

PA01-Processes and Thread

GRS

Name-Arpit Kumar

Roll No-MT25017

Github Repository link-<https://github.com/Arpit2919/GraduateSystem>

1. Details of Worker Function Implementation

Based on the definitions given in the assignment, the following code implements the three necessary worker functions:

CPU Worker: Does computations on the CPU instead of waiting for additional resources. The CPU is essentially overloaded by your implementation's use of integer bit-shifting and complicated mathematical operations like sin and cos.

Memory Worker: RAM speed and capacity are bottlenecks. In order to surpass CPU cache efficiency, your code writes and reads from a large allocated block to transfer data between the CPU and memory.

The majority of an **I/O worker's** time is spent waiting for I/O operations. This is accomplished by using `fwrite`, `fread`, and `fsync` to read and write files to the disk, ensuring that the CPU is idle while the physical disk write is finished.

2. Automated Measurement & Observations:

A Bash script that enables the following was used to automate the data collection process:

CPU Utilization: Calculated in batch mode using the `top` command. The script calculates a total utilization metric for Program A by adding the CPU percentage of all matching child processes.

Disk Statistics: During the execution of I/O workers, read and write throughput in KB/s was recorded using `iostat`.

Execution Time: Determined by computing the difference between system timestamps prior to and following execution, or by using the `time` utility.

3. Analysis and Discussion

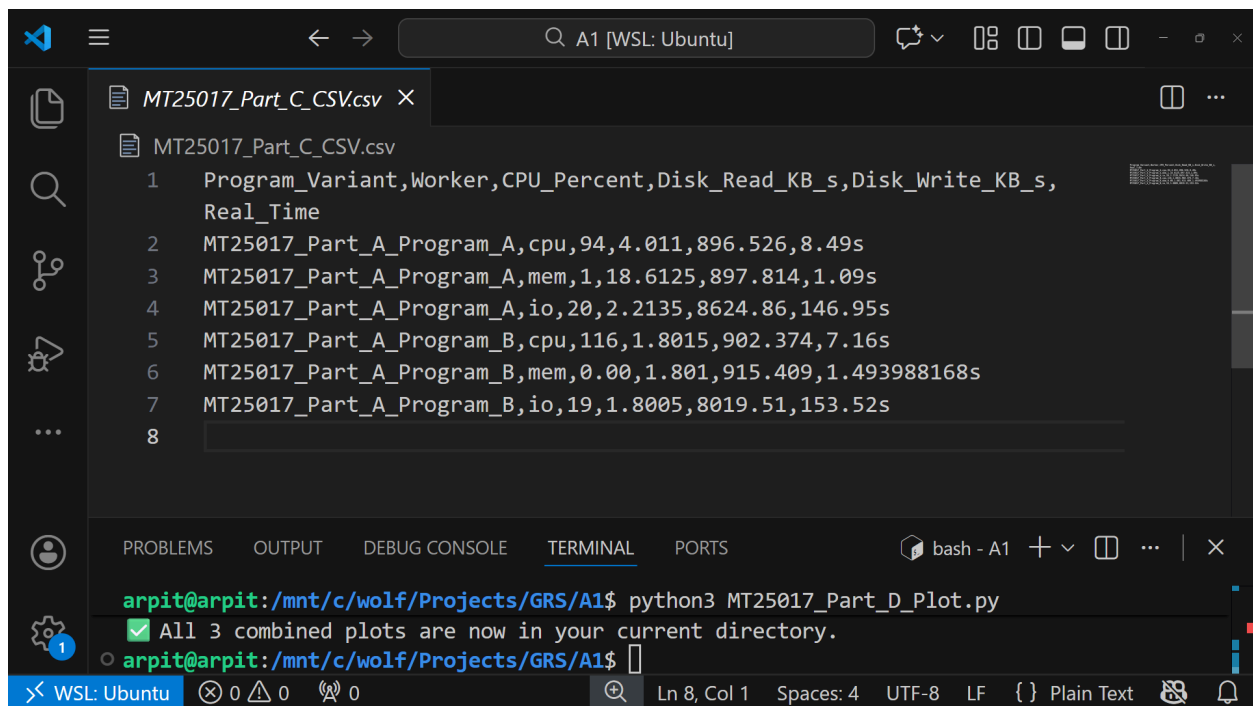
3.1 Comparative Analysis (Part C)

