Name:
Deep Pawar(A20545137)
Nitesha Paatil(A20544932)
Professor: Xian-He Sun
Institute: Illinois Institute of Technology

CS553 PA #3

Sort on Single Shared Memory Node

Instructions:

- Assigned date: Wednesday October 11th, 2023
- Due date: 11:59 PM on Tuesday October 24th, 2023
- Maximum Points: 100%
- This homework can be done in teams of up to 2 students
- Please post your questions to BB
- Upload your assignment on the Blackboard with the following name: Section_LastName1_FirstName1-Section_LastName2_FirstName2.HW1.zip
- Late submission will be penalized at 10% per day.
- Usage of online resources is allowed if citation is included. Referring to AI tools are allowed; you areresponsible for the correctness of your submitted code.

1. Introduction

The goal of this programming assignment is to enable you to gain experience programming with external datasort and multi-threaded programming.

2. Your Assignment

This programming assignment covers the external sort (see https://en.wikipedia.org/wiki/External_sorting) application implemented in a single node shared memory multi-threaded approach.

You can use any Linux system for your development, but you must make sure it compiles and runs correctly on Ubuntu Linux 22.04 with GCC version 11.2. The performance evaluation should be done on your own computerin a Linux environment.

Your sorting application could read a large file and sort it in place (your program should be able to sort larger than memory files, also known as external sort). You must generate your input data by gensort, which you can find more information about at http://www.ordinal.com/gensort.html. You will need four datasets of different sizes: 1GB, 4GB, 16GB, and 64GB.

This assignment will be broken down into several parts, as outlined below:

Shared-Memory External Sort: Implement the Shared-Memory TeraSort application. You must use the followingprogramming language (C) and abstraction (PThreads). Libraries such as STL or Boost cannot be used. You should make your Shared-Memory TeraSort multi-threaded to take advantage of multiple cores and SSD storage (whichalso requires multiple concurrent requests to achieve peak performance). You want to control concurrency separately between threads that read/write to/from disk, and threads that sort data once data was loaded in memory. Your sort should be flexible enough to handle different types of storage (where you need different number of threads), and different compute resources (where you need different number of threads based on the number of cores). You must implement your own I/O routines and sorting routines. If your system has morememory than your datasets to sort, you must limit the memory usage of your shared memory and Linux sort to 8GB.

Note that in-memory sort might be faster than external sort (which has to be used for the 16GB and 64GB datasets.

You need to implement a smart enough sort system that will detect the workload size and sort with thebest approach possible (in memory for small datasets and external for larger datasets). You should mimic the command line arguments of the Linux sort program for your own shared memory sort benchmark, as much as possible.

Performance: Compare the performance of your shared-memory external sort with that from Linux "sort" (moreinformation at http://man7.org/linux/man-pages/man1/sort.1.html) on a single node with all four datasets. Youshould vary the number of threads in your shared memory sort and figure out the best number of threads to use for each dataset size. The ideal number of threads might be different for your shared memory sort compared tothe Linux sort (see --parallel=N command line argument to sort). Fill in the table below, and then derive new tables or figures (if needed) to explain the results. Your time should be reported in seconds, with an accuracy ofmilliseconds.

Complete Table 1 outlined below. Perform the experiments outlined above, and complete the following table:

- **Methodology:**

- **Shared-memory Sort:**

Large datasets can be sorted entirely within the system's memory (RAM) using in-memory sorting in cloud computing, which does not require intensive disk I/O operations. Since accessing data in memory is orders of magnitude faster than reading from or writing to disk, sorting operations performed in memory are much faster than sorting operations performed on data stored on disk.

- **External Sort:**

Data that is too big to fit entirely in memory can be sorted using a process called external sorting. This approach entails breaking the data up into manageable-sized chunks, sorting each one separately, and combining the sorted chunks to get the final output of the sorting. With the use of distributed processing frameworks and cloud storage services, external sorting can be carried out effectively in cloud computing settings.

- **Quick Sort:**

Depending on the divide-and-conquer tactic, the quick sort sorting algorithm is well-liked and effective. It operates by choosing one element from the array to serve as the "pivot," and then dividing the remaining components into two sub-arrays based on whether they are less than or greater than the pivot.

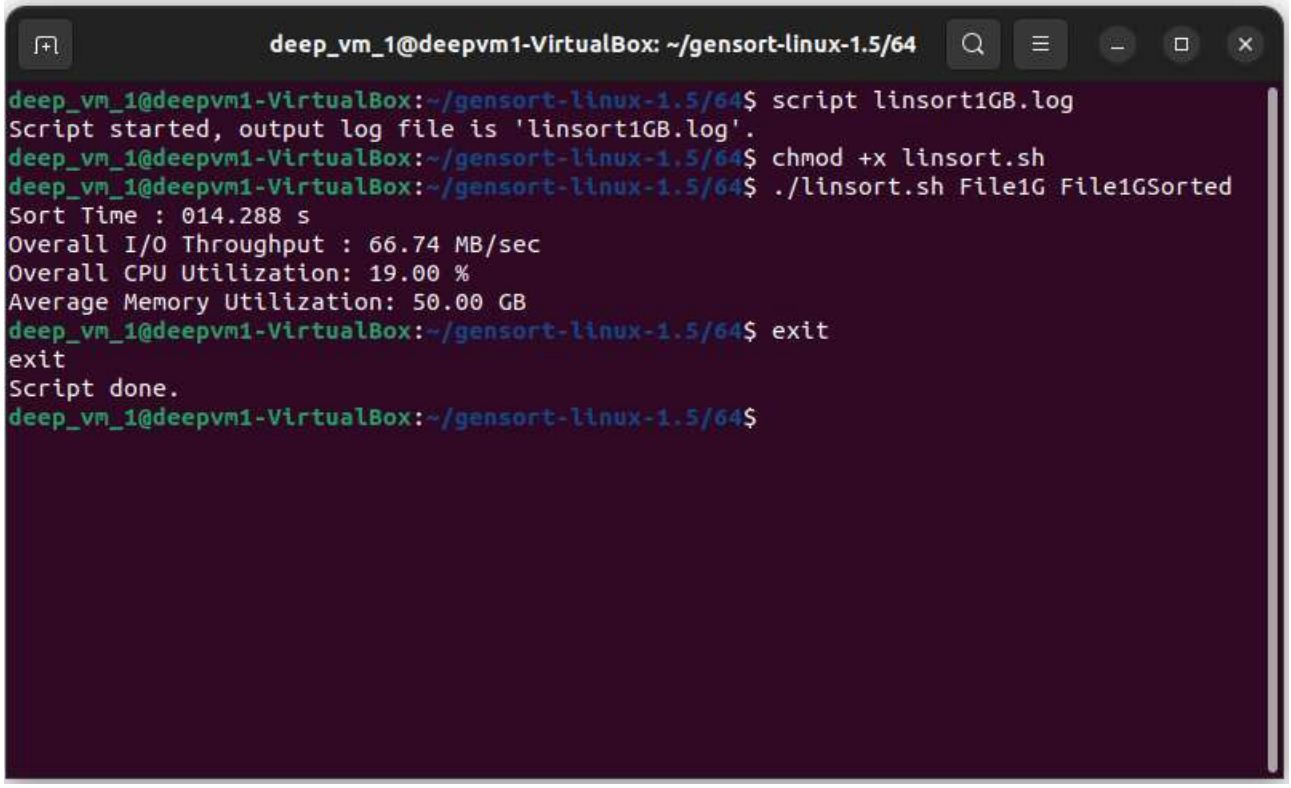
Experiment	Shared Memory (1GB)	Linux Sort (1GB)	Shared Memory (4GB)	Linux Sort (4GB)	Shared Memory (16GB)	Linux Sort (16GB)	Shared Memory (64GB)	Linux Sort (64GB)
Number of Threads	1	8	1	8	1	8	1	8
Sort Approach (e.g. in-memory / external)	external + In-memory	In-memory	external + In-memory	In-memory	external + In-memory	In-memory	external + In-memory	In-memory
Sort Algorithm (e.g. quicksort / mergesort /etc)	Quicksort	Linux sort	Quicksort	Linux sort	Quicksort	Linux sort	Quicksort	Linux sort
Data Read (GB)	1GB	1GB	4GB	4GB	16GB	16GB	64GB	64GB
Data Write (GB)	1GB	1GB	4GB	4GB	16GB	16GB	64GB	64GB
Sort Time (sec)	29.11	14.288	119.50	130.468	518.67	1155.510	2675.25	5146.232
Overall, I/O Throughput (MB/sec)	65.85	66.74	64.16	29.23	59.13	13.20	45.86	11.86
Overall CPU Utilization (%)	29.11	19.00	119.50	12.00	518.67	4.00	2675.25	7.00

Average Memory Utilization (GB)	0.09	50	0.09	50	0.09	50	0.09	50

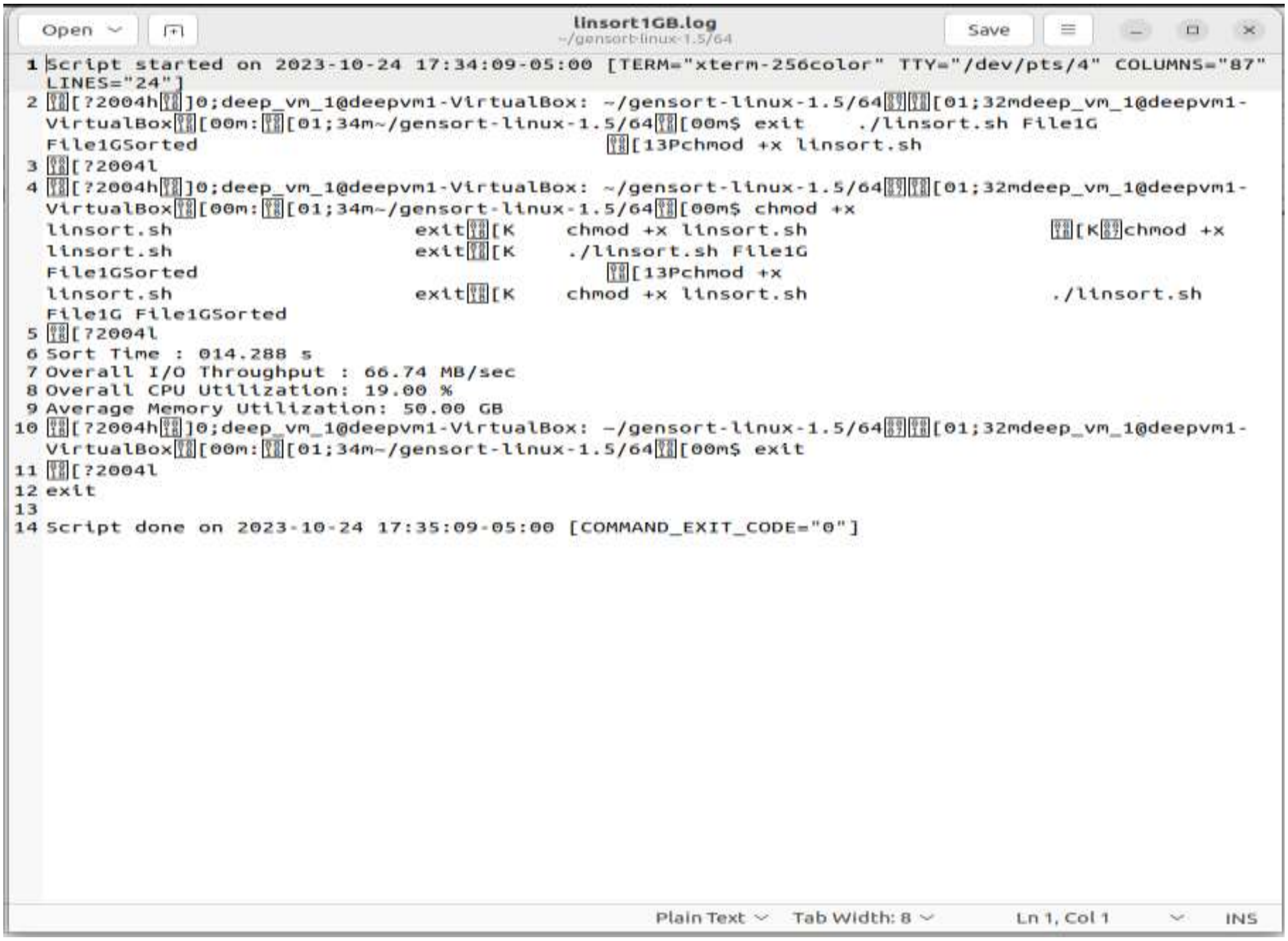
Table 1: Performance evaluation of Single Node TeraSort

- Linux Sort Implementation:

a) For 1GB:

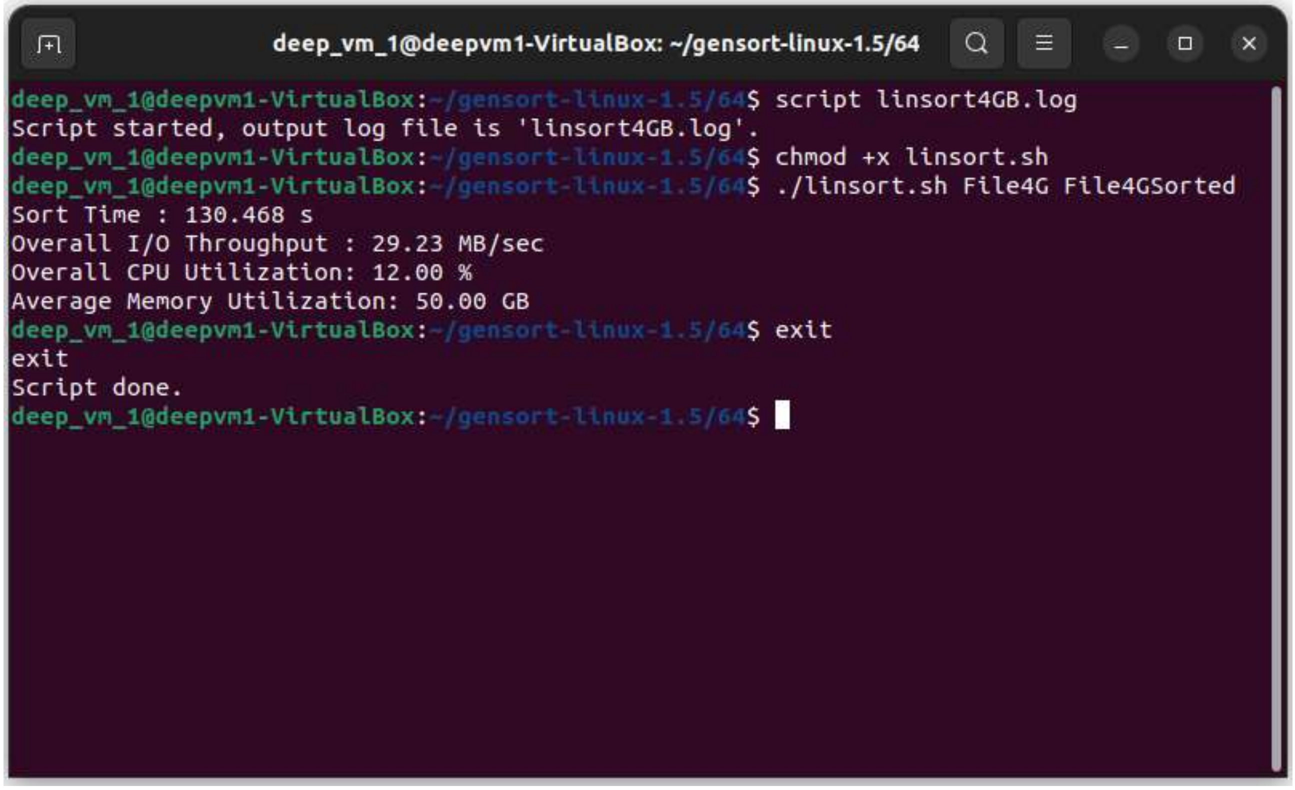


```
deep_vm_1@deepvm1-VirtualBox:~/gensort-linux-1.5/64$ script linsort1GB.log
Script started, output log file is 'linsort1GB.log'.
deep_vm_1@deepvm1-VirtualBox:~/gensort-linux-1.5/64$ chmod +x linsort.sh
deep_vm_1@deepvm1-VirtualBox:~/gensort-linux-1.5/64$ ./linsort.sh File1G File1GSorted
Sort Time : 014.288 s
Overall I/O Throughput : 66.74 MB/sec
Overall CPU Utilization: 19.00 %
Average Memory Utilization: 50.00 GB
deep_vm_1@deepvm1-VirtualBox:~/gensort-linux-1.5/64$ exit
exit
Script done.
deep_vm_1@deepvm1-VirtualBox:~/gensort-linux-1.5/64$
```

