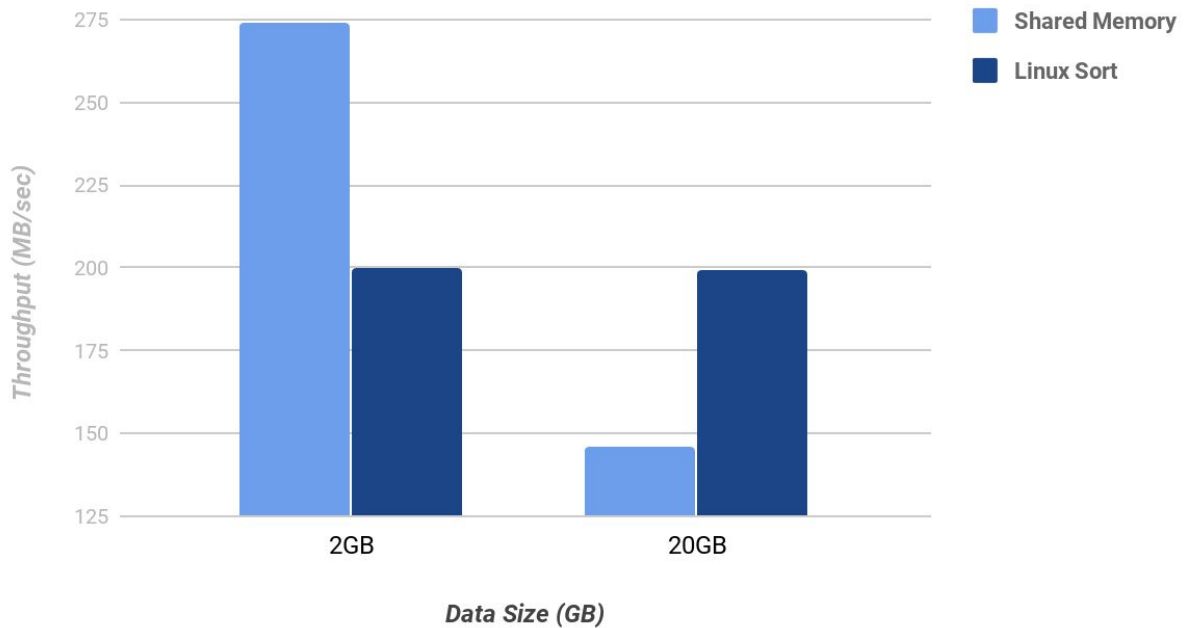
Timing for Sorting



Timing for Sorting



Throughput (MB/sec)



Here high value of throughput for 2GB data for Shared Memory than Linux sort is due to SSD performance.

6. Conclusion

Sorting can be done in memory when our data sets fits in memory. But we need to perform external sorting when our data set do not fit in memory size, so we need to divide big file into multiple files of small size and run parallel threads to sort data and then merge all sorted files into big file.

7. References

<https://stackoverflow.com/questions/13543843/random-pivot-quicksort-in-java>

<http://arnab.org/blog/quick-and-easy-multicore-sort>

<http://exceptional-code.blogspot.com/2011/07/external-sorting-for-sorting-large.html>