Design and Report Document for Sort on Single Shared Memory Node

Description of the problem:

- In this assignment we have implemented the external sorting on single shared memory node. We have performed the sorting on dataset of 2GB and 20GB using external sort program. I have use java programming using threading to compare the result with shared memory sort program.
- The data for this assignment is created using the gensort program. We have also validated the result with command using valsort program.

Design of code/Solution:

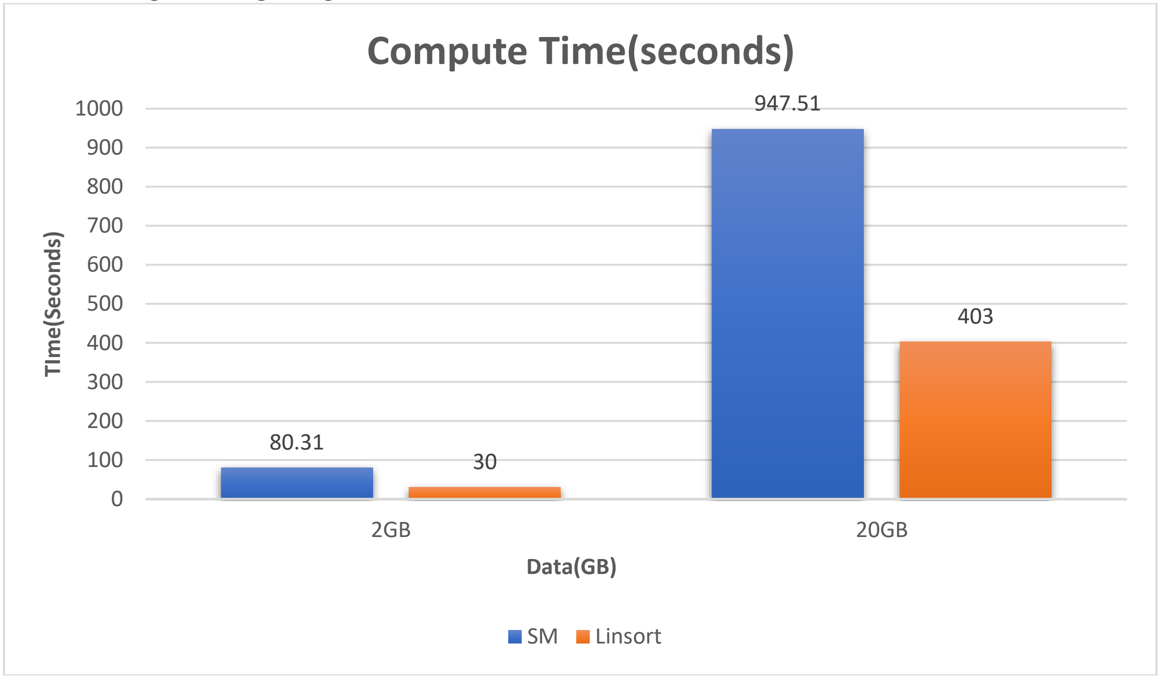
- For implementing the external sort, I have created the java class named MyExternalSort. In this class I have implemented the logic for external sorting.
- In the main function, I am assigning the values to variable like input file, output file, number of threads etc.
- Then I have called method imdSort() for creating chunk with 100 chunks for 20gb data and 10 chunks for 2gb of data. Each file will be having the same size of unsorted data. This function divides the number of chunks and intermediate sorted files are generated in the tmp folder.
- For sorting the data I have used quicksort which is implemented in imdMergeSort() function. This function will be recursively called for sorting the data. I have partitioning the data for sorting using lowerHalf() and upperHalf() method.
- Then merging the intermediate file I have used the mergesort in the finalMerge().This will read the intermediate file and will write the sorted data into the final output file.