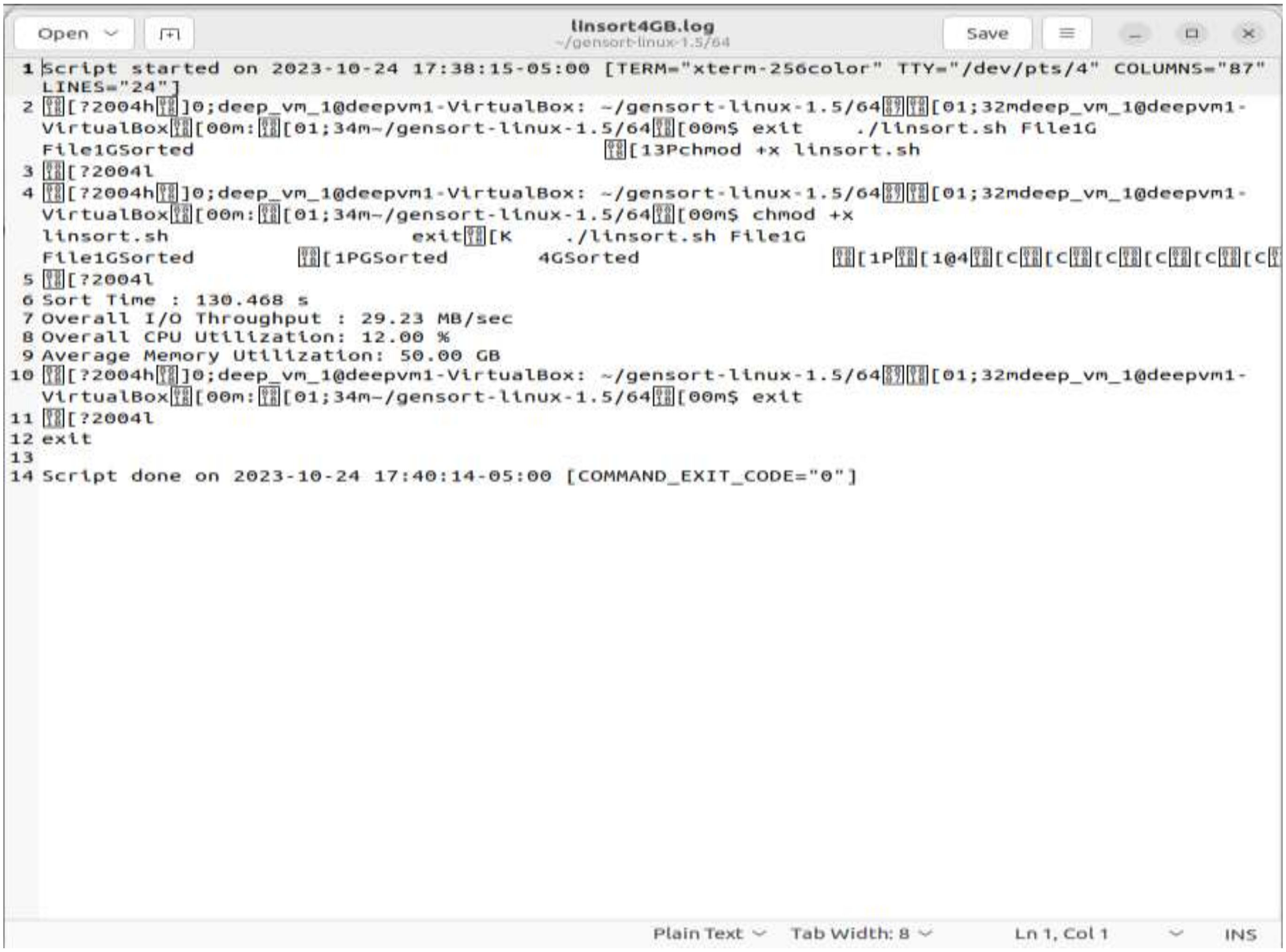


```
Script started on 2023-10-24 17:34:09-05:00 [TERM="xterm-256color" TTY="/dev/pts/4" COLUMNS="87"
LINES="24"]
□[?2004h□]0;deep_vm_1@deepvm1-VirtualBox: ~/gensort-linux-1.5/64□□[01;32mdeep_vm_1@deepvm1-
VirtualBox□[00m:□[01;34m~/gensort-linux-1.5/64□[00m$ exit     ./linsort.sh File1G
File1GSorted                                    □[13Pchmod +x linsort.sh
□[?2004l
□[?2004h□]0;deep_vm_1@deepvm1-VirtualBox: ~/gensort-linux-1.5/64□□[01;32mdeep_vm_1@deepvm1-
VirtualBox□[00m:□[01;34m~/gensort-linux-1.5/64□[00m$ chmod +x
linsort.sh                    exit□[K    chmod +x linsort.sh                    □[K□chmod +x
linsort.sh                    exit□[K    ./linsort.sh File1G
File1GSorted                            □[13Pchmod +x
linsort.sh                    exit□[K    chmod +x linsort.sh                    ./linsort.sh
File1G File1GSorted
□[?2004l
Sort Time : 014.288 s
Overall I/O Throughput : 66.74 MB/sec
Overall CPU Utilization: 19.00 %
Average Memory Utilization: 50.00 GB
□[?2004h□]0;deep_vm_1@deepvm1-VirtualBox: ~/gensort-linux-1.5/64□□[01;32mdeep_vm_1@deepvm1-
VirtualBox□[00m:□[01;34m~/gensort-linux-1.5/64□[00m$ exit
□[?2004l
exit

Script done on 2023-10-24 17:35:09-05:00 [COMMAND_EXIT_CODE="0"]
```

b) For 4GB:

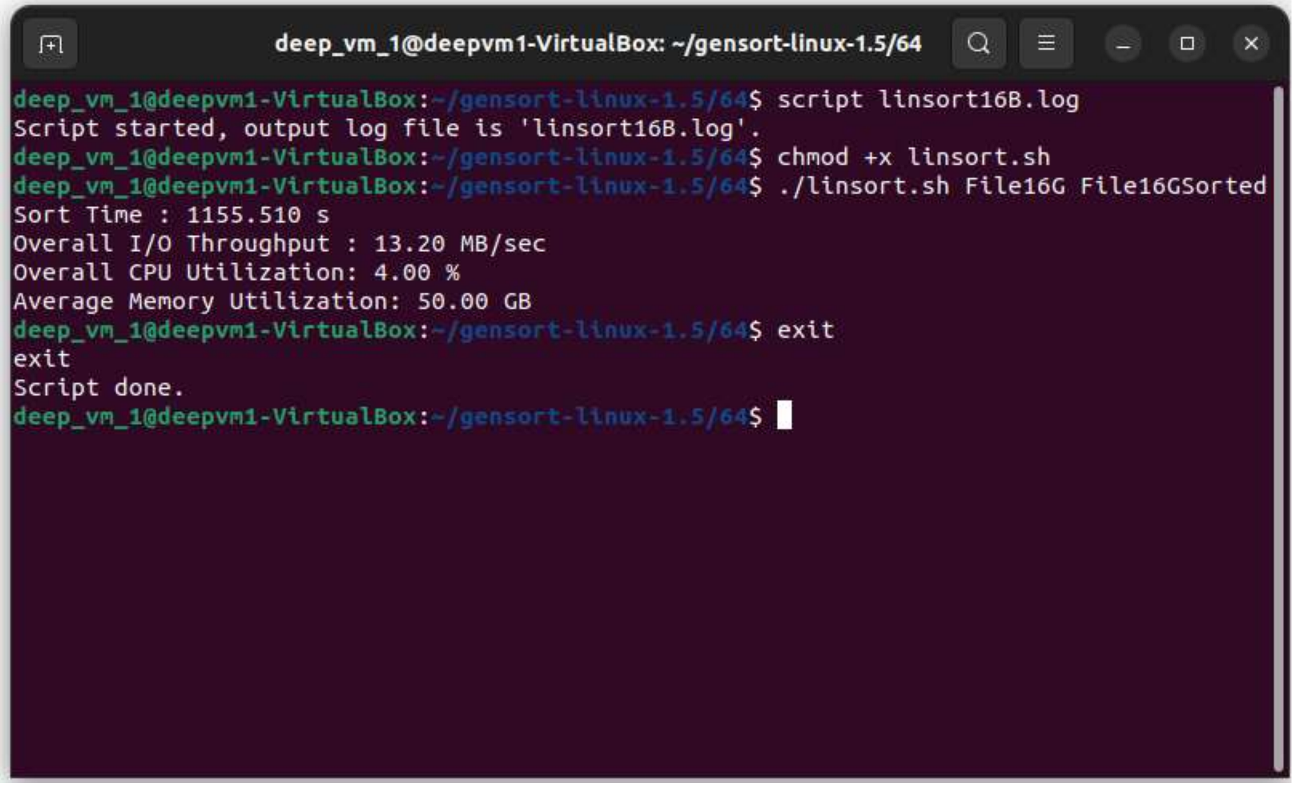


```
deep_vm_1@deepvm1-VirtualBox:~/gensort-linux-1.5/64$ script linsort4GB.log
Script started, output log file is 'linsort4GB.log'.
deep_vm_1@deepvm1-VirtualBox:~/gensort-linux-1.5/64$ chmod +x linsort.sh
deep_vm_1@deepvm1-VirtualBox:~/gensort-linux-1.5/64$ ./linsort.sh File4G File4GSorted
Sort Time : 130.468 s
Overall I/O Throughput : 29.23 MB/sec
Overall CPU Utilization: 12.00 %
Average Memory Utilization: 50.00 GB
deep_vm_1@deepvm1-VirtualBox:~/gensort-linux-1.5/64$ exit
exit
Script done.
deep_vm_1@deepvm1-VirtualBox:~/gensort-linux-1.5/64$ 
```

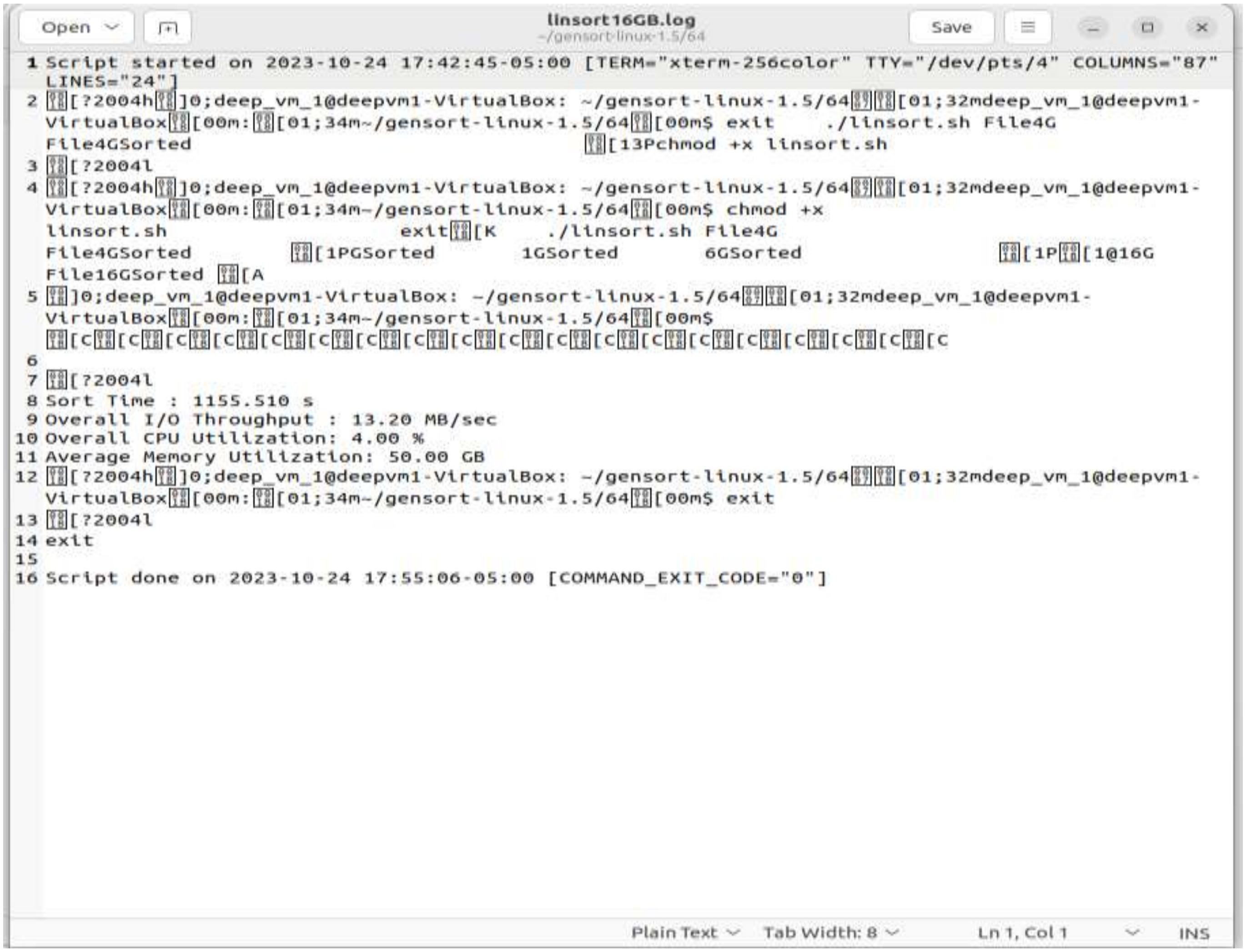


```
Script started on 2023-10-24 17:38:15-05:00 [TERM="xterm-256color" TTY="/dev/pts/4" COLUMNS="87" LINES="24"]
[?2004h]0;deep_vm_1@deepvm1-VirtualBox: ~/gensort-linux-1.5/64[01;32mdeep_vm_1@deepvm1-VirtualBox[00m:[01;34m~/gensort-linux-1.5/64[00m$ exit    ./linsort.sh File1G File1GSorted                                     [13Pchmod +x linsort.sh
[?2004l
[?2004h]0;deep_vm_1@deepvm1-VirtualBox: ~/gensort-linux-1.5/64[01;32mdeep_vm_1@deepvm1-VirtualBox[00m:[01;34m~/gensort-linux-1.5/64[00m$ chmod +x linsort.sh                   exit[K    ./linsort.sh File1G File1GSorted         [1PGSorted       4GSorted                [1P[1@4[C[C[C[C[C[C[C
[?2004l
Sort Time : 130.468 s
Overall I/O Throughput : 29.23 MB/sec
Overall CPU Utilization: 12.00 %
Average Memory Utilization: 50.00 GB
[?2004h]0;deep_vm_1@deepvm1-VirtualBox: ~/gensort-linux-1.5/64[01;32mdeep_vm_1@deepvm1-VirtualBox[00m:[01;34m~/gensort-linux-1.5/64[00m$ exit
[?2004l
exit

Script done on 2023-10-24 17:40:14-05:00 [COMMAND_EXIT_CODE="0"]
```

c) For 16GB:

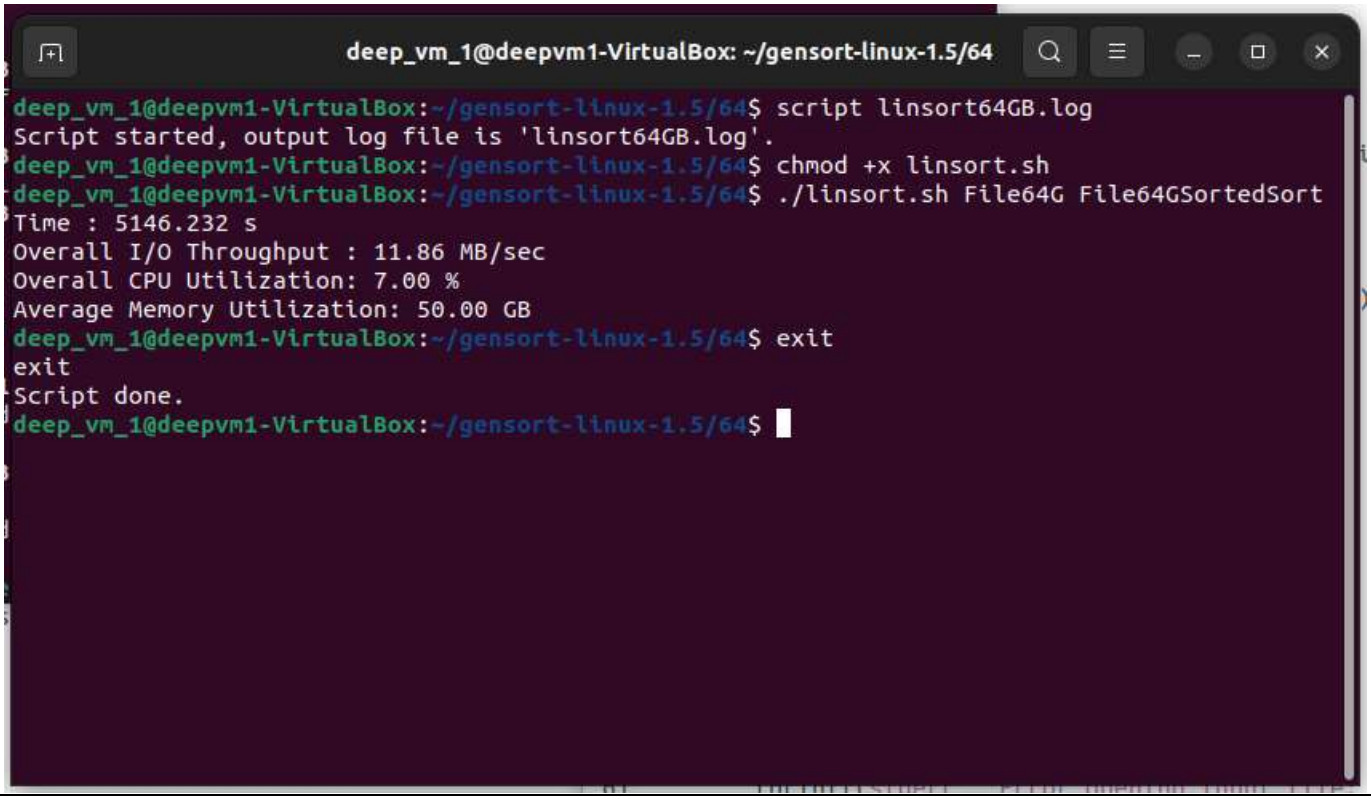


```
deep_vm_1@deepvm1-VirtualBox:~/gensort-linux-1.5/64$ script linsort16B.log
Script started, output log file is 'linsort16B.log'.
deep_vm_1@deepvm1-VirtualBox:~/gensort-linux-1.5/64$ chmod +x linsort.sh
deep_vm_1@deepvm1-VirtualBox:~/gensort-linux-1.5/64$ ./linsort.sh File16G File16GSorted
Sort Time : 1155.510 s
Overall I/O Throughput : 13.20 MB/sec
Overall CPU Utilization: 4.00 %
Average Memory Utilization: 50.00 GB
deep_vm_1@deepvm1-VirtualBox:~/gensort-linux-1.5/64$ exit
exit
Script done.
deep_vm_1@deepvm1-VirtualBox:~/gensort-linux-1.5/64$ 
```

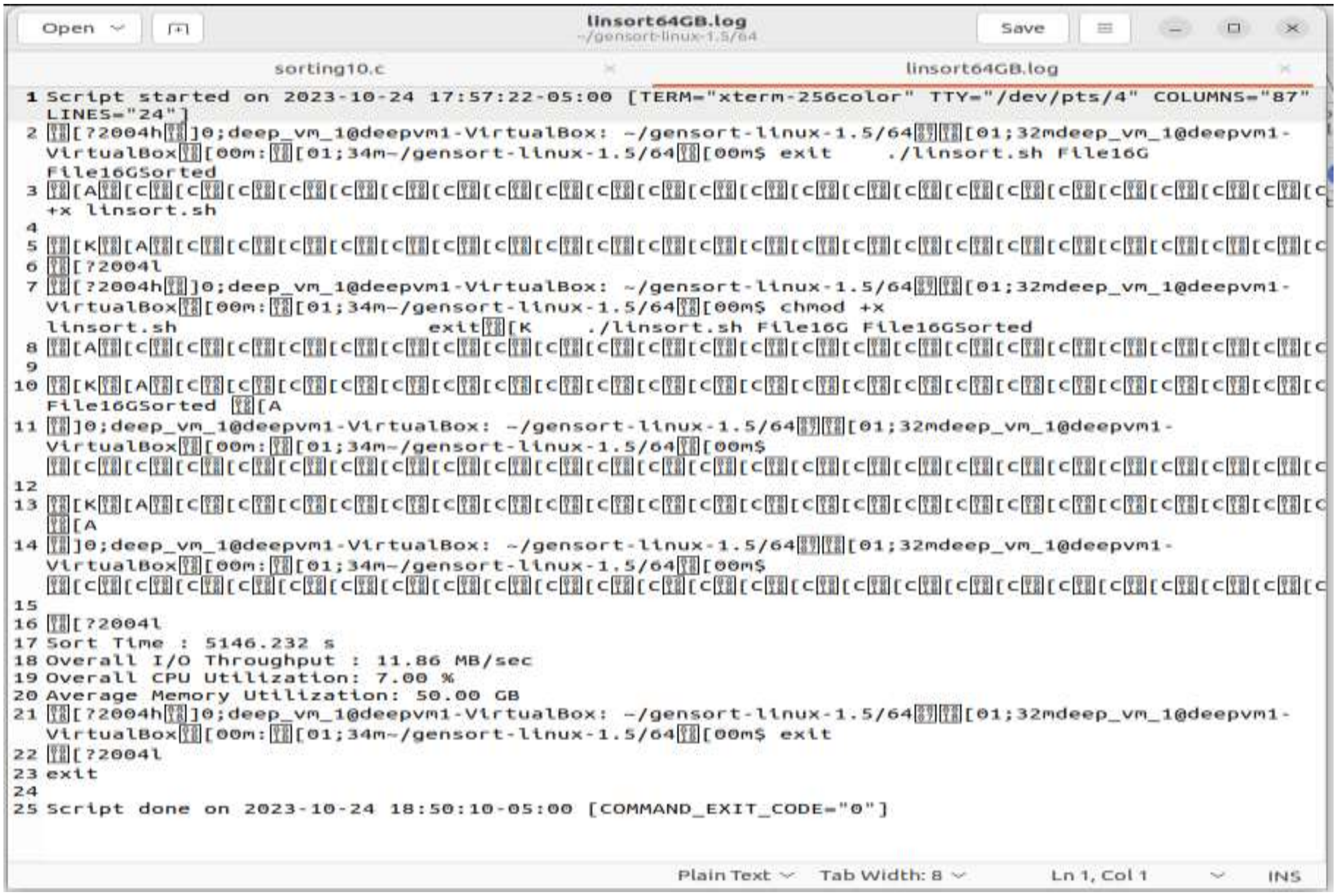


```
Script started on 2023-10-24 17:42:45-05:00 [TERM="xterm-256color" TTY="/dev/pts/4" COLUMNS="87"
LINES="24"]
□[?2004h□]0;deep_vm_1@deepvm1-VirtualBox: ~/gensort-linux-1.5/64□□[01;32mdeep_vm_1@deepvm1-
VirtualBox□[00m:□[01;34m~/gensort-linux-1.5/64□[00m$ exit    ./linsort.sh File4G
File4GSorted                                   □[13Pchmod +x linsort.sh
□[?2004l
□[?2004h□]0;deep_vm_1@deepvm1-VirtualBox: ~/gensort-linux-1.5/64□□[01;32mdeep_vm_1@deepvm1-
VirtualBox□[00m:□[01;34m~/gensort-linux-1.5/64□[00m$ chmod +x
linsort.sh                    exit□[K    ./linsort.sh File4G
File4GSorted        □[1PGSorted       1GSorted       6GSorted                 □[1P□[1@16G
File16GSorted □[A
□]0;deep_vm_1@deepvm1-VirtualBox: ~/gensort-linux-1.5/64□□[01;32mdeep_vm_1@deepvm1-
VirtualBox□[00m:□[01;34m~/gensort-linux-1.5/64□[00m$
□[C□[C□[C□[C□[C□[C□[C□[C□[C□[C□[C□[C□[C□[C□[C□[C□[C□[C□[C

□[?2004l
Sort Time : 1155.510 s
Overall I/O Throughput : 13.20 MB/sec
Overall CPU Utilization: 4.00 %
Average Memory Utilization: 50.00 GB
□[?2004h□]0;deep_vm_1@deepvm1-VirtualBox: ~/gensort-linux-1.5/64□□[01;32mdeep_vm_1@deepvm1-
VirtualBox□[00m:□[01;34m~/gensort-linux-1.5/64□[00m$ exit
□[?2004l
exit

Script done on 2023-10-24 17:55:06-05:00 [COMMAND_EXIT_CODE="0"]
```

d) For 64GB:



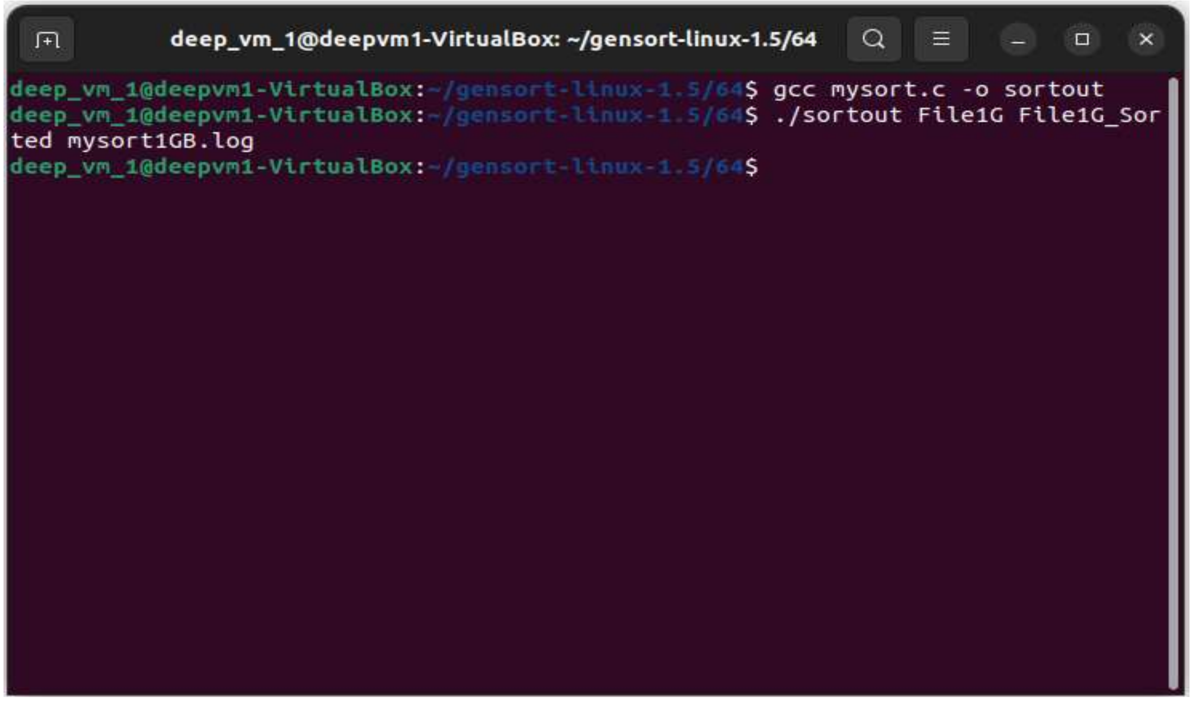
```
deep_vm_1@deepvm1-VirtualBox:~/gensort-linux-1.5/64$ script linsort64GB.log
Script started, output log file is 'linsort64GB.log'.
deep_vm_1@deepvm1-VirtualBox:~/gensort-linux-1.5/64$ chmod +x linsort.sh
deep_vm_1@deepvm1-VirtualBox:~/gensort-linux-1.5/64$ ./linsort.sh File64G File64GSortedSort
Time : 5146.232 s
Overall I/O Throughput : 11.86 MB/sec
Overall CPU Utilization: 7.00 %
Average Memory Utilization: 50.00 GB
deep_vm_1@deepvm1-VirtualBox:~/gensort-linux-1.5/64$ exit
exit
Script done.
deep_vm_1@deepvm1-VirtualBox:~/gensort-linux-1.5/64$ █
```



