Internal and External Sort (small file vs large file computations)

Ali Guzelyel

Oladapo Ogunnaike

Readme has a good explanation on how to run the code. Package dependencies are also in there. We have used psrecord to monitor cpu and memory usage, and sysstat-pidstat to monitor disk I/O. Tasks run for long durations, in order to not have broken pipe error, we run the tasks with nohup in the background. Also, we had to do separate test for CPU-MEM and Disk I/O because pidstat and psrecord we not compatible.

We have implemented the mysort algorithm with internal and external sort. Our implementation depends on malloc-buffer to allow memory allocation, and a few custom typedefs, and thread pools. We have used quicksort as our main algorithm since mergesort is going to be taking a lot of space, and disk I/O is going to be computationally expensive. We have also utilized Chameleon - compute_haswell_ib node to run our tests on 1GB, 4GB, 16GB and 64GB data. Since the node has 128GB memory, we limited the maximum memory usage to 8GB. Screenshot has been added to the end showing the node specs.

We have tested our program with multiple num_threads, and included the best resulting to the table. Similarly, we tested linux-sort with the same num_threads and included them to the table.

During testing, the chameleon instance node failed / got deleted (seemed to have happened to other users too), and we lost most of the data. So we had to run the same tests twice, but this time we omitted the test cases that performed badly. (ex. 16GB with 4 threads was way slower than the rest, so omitted that.) Additionally, some logs of the *valsort* were also lost, but we included the screenshots confirming that mysort is able to do perfect sort.

Here is the table that shows our experiment (all experiments logs are archived in psrecord_analysis_arch)

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	8	16	4	8	8	24	16	24
Sort Approach (internal / external)	Internal - in-memory	Internal - in-memory	Internal - in-memory	Internal - in-memory	External	Internal	External	External

Sort Algorithm (quicksort / mergesort)	quicksort	quicksort	quicksort	quicksort	quicksort	quicksort	quicksort	quicksort
Data Read (GB)	0.125 GB per thread	0.0625GB per thread	1GB per thread	0.5GB per thread	1GB per thread	0.33GB per thread	0.5GB per thread	0.33GB
Data Write (GB)	0.125 GB per thread	0.0625GB per thread	1GB per thread	0.5GB per thread	1GB per thread	0.33GB per thread	0.5GB per thread	0.33GB per thread
Sort Time (sec)	17.108	17.350	54.560	95.033	198.938	708.96	4076.221	4428.30
Overall I/O Throughput (MB/sec)	59.855	59.020	75.073	43.101	82.358	23.11	16.078	14.799
Overall CPU Utilization (%)	82.22	110.47	99.08	69.68	221.22	86.14	68.79	67.12
Average Memory Utilization (GB)	1.092	0.508	4.470	1.331	8.251	2.537	3.235	6.801

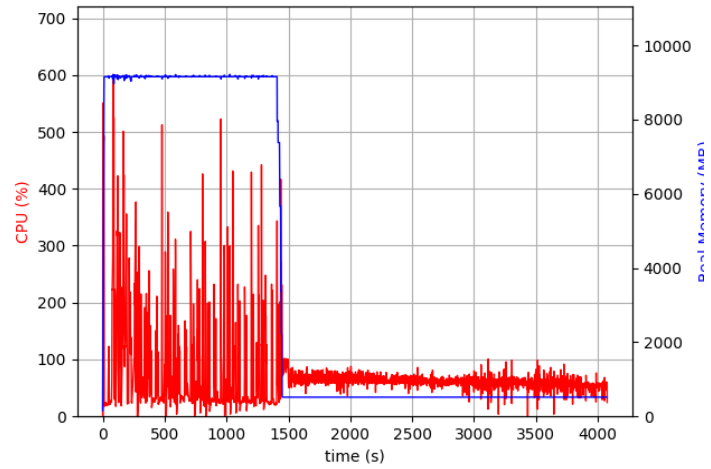
Table explanations:

First off, we can see that mysort performed better than linux sort on almost all test cases. This may be because of the disk vs memory specs of the machine, as I still assume our program might not be more optimized than the standard linux sort. The throughput gets smaller as the size of the file increases as we are dealing more with disk I/O than just memory. For internal sort, we are using only memory, so it's given that we would get a better throughput. For external sort, we are using the disk as well, so the throughput gets smaller as file size increases. Additionally, CPU utilization shows how much of the CPU power is used by the multi-threaded program. Average memory utilization shows that the mysort on 64GB could have performed better if the code for disk - memory was optimized better. However, our result is similar in scalar.

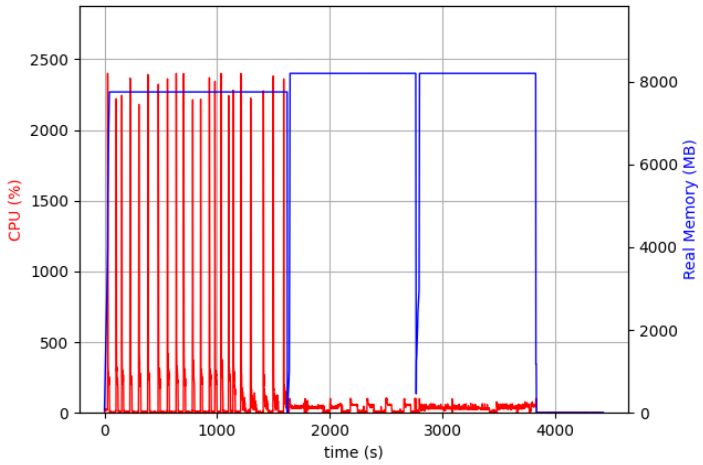
CPU - Memory - Disk I/O on 64GB data: plots

