CS 553 Cloud Computing
Homework 5

- Problem

  Implement the Shared-Memory TeraSort application for four datasets of different sizes : 1GB, 4GB,16GB, 64GB. Memory usage is limited to 8 GB.

- Methodology
  - For 1GB and 4 GB of dataset we use in- memory sort
  - For 16 GB and 64 GB we use external sort
  - We use parallel merge sort for all four data sets.
  - And use k -way merge for external sort we combine the result into a single file.
- Runtime Environment
  - Compute node: Skylake node
  - C++
  - Linux
  - Memory : 8 GB

- Overall benchmark design
  - MySort.cpp is the main file
  - Input to the file includes no. of threads and dataset file.
    - ./MySort 24 record_1gb.txt
  - getFileSize() function is used to get the size of the file on bytes.
  - If the size of the file is less than 8GB we call internal sort else, we perform external sort.
  - For in-memory sort
    - First we read the data from the file and store it in a vector using readData() function
    - Once the data is read we pass the vector and the thread count to inplace_merge_sort() which performs parallel merge sort.
    - inplace_merge_sort() : we divide the dataset among threads depending on the no. of threads given by the user.
    - Each thread performs merge sort on its own data and returns the result.
    - Then we merge the result and the final sorted data is written into the output file using writeData().
  - For external sort (Max Memory Size = 8GB)
    - We decide the no. of chunks the file has to be divided into using the formula:
      - Chunks = FileSize / Max Memory Size
    - Once the chunks are decided then we calculate the read and write buffer size.
      - Read buffer = Max Memory Size / Chunks +1
      - Write buffer = Max Memory Size – (Read Buffer * Chunks)
    - “counter” variable and memory_utilization_check() function keeps track of the memory being allocated and deallocated to make sure that we don’t exceed the given size.
    - Threads are created equal to the no. of chunks.
    - Each thread will read the max size allowed from the file and write it to another file.

- Once the file is divided into chunks, we sequentially sort each chunk using inplace merge sort.
- After all the files are sorted, we call the kway merge to create the final sorted file.
- For k-way merge, MinHeap is used to store min element of each file.

```
struct MinHeapNode
{
   string line;
   int i;
};
```

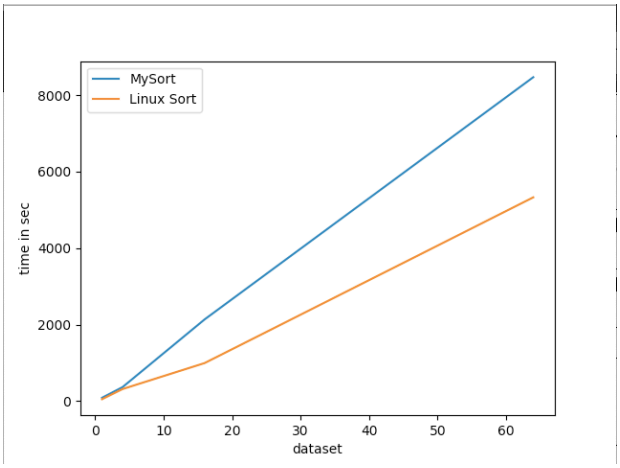
- All the input files are opened, then we store the min element from each file into array of MinHeap.
- Priority queue is used to compare the elements in min heap and store them in ascending order
- Then we pop the top element from the queue and write it to the file.
- Then we continue reading the next min element from all the files and store it queue.
- We perform the above steps until all the data from the files are read.
- The final output file is created with sorted data.

➢ Performance evaluation of Single Node TeraSort (using best # of threads for each case)

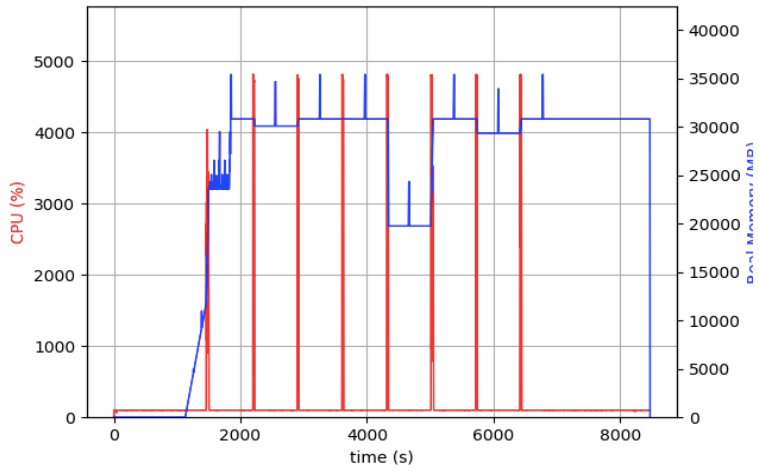
Experiment	**Shared Memory (1GB)**	**Linux Sort (1GB)**	**Shared Memory (4GB)**	**Linux Sort (4GB)**	**Shared Memory (16GB)**	**Linux Sort (16GB)**	**Shared Memory (64GB)**	**Linux Sort (64GB)**
Number of Threads	24	24	24	24	48+2(IO)	48	48+8(IO)	48
Sort Approach (e.g. in-memory / external)	In-memory	External	In-memory	External	External	External	External	External
Sort Algorithm (e.g. quicksort / mergesort / etc)	Merge-sort	Merge-sort	Merge-sort	Merge-sort	Merge-sort	Merge-sort	Merge-sort	Merge-sort
Data Read (GB)	1	1	4	4	3	2.58	0.89	2.58
Data Write (GB)	1	1	4	4	2	2.58	0.89	2.58
Sort Time (sec)	87.21	50.03	366.796	315.89	2138.51	995.92	8469.04	5328.08
Overall I/O Throughput (MB/sec)	11.74	20.46	11.167	12.97	7.66	16.45	7.74	12.3
Overall CPU Utilization (%)	88.24	55.6	92.3	61.4	99.6	98.7	100	99.7
Average Memory Utilization (GB)	0.13	0.13	1.71	1.14	4.08	1.20	4.27	1.21