Runtime Environment:

- Cluster Specification: Each node with 4-cores, 8GB of memory, and 80GB of SSD storage

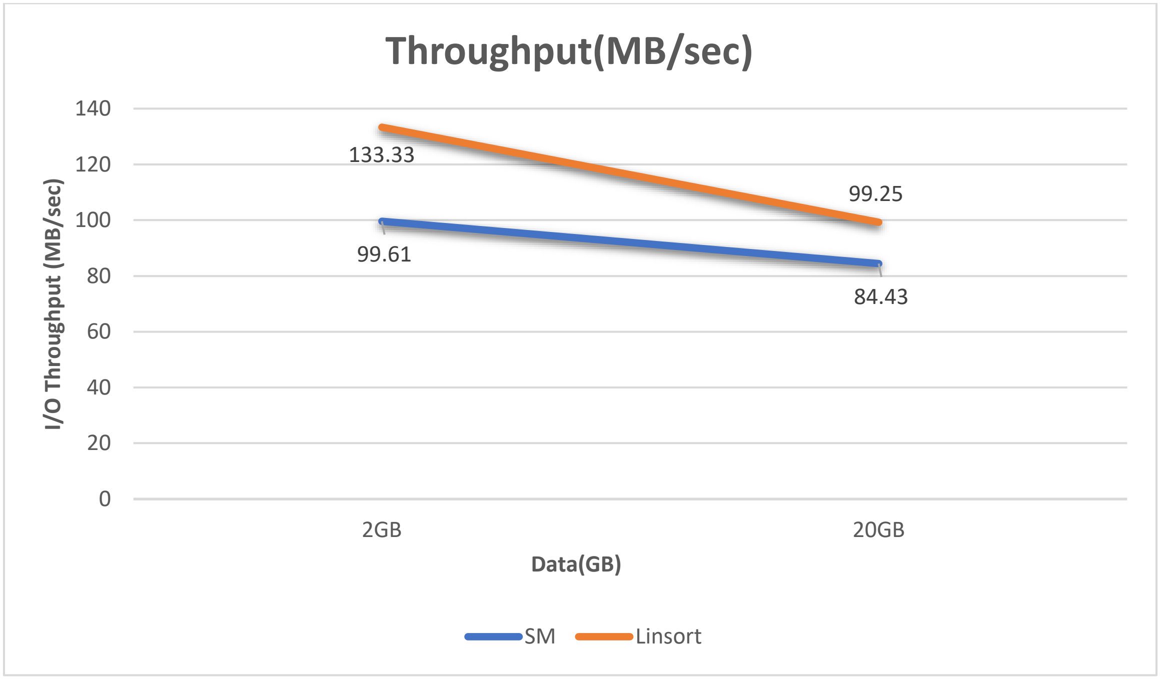
Performance Evaluation:

Experiment	**Shared Memory (1VM 2GB)**	**Linux Sort (1VM 2GB)**	**Shared Memory (1VM 20GB)**	**Linux Sort (1VM 20GB)**
Compute Time (sec)	80.312	30	947.517	403
Data Read (GB)	4	2	40	20
Data Write (GB)	4	2	40	20
I/O Throughput (MB/sec)	99.61	133.33	84.43	99.25



Conclusion:

- From graph we can conclude that as shared memory is taking more time as compared to the Linsort command.
- Also, in regards with data size 2 GB is taking less time than the 20GB which is obvious.
- Compute time will decrease the if we increase the concurrency i.e. number of threads. If there are more number of threads, then data can be divided among them and we can perform the intermediate sort operation on the concurrently which will increase the performance.