Based on the generated CSV data, the following trends were observed:

CPU Workers: The designated core was overloaded by both Program A and Program B. However, compared to Program A's entire process lifecycle, Program B (Threads) displayed somewhat shorter execution times, probably as a result of lower overhead in creation and context switching.

Memory Workers: Because the execution pipeline frequently stalled while awaiting data from the RAM subsystem, these tasks showed lower CPU percentages than the CPU worker.

I/O Workers: These variations demonstrated low CPU usage and high disk write/read metrics in iostat, indicating that the CPU is largely idle during hardware-bound operations.



The screenshot shows a VS Code editor window with a file named `MT25017_Part_C_CSV.csv` open. The file contains a CSV dataset with 8 rows. The first row is the header: `Program_Variant,Worker,CPU_Percent,Disk_Read_KB_s,Disk_Write_KB_s,Real_Time`. The subsequent rows show data for Program A and Program B across different workers (cpu, mem, io). The terminal window at the bottom shows the command `python3 MT25017_Part_D_Plot.py` being executed, and a message indicating that all 3 combined plots are now in the current directory.

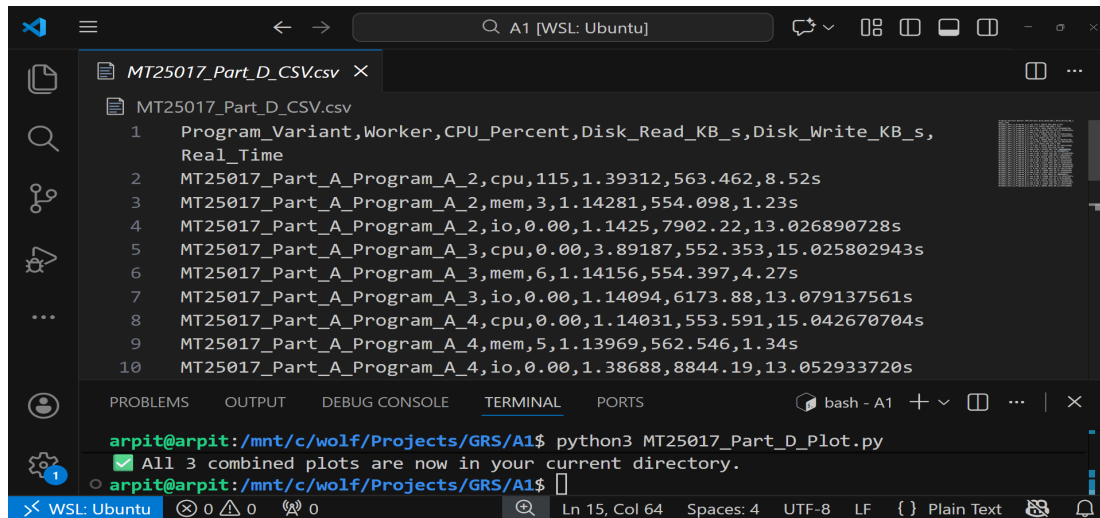
```
1 Program_Variant,Worker,CPU_Percent,Disk_Read_KB_s,Disk_Write_KB_s,Real_Time
2 MT25017_Part_A_Program_A,cpu,94,4.011,896.526,8.49s
3 MT25017_Part_A_Program_A,mem,1,18.6125,897.814,1.09s
4 MT25017_Part_A_Program_A,io,20,2.2135,8624.86,146.95s
5 MT25017_Part_A_Program_B,cpu,116,1.8015,902.374,7.16s
6 MT25017_Part_A_Program_B,mem,0.00,1.801,915.409,1.493988168s
7 MT25017_Part_A_Program_B,io,19,1.8005,8019.51,153.52s
8
```

```
arpit@arpit:/mnt/c/wolf/Projects/GRS/A1$ python3 MT25017_Part_D_Plot.py
✓ All 3 combined plots are now in your current directory.
arpit@arpit:/mnt/c/wolf/Projects/GRS/A1$
```