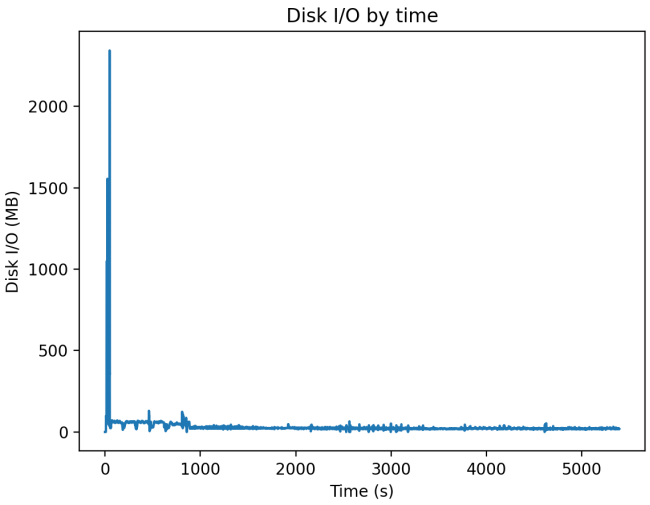
CPU - Memory - Disk I/O analysis plot for 64GB data (mysort on left column, linux-sort on right column. First row is for CPU(%) and Real Memory (MB) vs Time(s), second row is for Disk I/O.

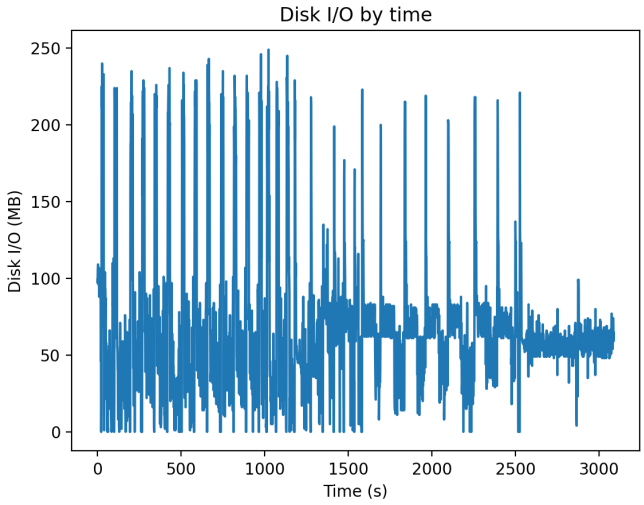
Mysort 16 threads (CPU-MEM-Disk I/O)



Linux-Sort 24 threads(CPU-MEM-Disk I/O)







Analysis: Looking at the graphs, we can see that linux-sort utilized memory a lot better than mysort (almost double average). Additionally, we can see that the CPU was used a lot more efficiently in linux-sort compared to mysort. The Disk I/O test on linux-sort shows a very good utilization of the disk, compared to mysort. We are also curious why mysort Disk I/O did not show spikes in the other parts of the run like linux-sort, which is based on our implementation. One other curious matter is how mysort outperforms linux-sort despite linux-sort being more optimized.

VALSORT screenshots for mysort accuracy (both for internal and external algorithms):

```
Starting up command './mysort 4GB.txt sort.txt 4 >> mysort4GB.log' and attaching to process
Process finished (104.33 seconds)
Records: 40000000
Checksum: 1312774ebf75c93
Duplicate keys: 0
SUCCESS - all records are in order
Starting up command './mysort 4GB.txt sort.txt 8 >> mysort4GB.log' and attaching to process
Process finished (77.71 seconds)
Records: 40000000
Checksum: 1312774ebf75c93
Duplicate keys: 0
SUCCESS - all records are in order
Starting up command './mysort 4GB.txt sort.txt 16 >> mysort4GB.log' and attaching to process
Process finished (69.58 seconds)
Records: 40000000
Checksum: 1312774ebf75c93
Duplicate keys: 0
SUCCESS - all records are in order
Starting up command './mysort 4GB.txt sort.txt 32 >> mysort4GB.log' and attaching to process
Process finished (78.98 seconds)
Records: 40000000
Checksum: 1312774ebf75c93
Duplicate keys: 0
SUCCESS - all records are in order
Starting up command './mysort 4GB.txt sort.txt 48 >> mysort4GB.log' and attaching to process
Process finished (78.81 seconds)
Records: 40000000
Checksum: 1312775f90d7762
Duplicate keys: 15
SUCCESS - all records are in order
Starting up command './mysort 16GB.txt sort.txt 4 >> mysort16GB.log' and attaching to process
Process finished (357.21 seconds)
Records: 160000000
Checksum: 4c4a5075594b41e
Duplicate keys: 5
SUCCESS - all records are in order
```

Chameleon node specs:

c01-08 compute_haswell_ib

Site: tacc	**Platform Type:** x86_64	**# CPUS:** 2	**# Threads:** 48
RAM Size: 128 GiB	**Total Storage:** 200-400GB		

Processor

Cache L1:	**Cache L1 D:** 32768	**Cache L1 I:** 32768	**Cache L2:** 262144
Cache L3: 31457280	**Clock Speed:** 3100000000	**Instruction Set:** x86-64	**Model:** Intel Xeon
Other Description: Intel(R) Xeon(R) CPU E5-2670 v3 @ 2.30GHz	**Vendor:** Intel	**Version:** E5-2670 v3	

SSD

Device: sda	**Driver:** megaraid_sas	**Humanized Size:** 250 GB	**Interface:** SATA
Model: ST9250610NS	**Rev:** AA63	**Size:** 250059350016	**Vendor:** Seagate
SSD: No			