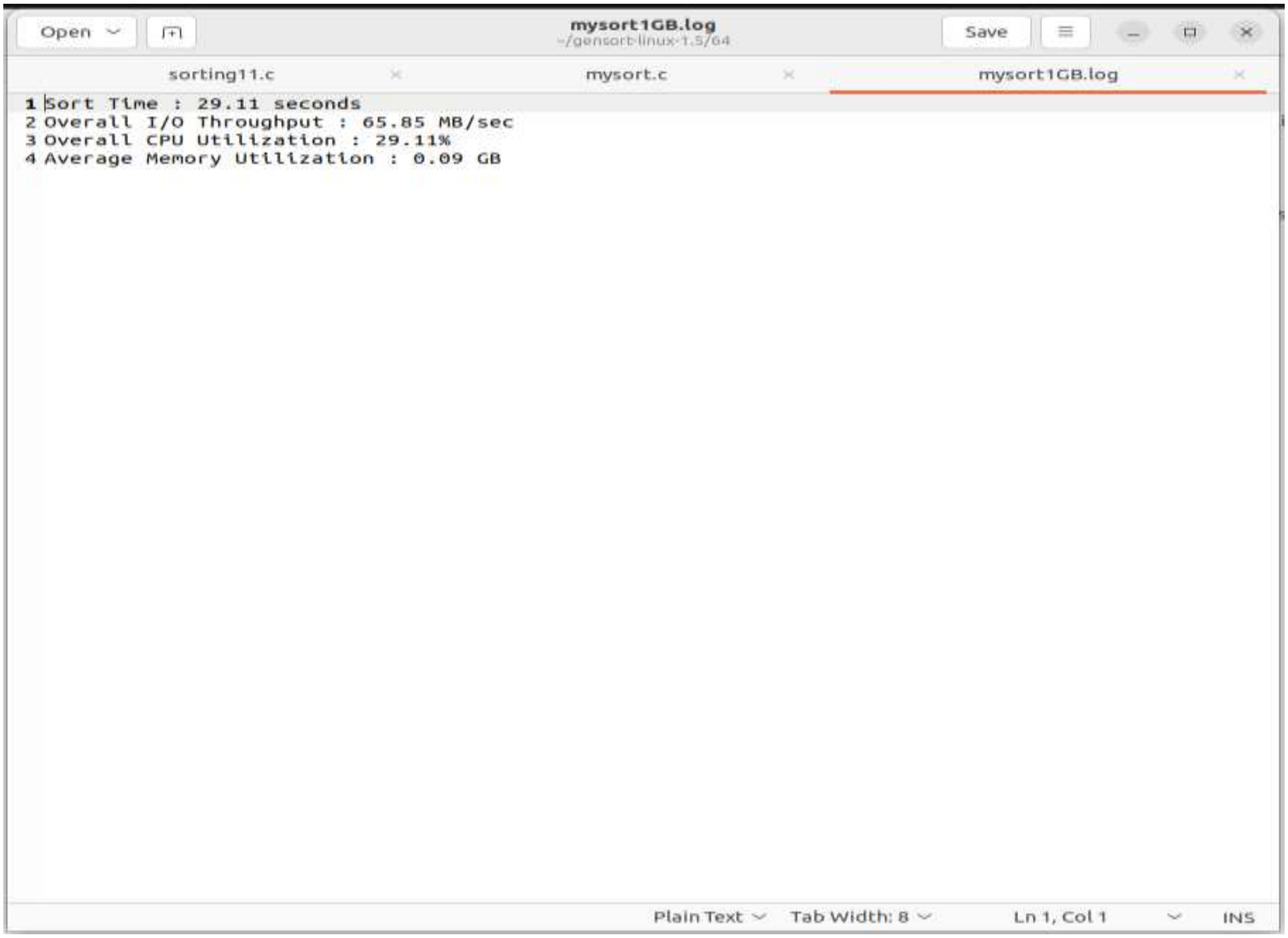
```
Script started on 2023-10-24 17:57:22-05:00 [TERM="xterm-256color" TTY="/dev/pts/4" COLUMNS="87"
LINES="24"]
▯[?2004h▯]0;deep_vm_1@deepvm1-VirtualBox: ~/gensort-linux-1.5/64▯▯[01;32mdeep_vm_1@deepvm1-
VirtualBox▯[00m:▯[01;34m~/gensort-linux-1.5/64▯[00m$ exit     ./linsort.sh File16G
File16GSorted
▯[A▯[C▯[C▯[C▯[C▯[C▯[C▯[C▯[C▯[C▯[C▯[C▯[C▯[C▯[C▯[C▯[C▯[C▯[C▯[C▯[C▯[C▯[C▯[C▯[C▯[C
+x linsort.sh

▯[K▯[A▯[C▯[C▯[C▯[C▯[C▯[C▯[C▯[C▯[C▯[C▯[C▯[C▯[C▯[C▯[C▯[C▯[C▯[C▯[C▯[C▯[C▯[C▯[C▯[C
▯[?2004l
▯[?2004h▯]0;deep_vm_1@deepvm1-VirtualBox: ~/gensort-linux-1.5/64▯▯[01;32mdeep_vm_1@deepvm1-
VirtualBox▯[00m:▯[01;34m~/gensort-linux-1.5/64▯[00m$ chmod +x
linsort.sh                   exit▯[K     ./linsort.sh File16G File16GSorted
▯[A▯[C▯[C▯[C▯[C▯[C▯[C▯[C▯[C▯[C▯[C▯[C▯[C▯[C▯[C▯[C▯[C▯[C▯[C▯[C▯[C▯[C▯[C▯[C▯[C▯[C

▯[K▯[A▯[C▯[C▯[C▯[C▯[C▯[C▯[C▯[C▯[C▯[C▯[C▯[C▯[C▯[C▯[C▯[C▯[C▯[C▯[C▯[C▯[C▯[C▯[C▯[C
File16GSorted ▯[A
▯]0;deep_vm_1@deepvm1-VirtualBox: ~/gensort-linux-1.5/64▯▯[01;32mdeep_vm_1@deepvm1-
VirtualBox▯[00m:▯[01;34m~/gensort-linux-1.5/64▯[00m$
▯[C▯[C▯[C▯[C▯[C▯[C▯[C▯[C▯[C▯[C▯[C▯[C▯[C▯[C▯[C▯[C▯[C▯[C▯[C▯[C▯[C▯[C▯[C▯[C▯[C

▯[K▯[A▯[C▯[C▯[C▯[C▯[C▯[C▯[C▯[C▯[C▯[C▯[C▯[C▯[C▯[C▯[C▯[C▯[C▯[C▯[C▯[C▯[C▯[C▯[C▯[C
▯[A
▯]0;deep_vm_1@deepvm1-VirtualBox: ~/gensort-linux-1.5/64▯▯[01;32mdeep_vm_1@deepvm1-
VirtualBox▯[00m:▯[01;34m~/gensort-linux-1.5/64▯[00m$
▯[C▯[C▯[C▯[C▯[C▯[C▯[C▯[C▯[C▯[C▯[C▯[C▯[C▯[C▯[C▯[C▯[C▯[C▯[C▯[C▯[C▯[C▯[C▯[C▯[C

▯[?2004l
Sort Time : 5146.232 s
Overall I/O Throughput : 11.86 MB/sec
Overall CPU Utilization: 7.00 %
Average Memory Utilization: 50.00 GB
▯[?2004h▯]0;deep_vm_1@deepvm1-VirtualBox: ~/gensort-linux-1.5/64▯▯[01;32mdeep_vm_1@deepvm1-
VirtualBox▯[00m:▯[01;34m~/gensort-linux-1.5/64▯[00m$ exit
▯[?2004l
exit

Script done on 2023-10-24 18:50:10-05:00 [COMMAND_EXIT_CODE="0"]
```

- Shared Memory Implementation:

a) For 1GB:

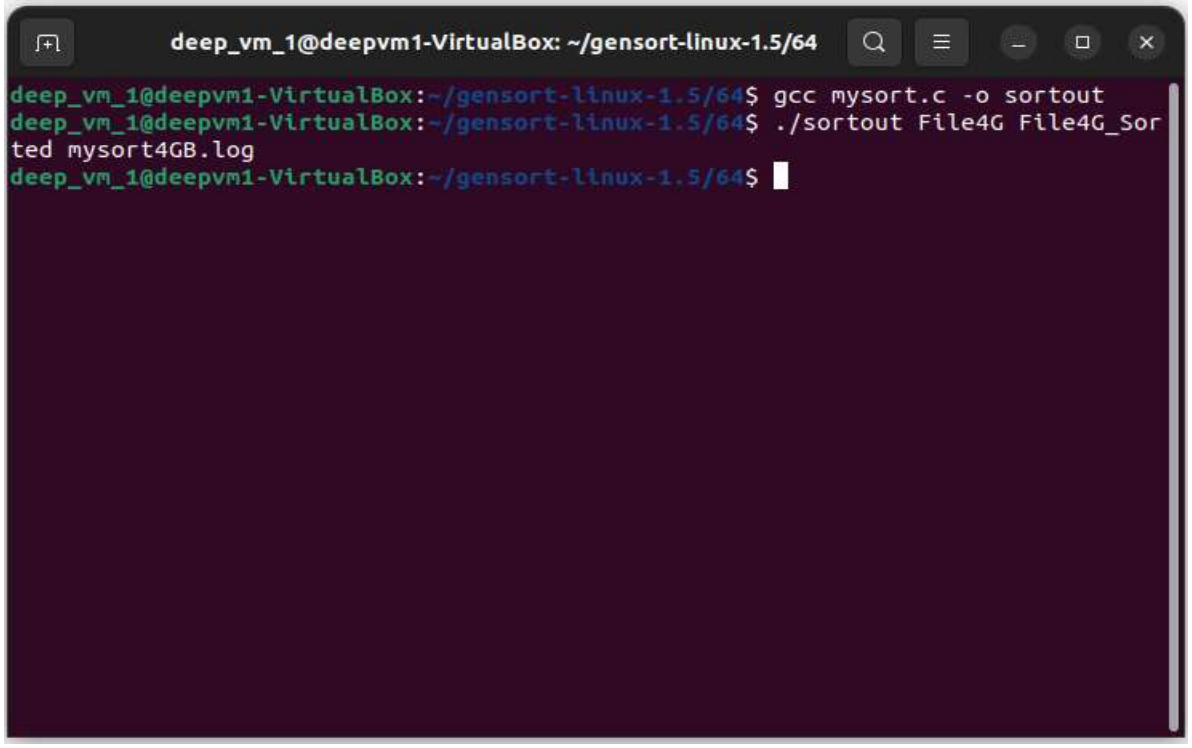


```
deep_vm_1@deepvm1-VirtualBox:~/gensort-linux-1.5/64$ gcc mysort.c -o sortout
deep_vm_1@deepvm1-VirtualBox:~/gensort-linux-1.5/64$ ./sortout File1G File1G_Sor
ted mysort1GB.log
deep_vm_1@deepvm1-VirtualBox:~/gensort-linux-1.5/64$
```

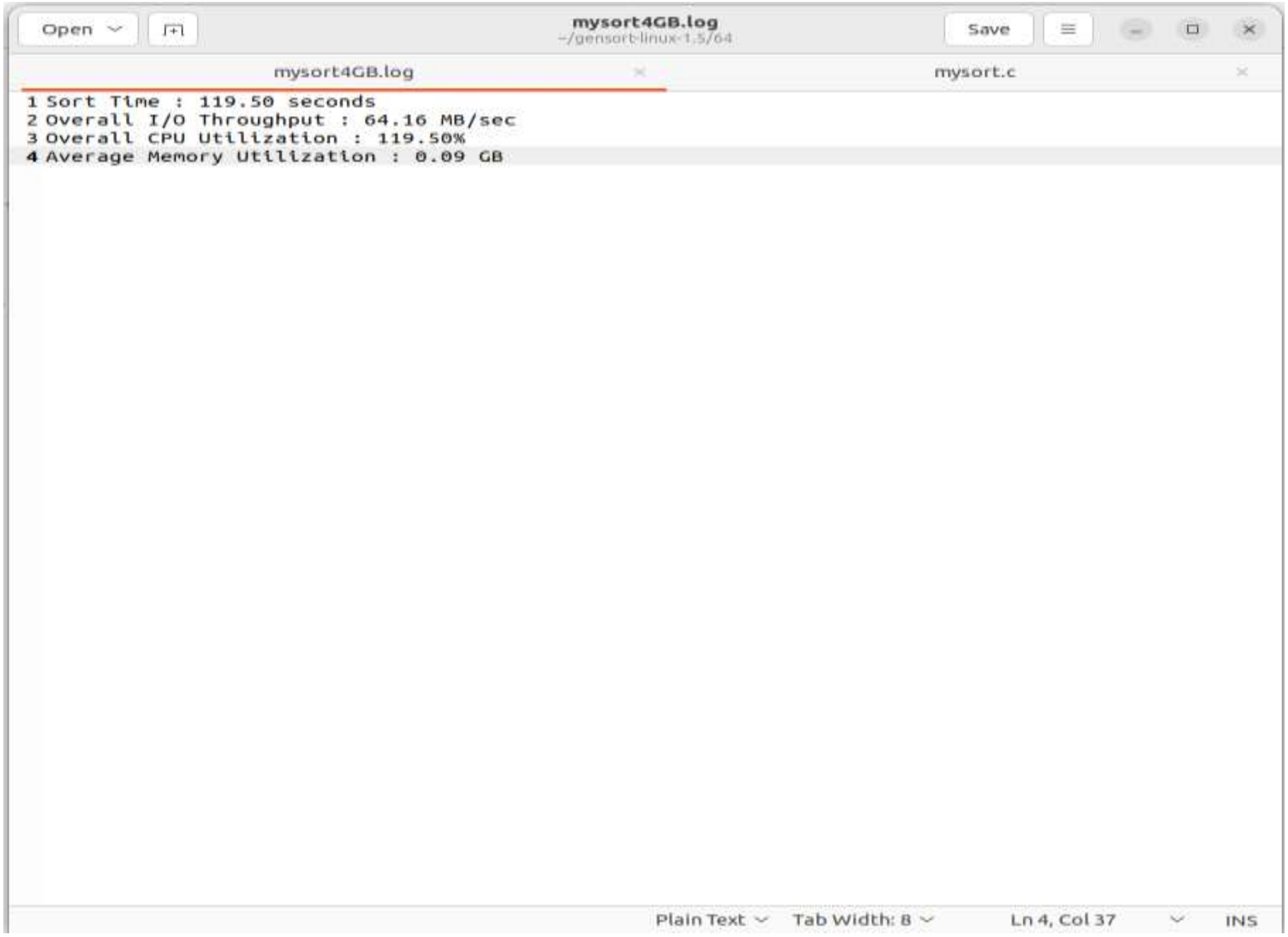


```
Sort Time : 29.11 seconds
Overall I/O Throughput : 65.85 MB/sec
Overall CPU Utilization : 29.11%
Average Memory Utilization : 0.09 GB
```

b) For 4GB:

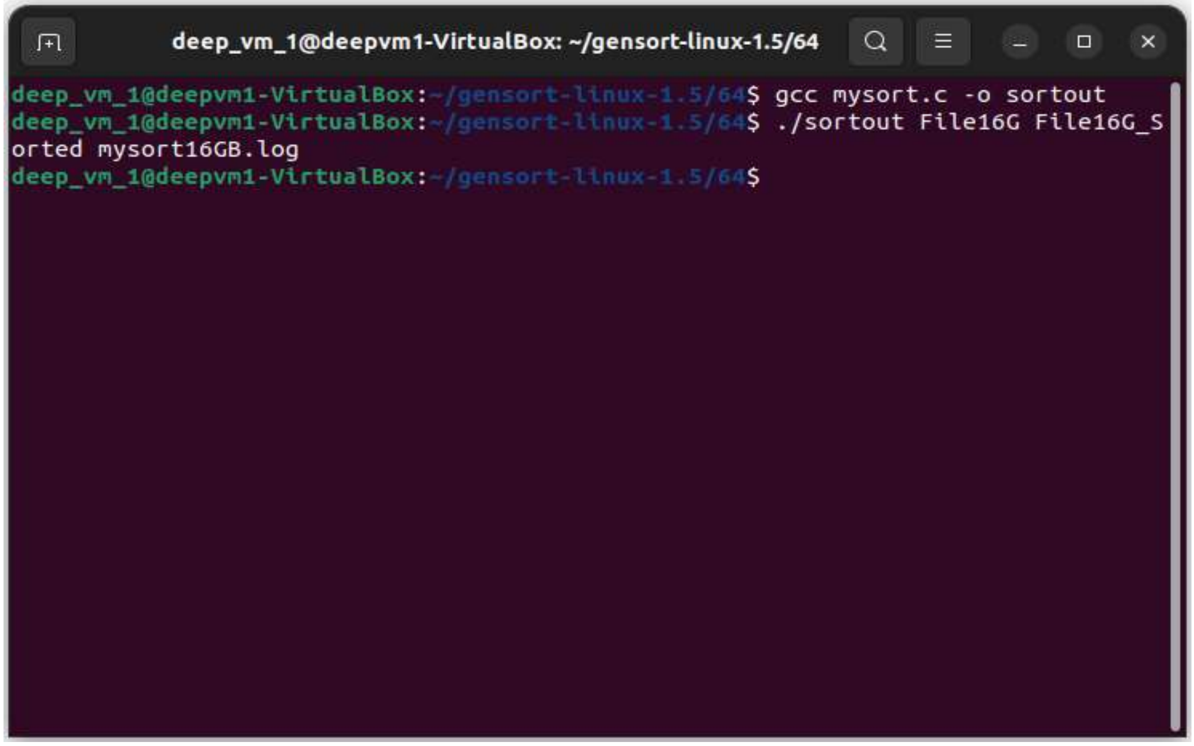


```
deep_vm_1@deepvm1-VirtualBox:~/gensort-linux-1.5/64$ gcc mysort.c -o sortout
deep_vm_1@deepvm1-VirtualBox:~/gensort-linux-1.5/64$ ./sortout File4G File4G_Sor
ted mysort4GB.log
deep_vm_1@deepvm1-VirtualBox:~/gensort-linux-1.5/64$
```

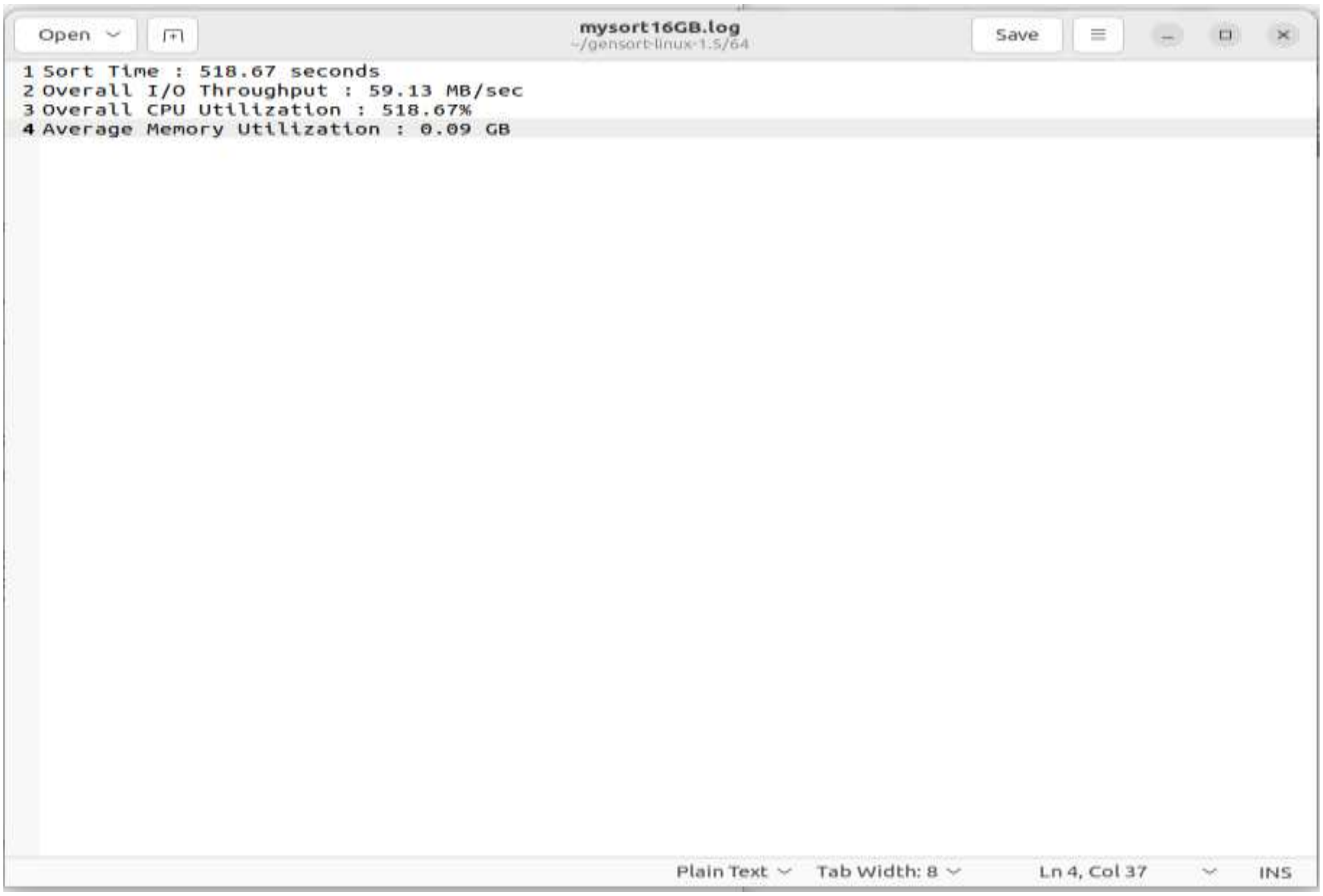


```
Sort Time : 119.50 seconds
Overall I/O Throughput : 64.16 MB/sec
Overall CPU Utilization : 119.50%
Average Memory Utilization : 0.09 GB
```

c) For 16GB:

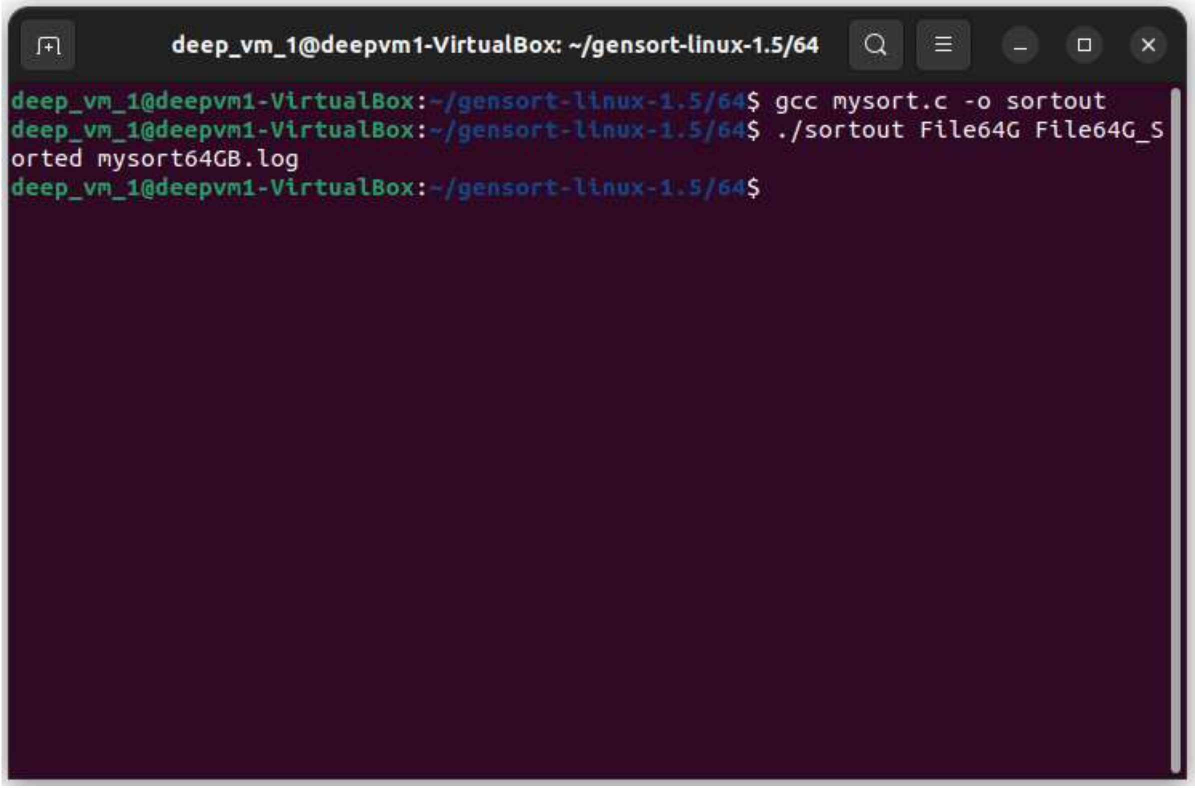


```
deep_vm_1@deepvm1-VirtualBox:~/gensort-linux-1.5/64$ gcc mysort.c -o sortout
deep_vm_1@deepvm1-VirtualBox:~/gensort-linux-1.5/64$ ./sortout File16G File16G_S
orted mysort16GB.log
deep_vm_1@deepvm1-VirtualBox:~/gensort-linux-1.5/64$
```

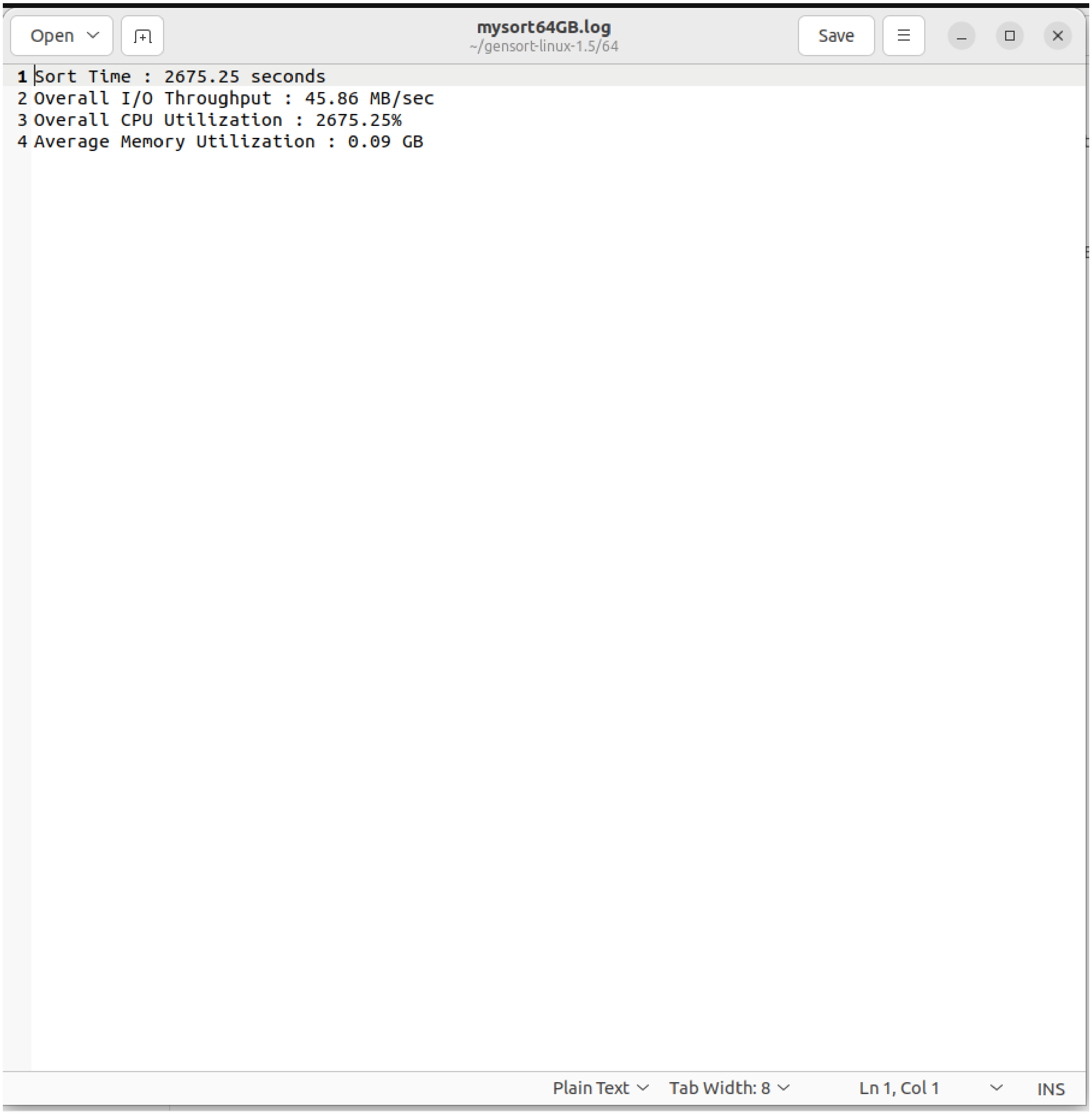


```
Sort Time : 518.67 seconds
Overall I/O Throughput : 59.13 MB/sec
Overall CPU Utilization : 518.67%
Average Memory Utilization : 0.09 GB
```

d) For 64GB:



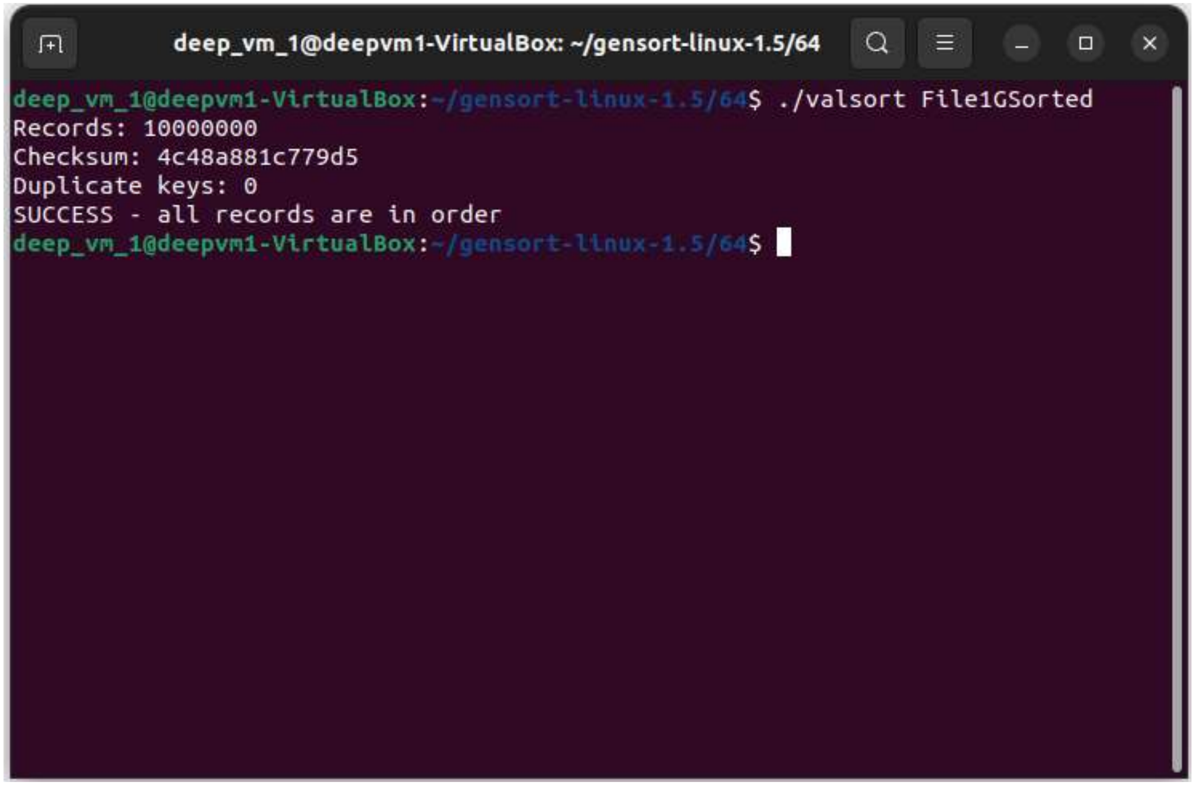
```
deep_vm_1@deepvm1-VirtualBox:~/gensort-linux-1.5/64$ gcc mysort.c -o sortout
deep_vm_1@deepvm1-VirtualBox:~/gensort-linux-1.5/64$ ./sortout File64G File64G_S
orted mysort64GB.log
deep_vm_1@deepvm1-VirtualBox:~/gensort-linux-1.5/64$
```



```
Sort Time : 2675.25 seconds
Overall I/O Throughput : 45.86 MB/sec
Overall CPU Utilization : 2675.25%
Average Memory Utilization : 0.09 GB
```

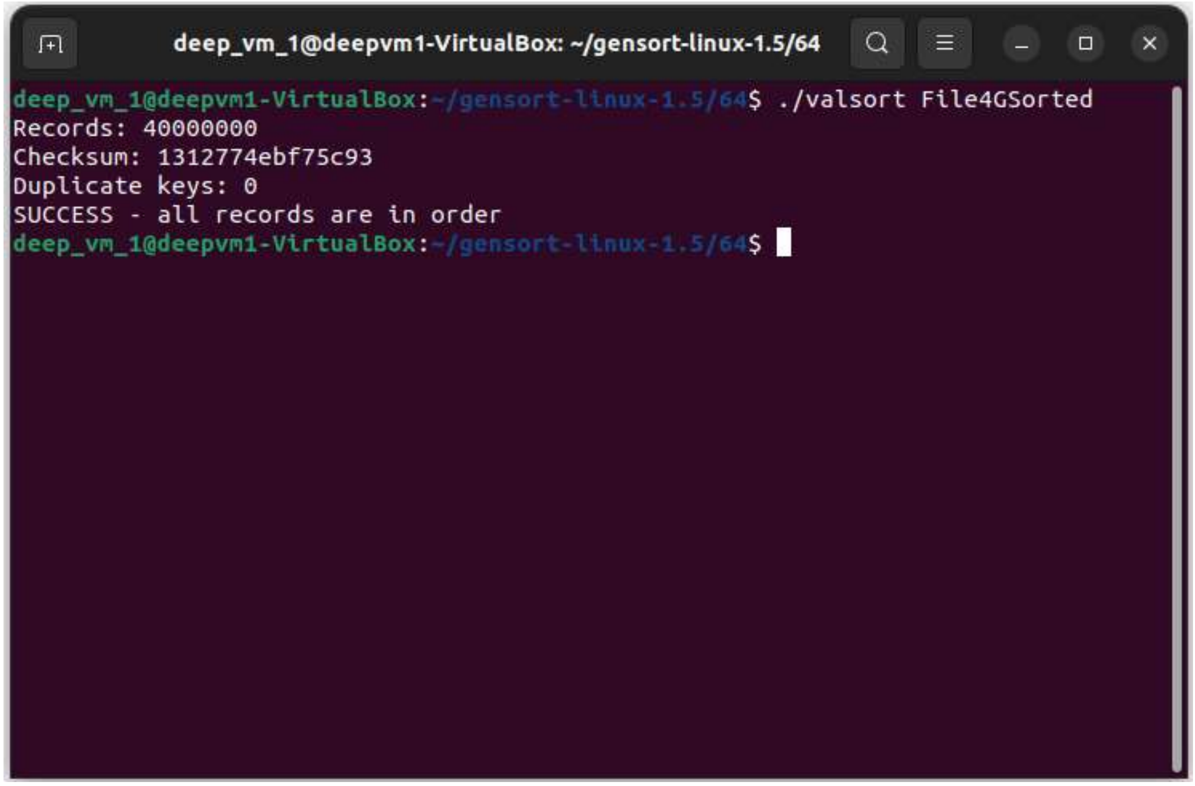
- Using valsort to verify the sorted outputs:

a) For 1GB:



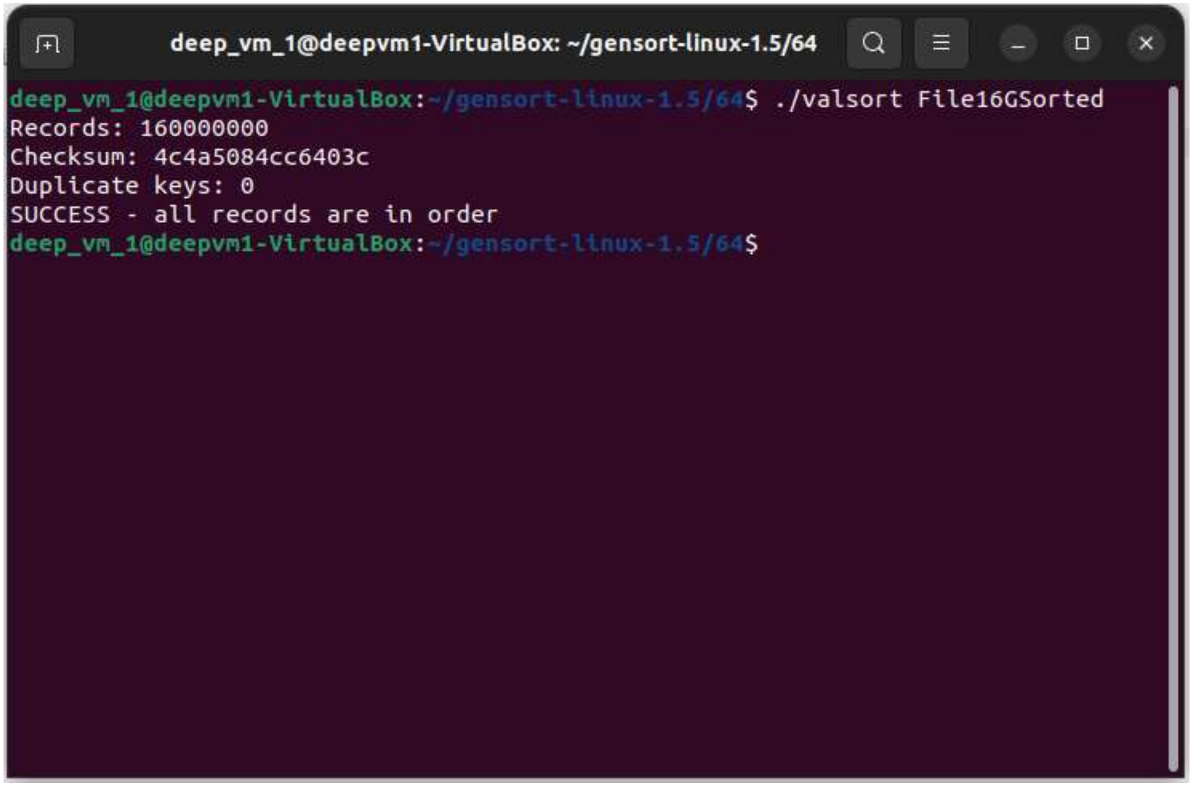
```
deep_vm_1@deepvm1-VirtualBox:~/gensort-linux-1.5/64$ ./valsort File1GSorted
Records: 10000000
Checksum: 4c48a881c779d5
Duplicate keys: 0
SUCCESS - all records are in order
deep_vm_1@deepvm1-VirtualBox:~/gensort-linux-1.5/64$
```

b) For 4GB:



```
deep_vm_1@deepvm1-VirtualBox:~/gensort-linux-1.5/64$ ./valsort File4GSorted
Records: 40000000
Checksum: 1312774ebf75c93
Duplicate keys: 0
SUCCESS - all records are in order
deep_vm_1@deepvm1-VirtualBox:~/gensort-linux-1.5/64$
```

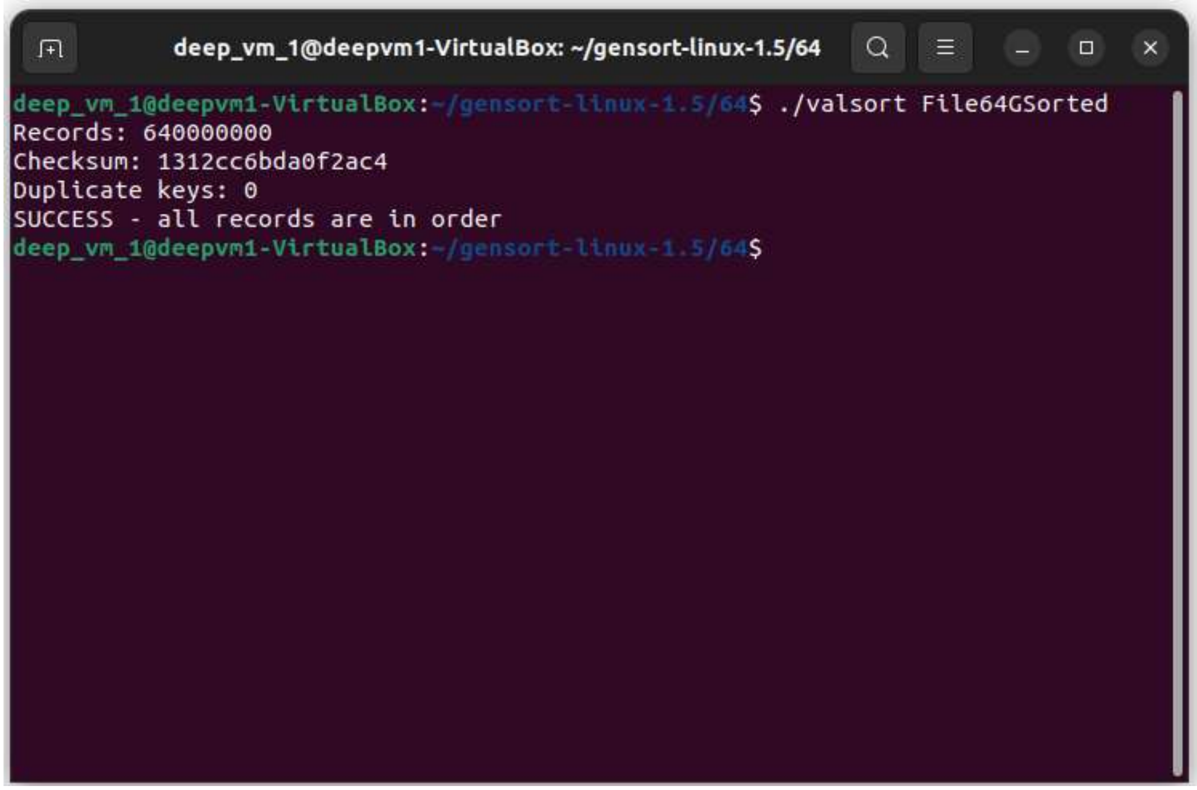
c) For 16GB:



deep_vm_1@deepvm1-VirtualBox: ~/gensort-linux-1.5/64

```
deep_vm_1@deepvm1-VirtualBox:~/gensort-linux-1.5/64$ ./valsort File16GSorted
Records: 160000000
Checksum: 4c4a5084cc6403c
Duplicate keys: 0
SUCCESS - all records are in order
deep_vm_1@deepvm1-VirtualBox:~/gensort-linux-1.5/64$
```

d) For 64GB:



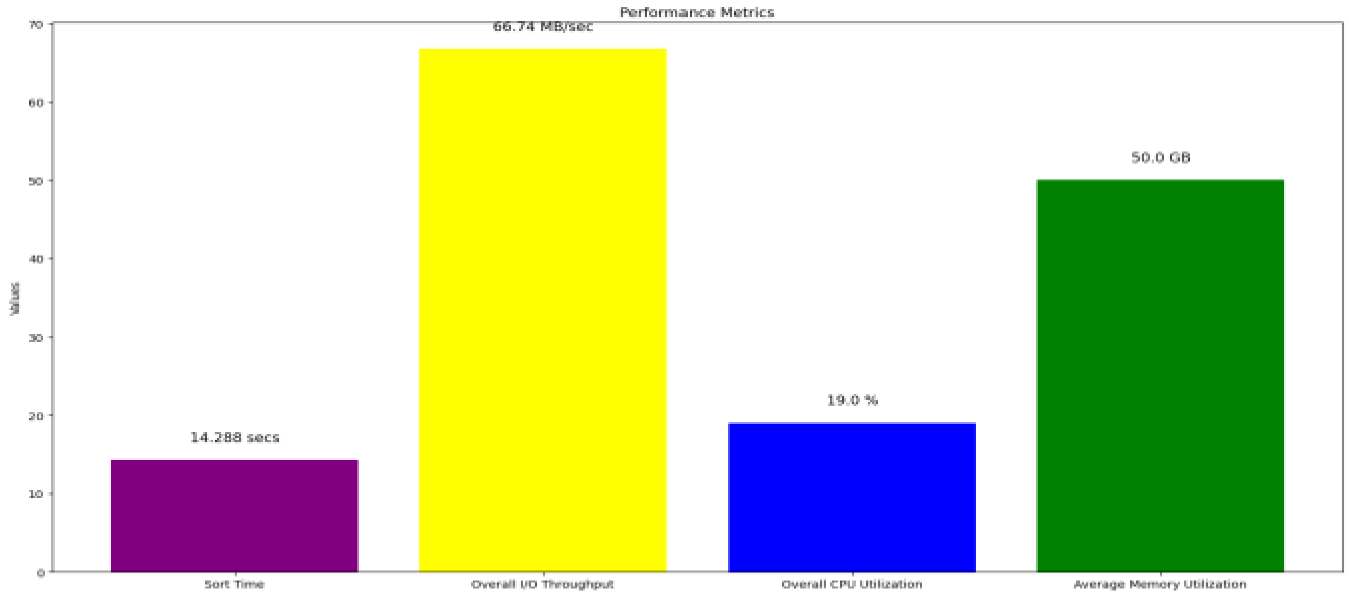
deep_vm_1@deepvm1-VirtualBox: ~/gensort-linux-1.5/64

```
deep_vm_1@deepvm1-VirtualBox:~/gensort-linux-1.5/64$ ./valsort File64GSorted
Records: 640000000
Checksum: 1312cc6bda0f2ac4
Duplicate keys: 0
SUCCESS - all records are in order
deep_vm_1@deepvm1-VirtualBox:~/gensort-linux-1.5/64$
```