3.2 Comparative Analysis (Part D)

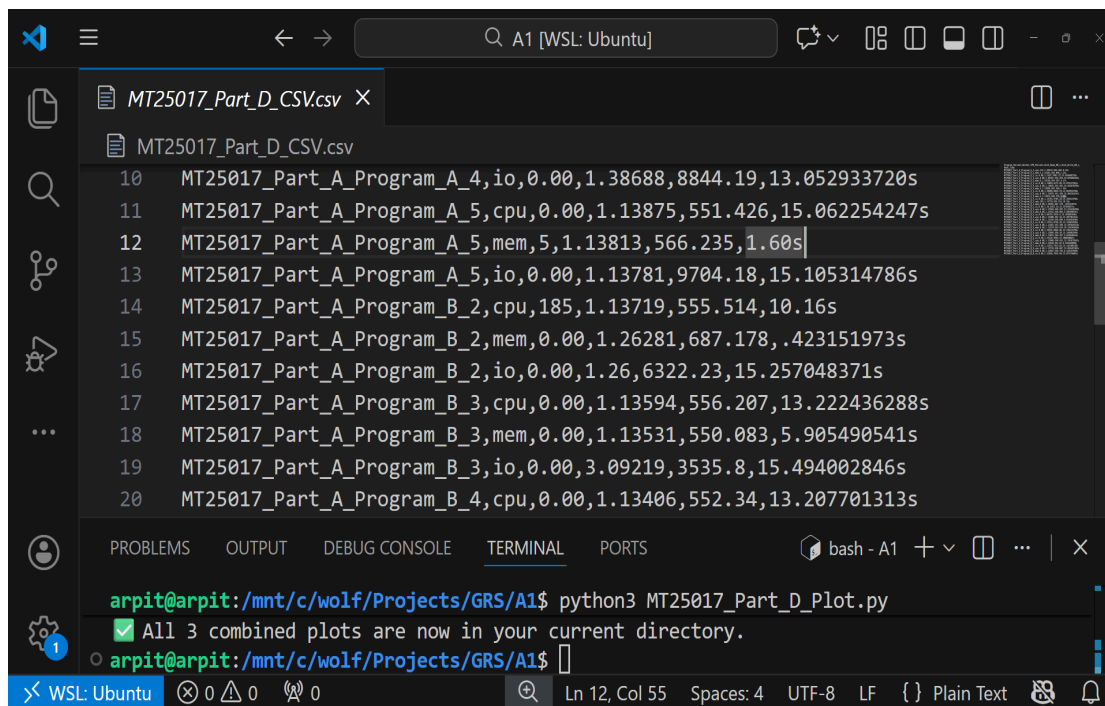
Based on the generated CSV data, the following trends were observed:

As the number of processes (2 to 5) and threads (2 to 8) was increased on a single core, total execution time increased linearly. This demonstrates resource contention where multiple workers must share the same physical execution time-slices on Core 0.



```
MT25017_Part_D_CSV.csv
1 Program_Variant,Worker,CPU_Percent,Disk_Read_KB_s,Disk_Write_KB_s,Real_Time
2 MT25017_Part_A_Program_A_2,cpu,115,1.39312,563.462,8.52s
3 MT25017_Part_A_Program_A_2,mem,3,1.14281,554.098,1.23s
4 MT25017_Part_A_Program_A_2,io,0.00,1.1425,7902.22,13.026890728s
5 MT25017_Part_A_Program_A_3,cpu,0.00,3.89187,552.353,15.025802943s
6 MT25017_Part_A_Program_A_3,mem,6,1.14156,554.397,4.27s
7 MT25017_Part_A_Program_A_3,io,0.00,1.14094,6173.88,13.079137561s
8 MT25017_Part_A_Program_A_4,cpu,0.00,1.14031,553.591,15.042670704s
9 MT25017_Part_A_Program_A_4,mem,5,1.13969,562.546,1.34s
10 MT25017_Part_A_Program_A_4,io,0.00,1.38688,8844.19,13.052933720s

arpit@arpit:/mnt/c/wolf/Projects/GRS/A1$ python3 MT25017_Part_D_Plot.py
All 3 combined plots are now in your current directory.
arpit@arpit:/mnt/c/wolf/Projects/GRS/A1$
```