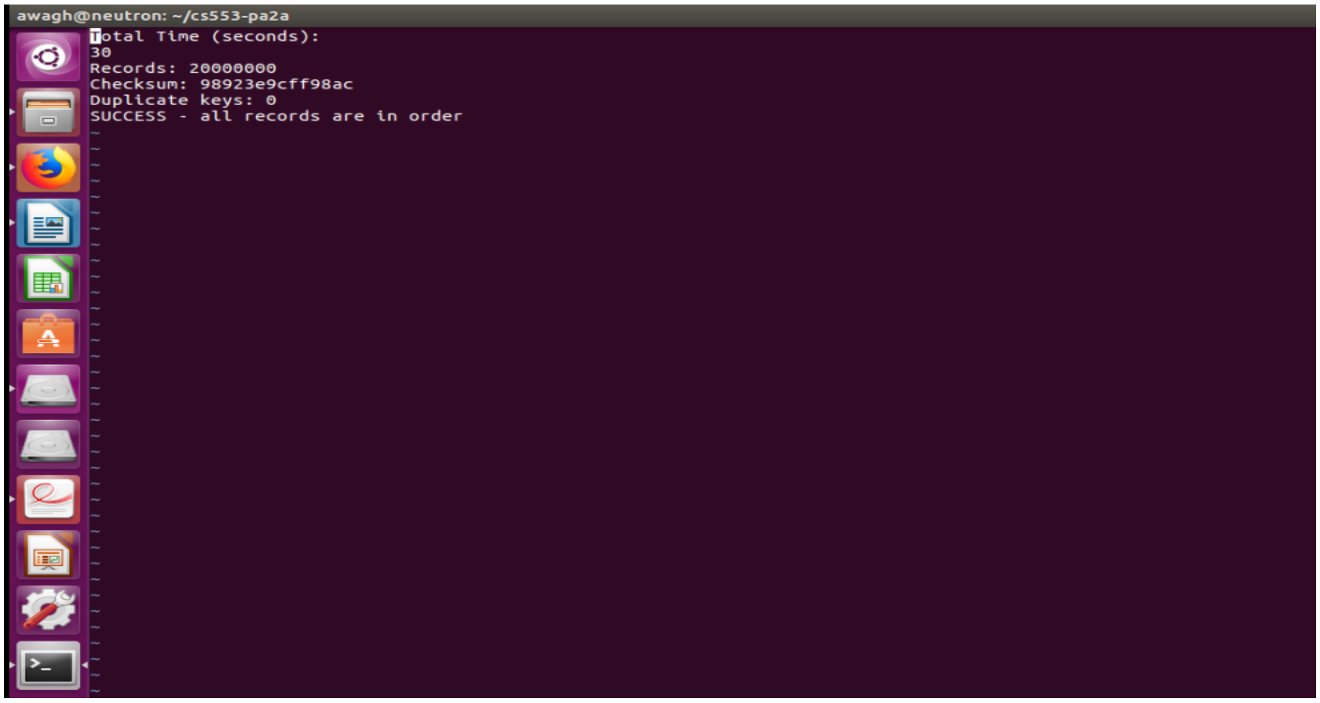


Conclusion:

- From graph we see from the graphs that throughput of the Linsort is more than shared memory.
- As data size increases the throughput values for both Linsort and Shared memory is decreasing.
- For improving the throughput, we must increase the number threads which will decrease the computation time for reading the same amount of data.

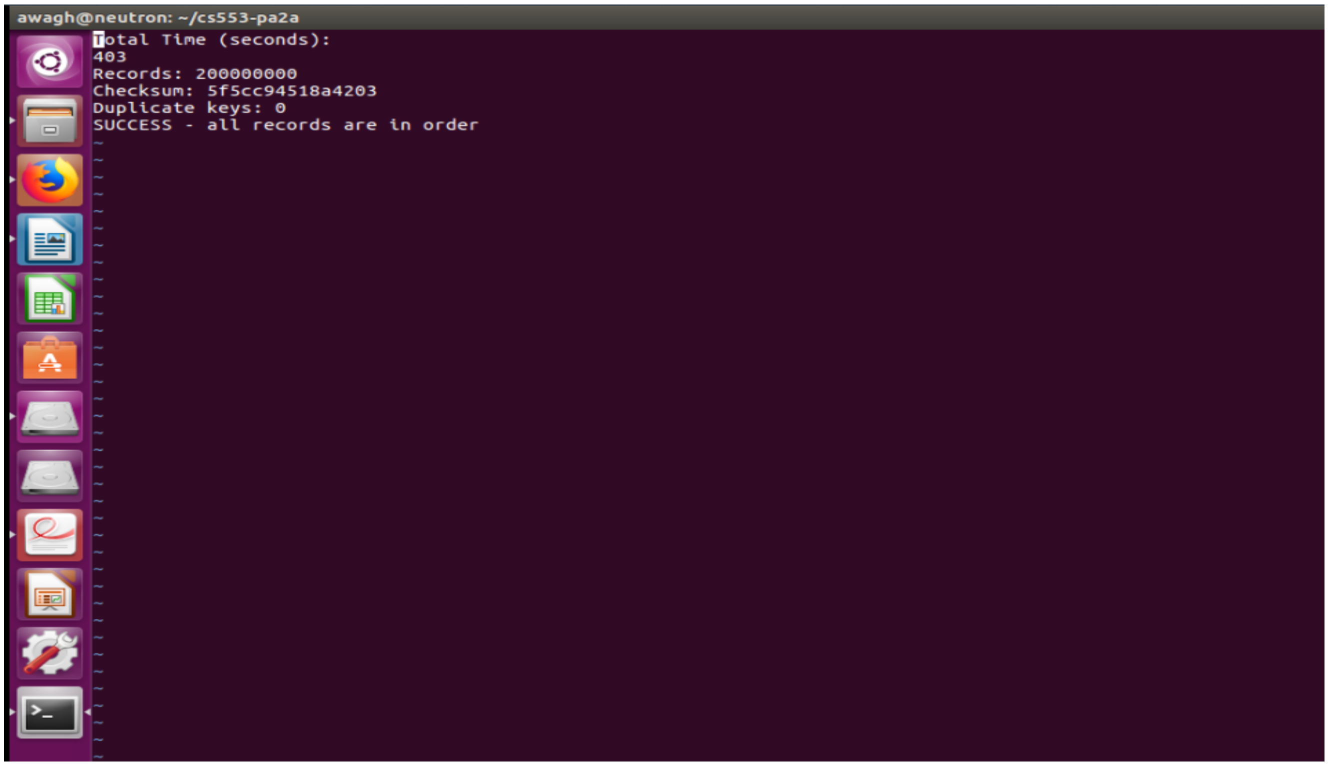
Screenshots:

1. Linsort for 2GB data:



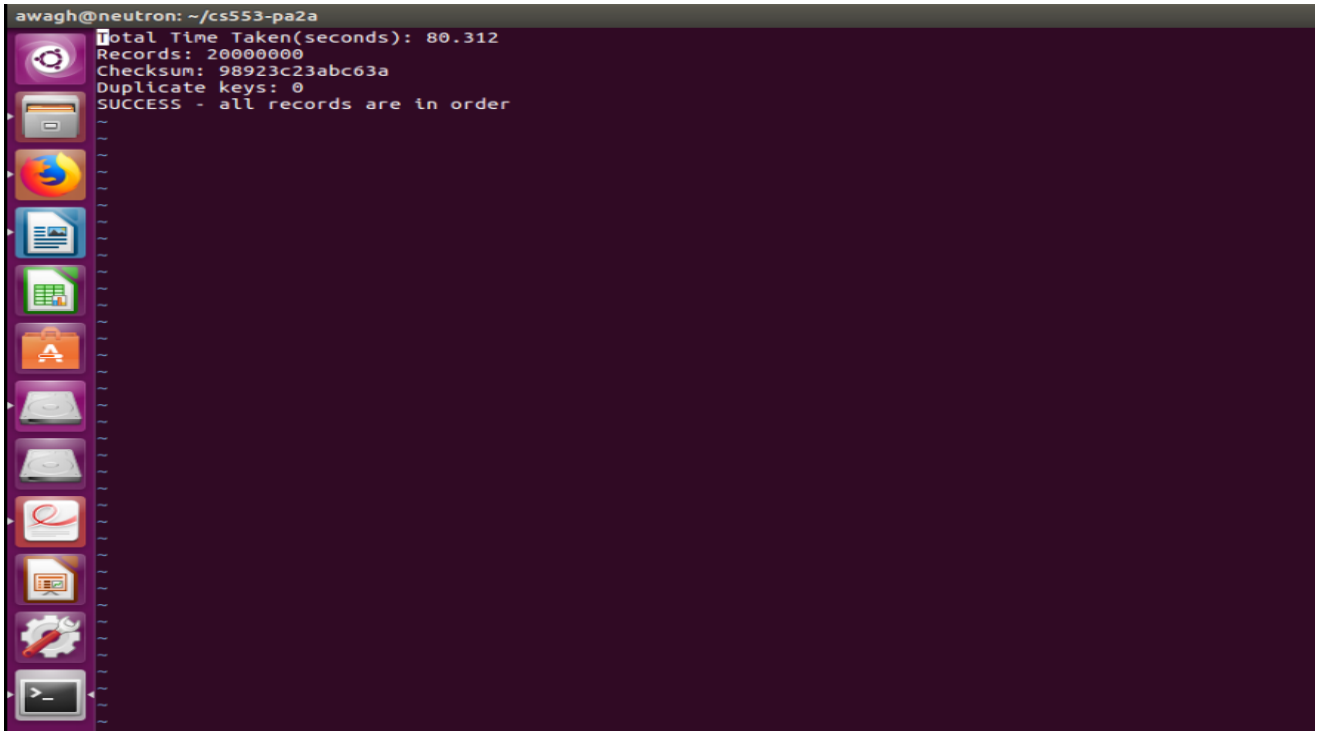
```
awagh@neutron: ~/cs553-pa2a
Total Time (seconds):
30
Records: 20000000
Checksum: 98923e9cff98ac
Duplicate keys: 0
SUCCESS - all records are in order
```