- Results
  - For 1GB and 4GB dataset MySort performs in-memory merge sort with throughput of 11.74 and 11.16 respectively. Whereas Linux sort has a throughput of 20.46 and 12.97 respectively.
  - Sort time and Average memory utilization is similar for both MySort and Linux Sort for 1GB and 4 GB dataset.
  - For 16 GB and 64 GB dataset MySort takes more time as compared to Linux sort and throughput is also less. MySort uses external merge sort along with k-way merge to sort the data.
  - Average memory utilization is greater for MySort as compared to Linus Sort for 16GB and 64 GB dataset.

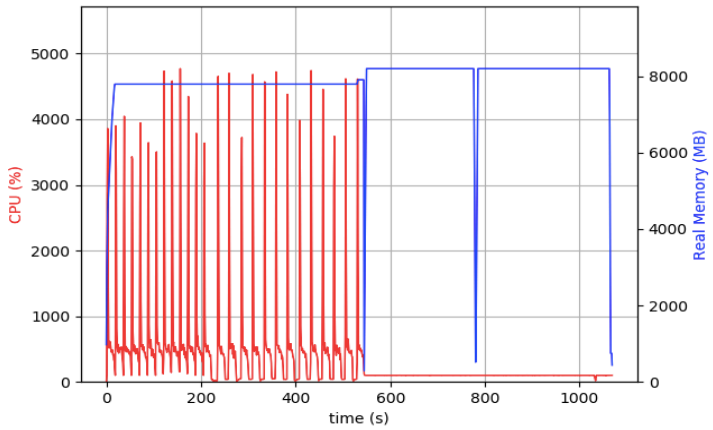
- Graph for Dataset vs Sort time for both sorts



- MySort 64GB graph



➢ Linux Sort 64GB graph



- For MySort, CPU utilization spikes when sorting each chunk of the file and merging the files.
- The memory utilization is also high compared to Linux Sort.

➢ Logs

- **MySort 1GB**

  Data Read (GB): 1

  Data Write (GB): 1

  Sort Time (sec): 87.21

  Throughput(MB/sec): 11.74

  ./valsort record_1GB_sorted.txt
  Records: 10737418
  Checksum: 51e82f1348dc5f
  Duplicate keys: 335752
  SUCCESS - all records are in order

- **Linux Sort 1GB**

  Data Read (GB): 1

  Data Write (GB): 1

  Sort Time (sec): 50.03

  Throughput(MB/sec): 20.46

  ./valsort output_1GB.txt

Records: 10737418
Checksum: 51e88faefe7b34
Duplicate keys: 0
SUCCESS - all records are in order

- **MySort 4GB**

  Data Read (GB): 4

  Data Write (GB): 4

  Sort Time (sec): 366.796

  Throughput(MB/sec): 11.167

  ./valsort record_4GB_sorted.txt
  Records: 42949672
  Checksum: 147a6e74151eb00
  Duplicate keys: 2426680
  SUCCESS - all records are in order

- **Linux Sort 4GB**

  Data Read (GB): 4

  Data Write (GB): 4

  Sort Time (sec): 315.89

  Throughput(MB/sec): 12.97

  ./valsort output_4GB.txt
  Records: 42949672
  Checksum: 147a91dd0ea4d36
  Duplicate keys: 0
  SUCCESS - all records are in order

- **MySort 16GB**

  Data Read (GB): 3

  Data Write (GB): 2

  Sort Time (sec): 2138.51

  Throughput(MB/sec): 7.66

  ./valsort output_sorted.txt
  Records: 171798690

Checksum: 51ecbbabb0942db
Duplicate keys: 86040842
SUCCESS - all records are in order

- **Linux Sort 16GB**

  Data Read (GB): 2.58

  Data Write (GB): 2.58

  Sort Time (sec): 995.92

  Throughput(MB/sec): 16.45

  ./valsort output_16GB.txt
  Records: 171798691
  Checksum: 51ea5e584e0bcd3
  Duplicate keys: 0
  SUCCESS - all records are in order

- **MySort 64GB**

  Data Read (GB): 0.89

  Data Write (GB): 0.89

  Sort Time (sec): 8469.04

  Throughput(MB/sec): 7.74

  ./valsort output_sorted.txt
  Records: 687194760
  Checksum: 147abb9423e3f49d
  Duplicate keys: 601295497
  SUCCESS - all records are in order

- **Linux Sort 64GB**

  Data Read (GB): 2.58

  Data Write (GB): 2.58

  Sort Time (sec): 5328.08

  Throughput(MB/sec): 12.3

  ./valsort output_64GB.txt
  Records: 687194767

Checksum: 147ad98a21f6e7a7
Duplicate keys: 0
SUCCESS - all records are in order