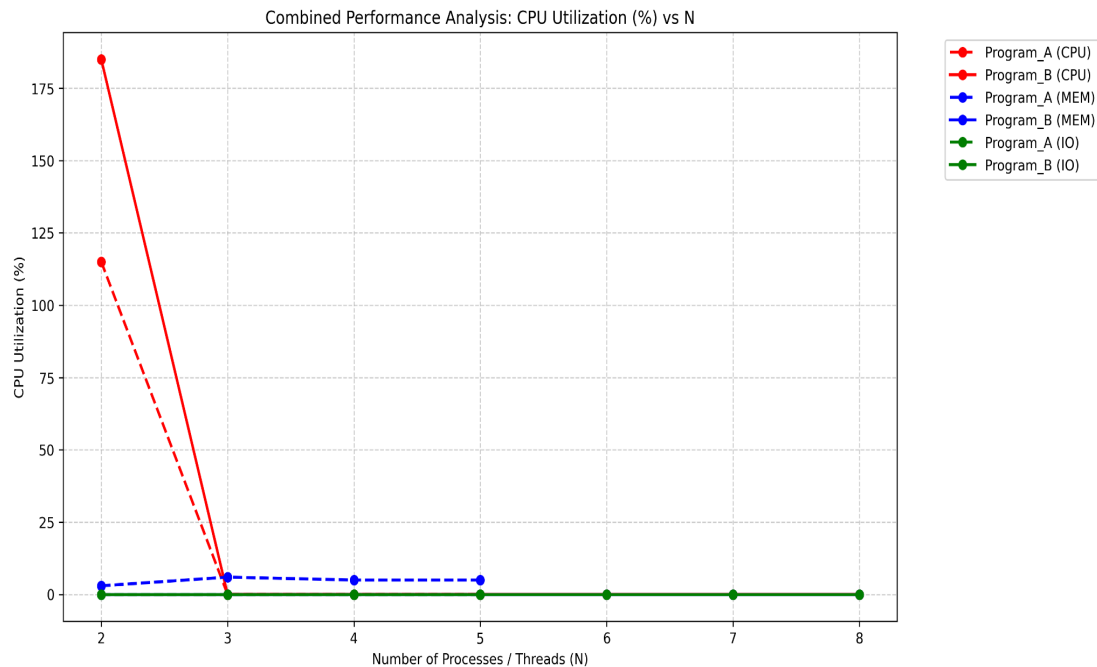
```
MT25017_Part_D_CSV.csv
10 MT25017_Part_A_Program_A_4,io,0.00,1.38688,8844.19,13.052933720s
11 MT25017_Part_A_Program_A_5,cpu,0.00,1.13875,551.426,15.062254247s
12 MT25017_Part_A_Program_A_5,mem,5,1.13813,566.235,1.60s
13 MT25017_Part_A_Program_A_5,io,0.00,1.13781,9704.18,15.105314786s
14 MT25017_Part_A_Program_B_2,cpu,185,1.13719,555.514,10.16s
15 MT25017_Part_A_Program_B_2,mem,0.00,1.26281,687.178,.423151973s
16 MT25017_Part_A_Program_B_2,io,0.00,1.26,6322.23,15.257048371s
17 MT25017_Part_A_Program_B_3,cpu,0.00,1.13594,556.207,13.222436288s
18 MT25017_Part_A_Program_B_3,mem,0.00,1.13531,550.083,5.905490541s
19 MT25017_Part_A_Program_B_3,io,0.00,3.09219,3535.8,15.494002846s
20 MT25017_Part_A_Program_B_4,cpu,0.00,1.13406,552.34,13.207701313s

arpit@arpit:/mnt/c/wolf/Projects/GRS/A1$ python3 MT25017_Part_D_Plot.py
All 3 combined plots are now in your current directory.
arpit@arpit:/mnt/c/wolf/Projects/GRS/A1$
```