2. Linsort for 20GB data:



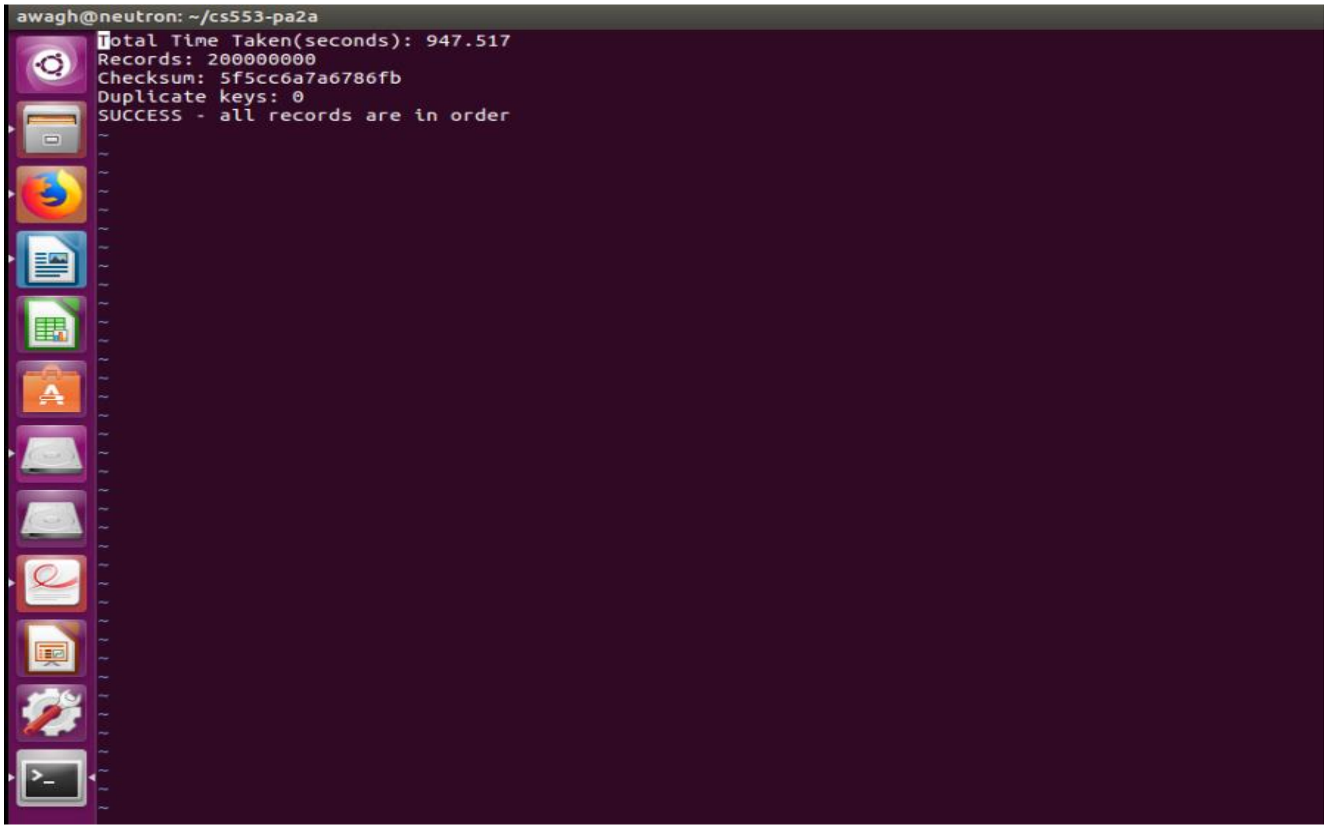
```
awagh@neutron: ~/cs553-pa2a
Total Time (seconds):
403
Records: 200000000
Checksum: 5f5cc94518a4203
Duplicate keys: 0
SUCCESS - all records are in order
```

3. Mysort for 2GB data:



```
awagh@neutron: ~/cs553-pa2a
Total Time Taken(seconds): 80.312
Records: 20000000
Checksum: 98923c23abc63a
Duplicate keys: 0
SUCCESS - all records are in order
```

4. Mysort for 20GB data:



```
awagh@neutron: ~/cs553-pa2a
Total Time Taken(seconds): 947.517
Records: 200000000
Checksum: 5f5cc6a7a6786fb
Duplicate keys: 0
SUCCESS - all records are in order
```