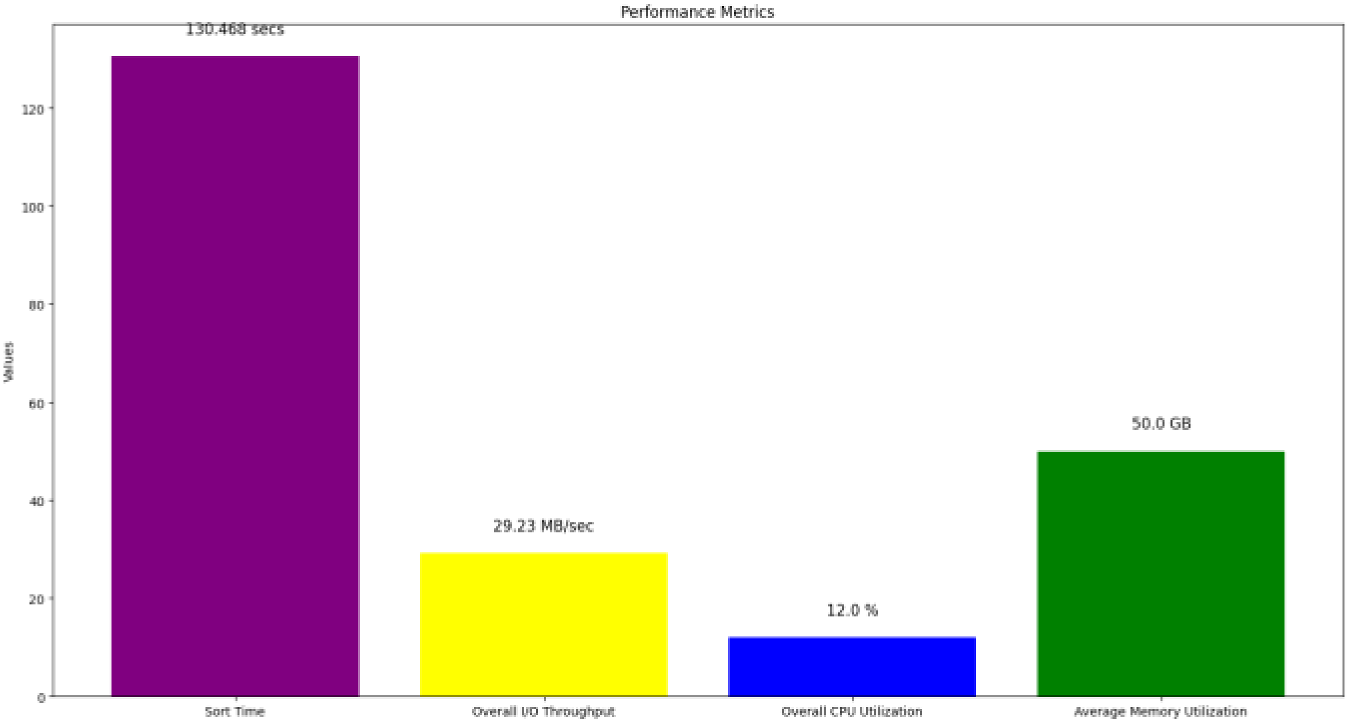
- **Plotting the graph to plot sort time (sec), Overall, I/O Throughput (MB/sec) Overall CPU Utilization (%), and Average Memory Utilization (GB) (%) as a function of time, and generate a plot:**

- Plotting for Linux Sort:

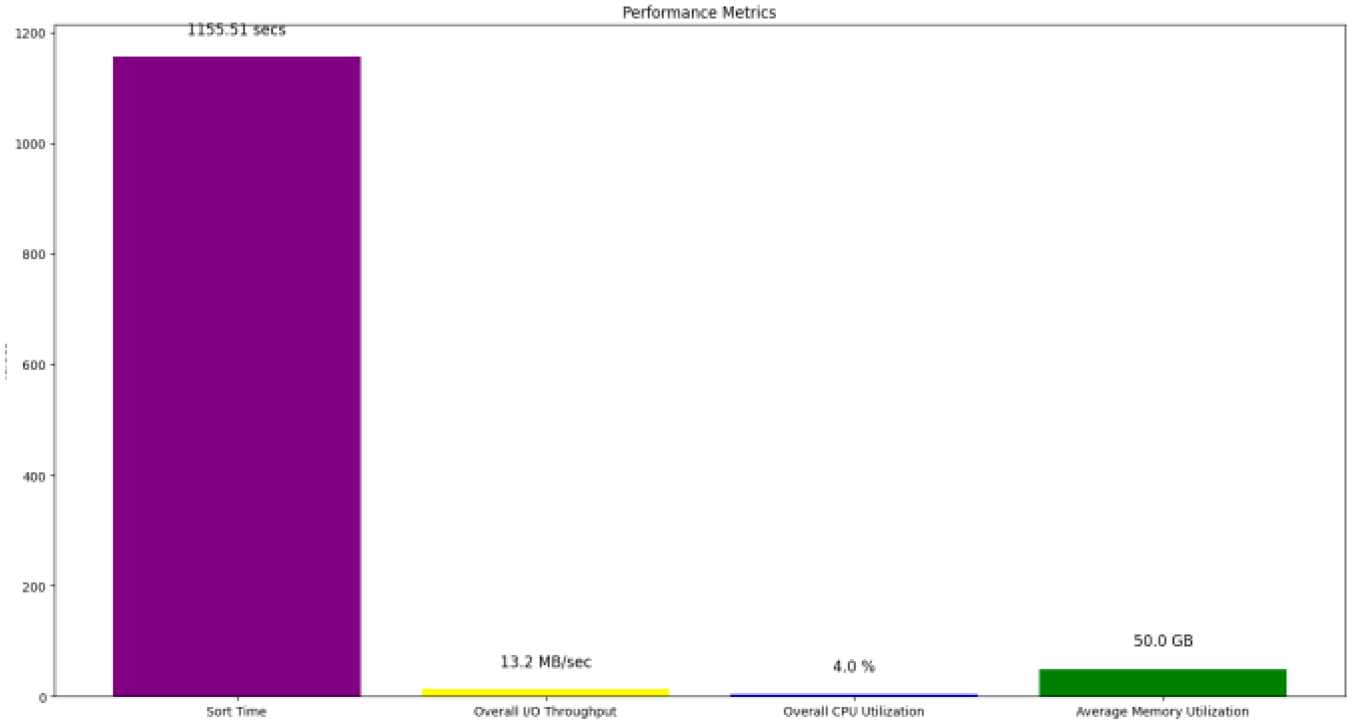
a) For 1GB:



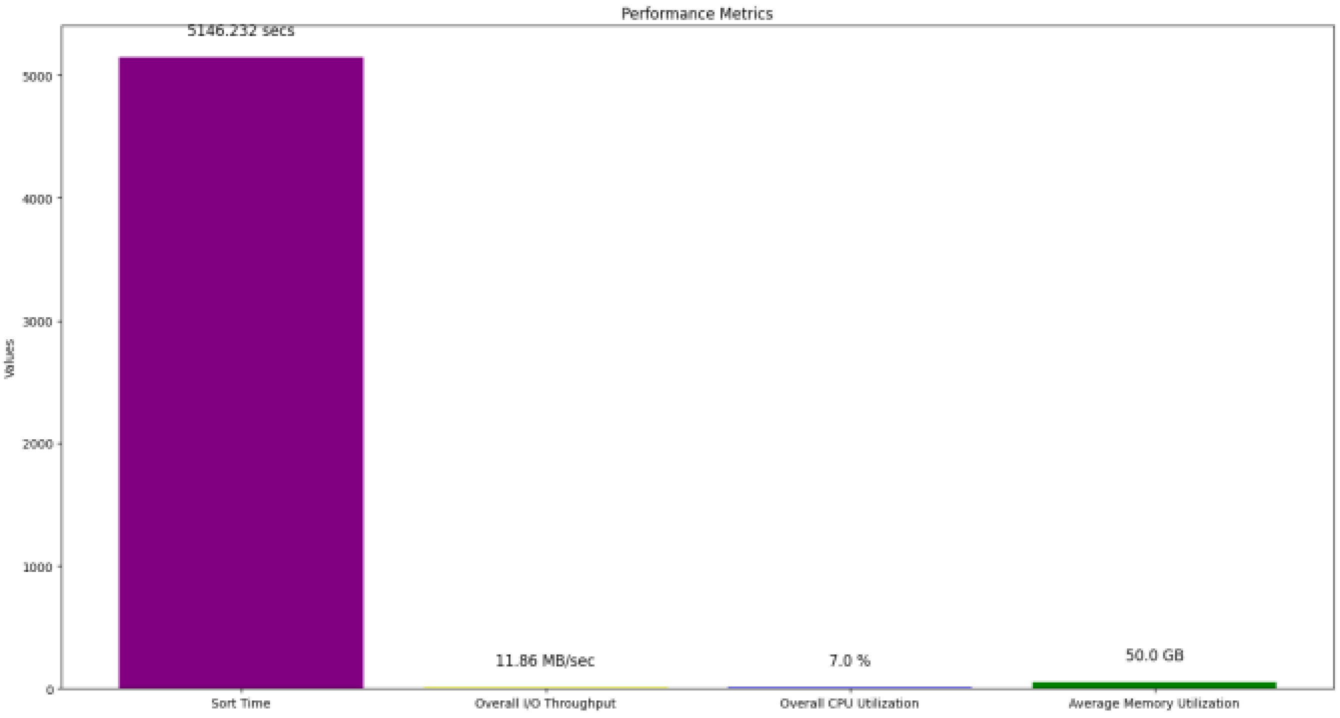
b) For 4GB:



c) For 16GB:

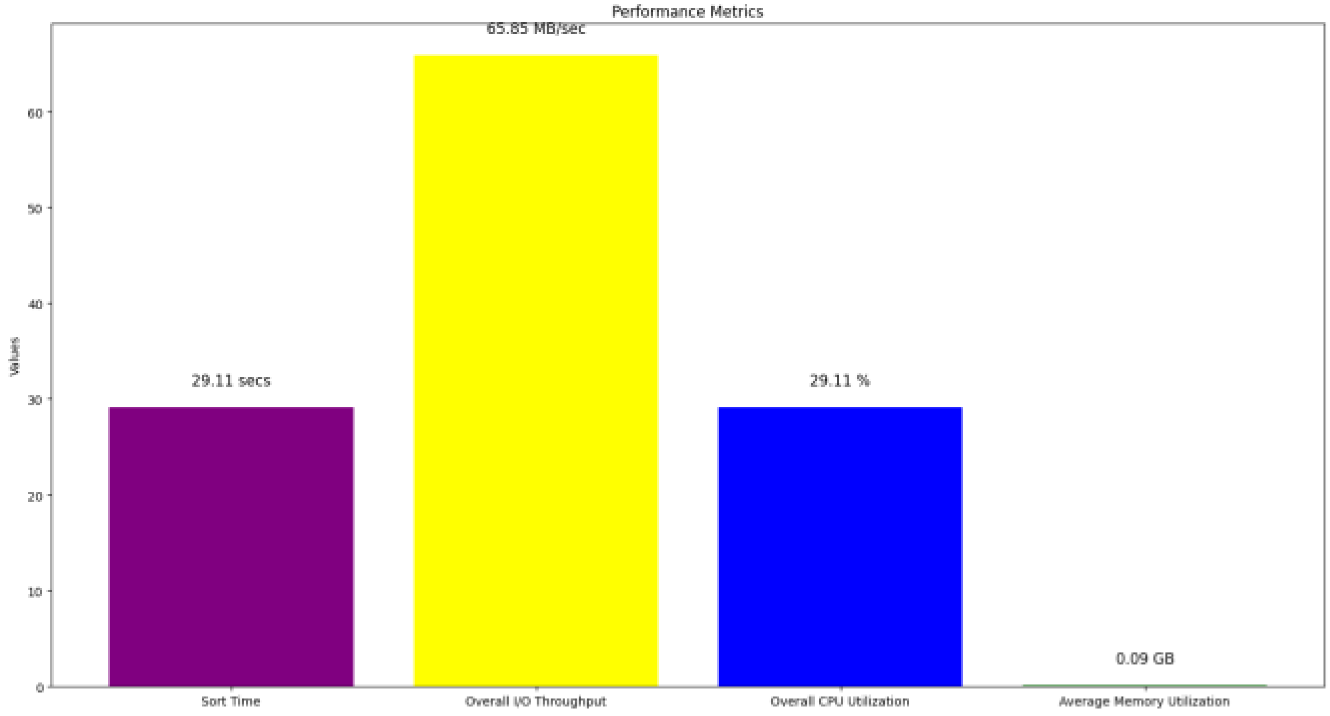


d) For 64GB:

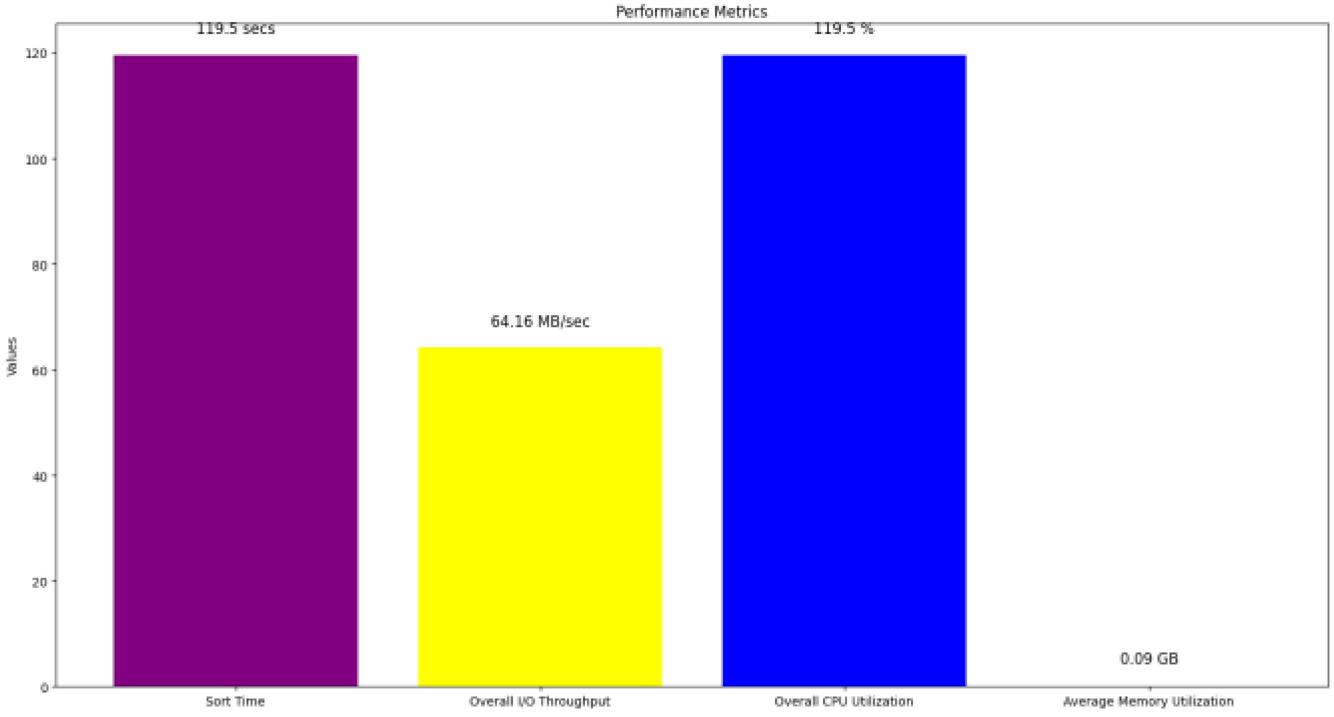


- **Plotting for Shared Memory Sort:**

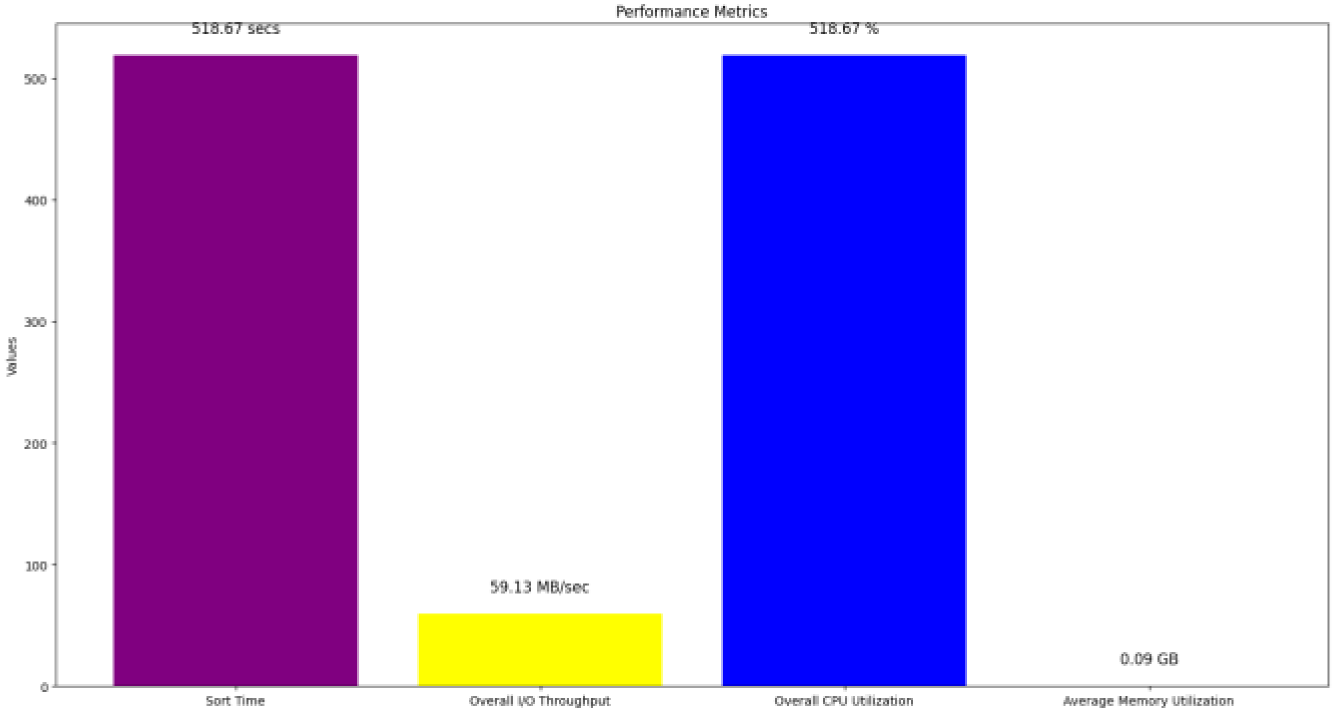
a) For 1GB:



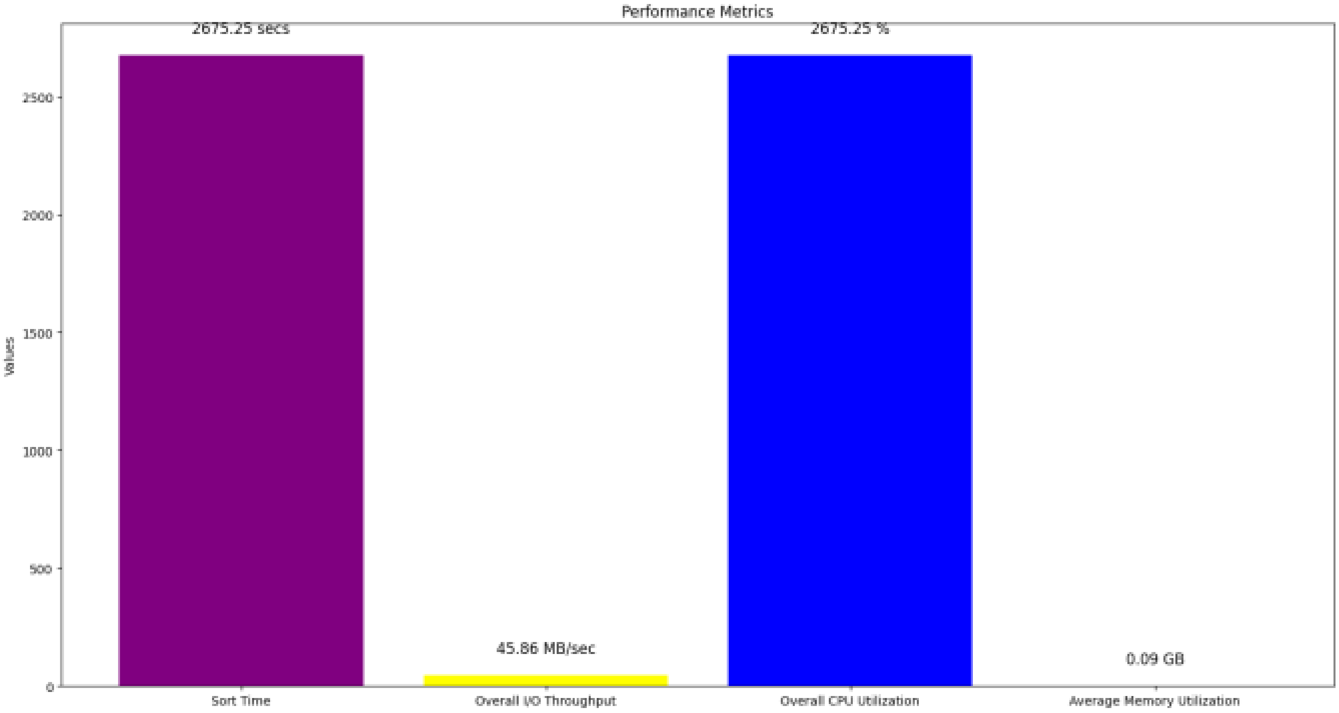
b) For 4GB:



c) **For 16GB:**



d) **For 64GB:**

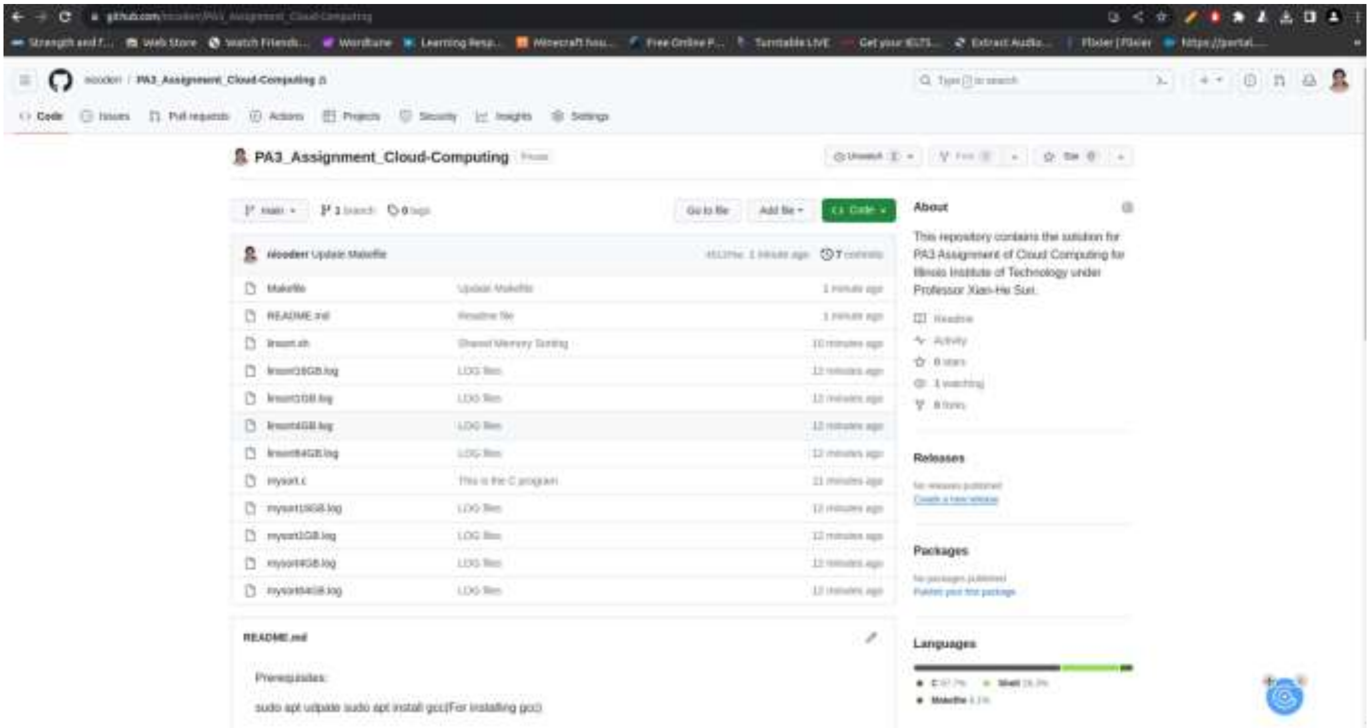


- **Conclusion:**

The project provided an excellent chance to practice benchmarking, keeping track of CPU and memory consumption, and using a superb sort approach with constrained computing resources. Additionally, we became aware of how blindly and naively we had been creating software without consideration for memory, CPU, or any other available resources. Even though we haven't finished implementing, we are astonished by the cost of I/O operations, which we have heard about frequently but have never had to deal with.

The above implementation indicates that it takes lot of time for executing and sorting 64GB file as compared to 1GB, 4GB and 16GB file because of the quicksort technique and large files sizes.

- **Github Repository:**



PA3_Assignment_Cloud-Computing Private

nicoderr Update Makefile · 1 minute ago · 7 commits

Makefile	Update Makefile	1 minute ago
README.md	Readme file	1 minute ago
lrsort.sh	Shared Memory Sorting	10 minutes ago
lrsort16GB.log	LOG files	12 minutes ago
lrsort1GB.log	LOG files	12 minutes ago
lrsort4GB.log	LOG files	12 minutes ago
lrsort64GB.log	LOG files	12 minutes ago
mysort.c	This is the C program	11 minutes ago
mysort16GB.log	LOG files	12 minutes ago
mysort1GB.log	LOG files	12 minutes ago
mysort4GB.log	LOG files	12 minutes ago
mysort64GB.log	LOG files	12 minutes ago

README.md

Prerequisites:

sudo apt udpate sudo apt install gcc(For installing gcc)

About

This repository contains the solution for PA3 Assignment of Cloud Computing for Illinois Institute of Technology under Professor Xian-He Sun.

Readme
Activity
0 stars
1 watching
0 forks

Releases

No releases published
Create a new release

Packages

No packages published
Publish your first package

Languages

- C [illegible]
- Shell [illegible]
- Makefile [illegible]

- **Github link:** https://github.com/nicoderr/PA3_Assignment_Cloud-Computing

For the 64GB workload, monitor the disk I/O speed (in MB/sec), memory utilization (GB), and processor utilization (%) as a function of time, and generate a plot for the entire experiment. Here is an example of a plot that has cpu utilization and memory utilization (https://i.stack.imgur.com/dmYAB.png), plot a similar looking graph but with the disk I/O data as well as a 3rd line. Do this for both shared memory benchmark (your code) and for the Linux Sort. You might find some online info useful on how to monitor this type of information (https://unix.stackexchange.com/questions/554/how-to-monitor-cpu-memory-usage-of-a-single-process).
After you have both graphs, discuss the differences you see, which might explain the difference in performance you get between the two implementations. Make sure your data is not cached in the OS memory before you run your experiments.

3. What you will submit

The grading will be done according to the rubric below:

- Shared memory sort implementation/scripts: 50 points
- README.md: 5 points
- Performance evaluation, data, explanations, etc: 40 points
- Followed instructions on deliverables: 5

points The maximum score that will be allowed is

100 points.

You must have working code that compiles and runs on Ubuntu Linux 22.04 to receive credit for report/performance. A separate (typed) design document (named hw5-report.pdf) of approximately 1-3 pages describing the overall benchmark design, and design tradeoffs considered and made. Add a brief description of the problem, methodology, and runtime environment settings. You are to fill in the table on the previous page. Please explain your results, and explain the difference in performance? Include logs from your application as well as valsort (e.g., standard output) that clearly shows the completion of the sort invocations with clear timing information and experiment details; include separate logs for shared memory sort and Linux sort, for each dataset. Valsort can be found as part of the gensort suite (http://www.ordinal.com/gensort.html), and it is used to validate the sort. As part of your submission, you need to upload to your private git repository a build script, the source code for your implementation, benchmark scripts, the report, a readme file, and 6 log files.

A detailed manual describing how the program works (README.md). The manual should be able to instruct users other than the developer to run the program step by step. The manual should contain example commands to invoke the benchmark. This should be included as README.md in the source code folder.

Here are the naming conventions for the required files:

- Makefile
- mysort.c
- hw5_report.pdf
- README.md

- mysort1GB.log
- mysort4GB.log
- mysort16GB.log
- mysort64GB.log

- linsort1GB.log
- linsort4GB.log
- linsort16GB.log
- linsort64GB.log

When you have finished implementing the complete assignment as described above, you should submit your solution to the blackboard. The timestamp on the BB submission will be used to determineif the submission is on time. Please put all your homework content into one .zip file and upload it to the blackboard.

The name of .zip should follow this format:
"***Section_LastName1_FirstName1-Section_LastName2_FirstName2.HW1.zip***"

Submit step #1: put all your files and documents for submission in a zip file.Submit step #2: Name the file correctly and submit it to the blackboard.

Grades for late programs will be lowered 10% per day late.