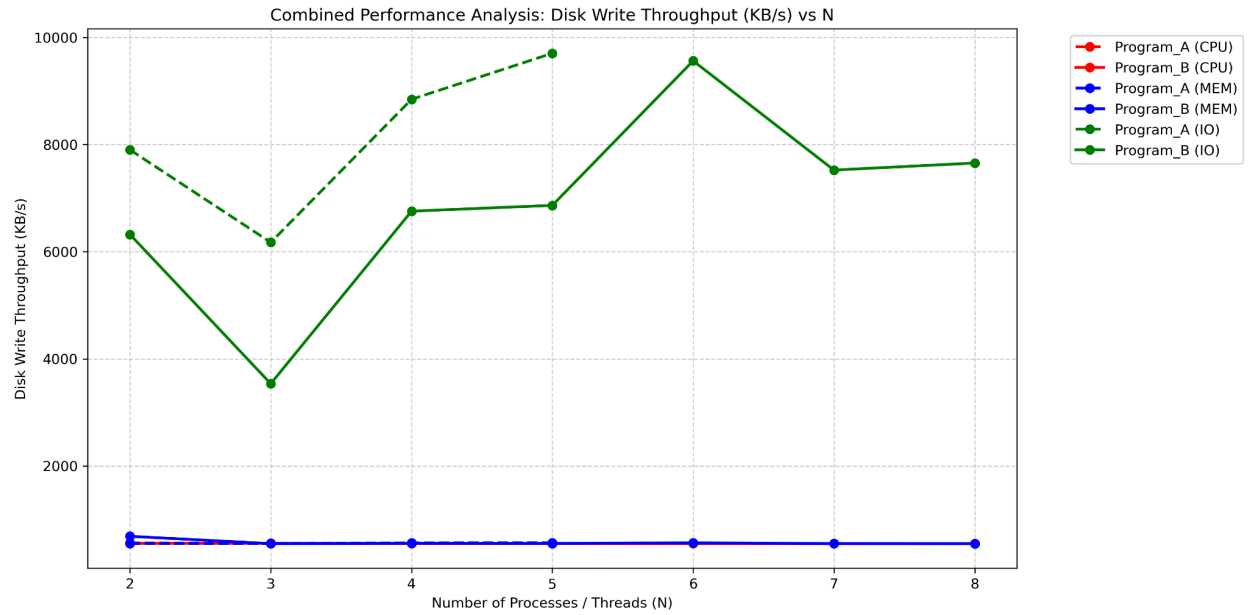
```
MT25017_Part_D_CSV.csv X
MT25017_Part_D_CSV.csv
21 MT25017_Part_A_Program_B_4,mem,0.00,1.13344,556.869,4.824582649s
22 MT25017_Part_A_Program_B_4,io,0.00,1.13312,6756.46,13.143023938s
23 MT25017_Part_A_Program_B_5,cpu,0.00,1.13281,552.349,15.207629118s
24 MT25017_Part_A_Program_B_5,mem,0.00,1.13219,554.918,10.154303658s
25 MT25017_Part_A_Program_B_5,io,0.00,1.25531,6864.86,13.193113795s
26 MT25017_Part_A_Program_B_6,cpu,0.00,1.13094,550.39,15.149402589s
27 MT25017_Part_A_Program_B_6,mem,0.00,1.13031,564.387,3.829840544s
28 MT25017_Part_A_Program_B_6,io,0.00,1.25125,9563,15.300840067s
29 MT25017_Part_A_Program_B_7,cpu,0.00,1.12906,549.832,13.291073567s
30 MT25017_Part_A_Program_B_7,mem,0.00,1.12844,551.62,6.923426090s
31 MT25017_Part_A_Program_B_7,io,0.00,1.12813,7523.86,15.142200130s

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS bash - A1 + v [ ] ... | X
arpit@arpit:/mnt/c/wolf/Projects/GRS/A1$ python3 MT25017_Part_D_Plot.py
✓ All 3 combined plots are now in your current directory.
arpit@arpit:/mnt/c/wolf/Projects/GRS/A1$
```

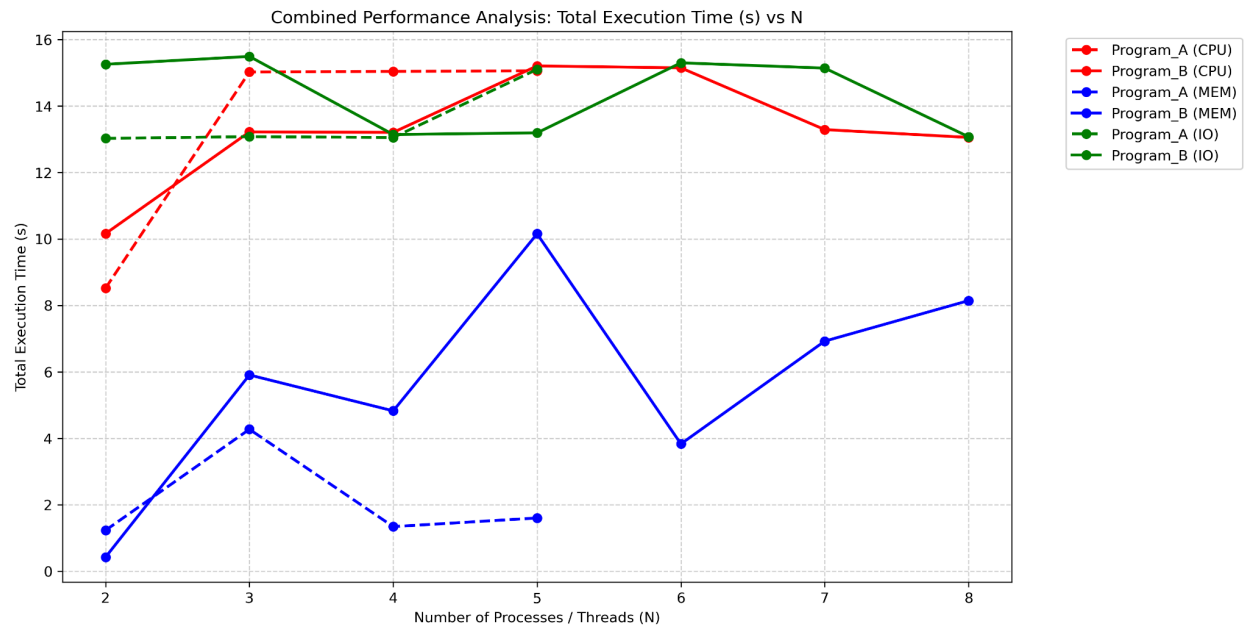
3.3 CPU Utilization Analysis



3.4 Disk Throughput Analysis



3.5 Execution Time Scaling



4. AI Usage Declaration:

I hereby declare that the following components of this project were developed using artificial intelligence (Gemini 3 Flash) in compliance with the assignment guidelines:

Shell Script Generation: AI was utilized to create the automation logic for MT25017_Part_C_shell.sh and MT25017_Part_D_shell.sh, with a focus on capturing and summarizing real-time data from the iostat and top utilities.

Plot Generation: AI was used in the development of the Python script that processed the CSV data and produced the combined performance graphs.

Debugging: AI helped solve technical problems, like managing memory allocation safety in multi-threaded variants and capturing child process CPU utilization in multi-processing variants.