Graduate System (CSE638)

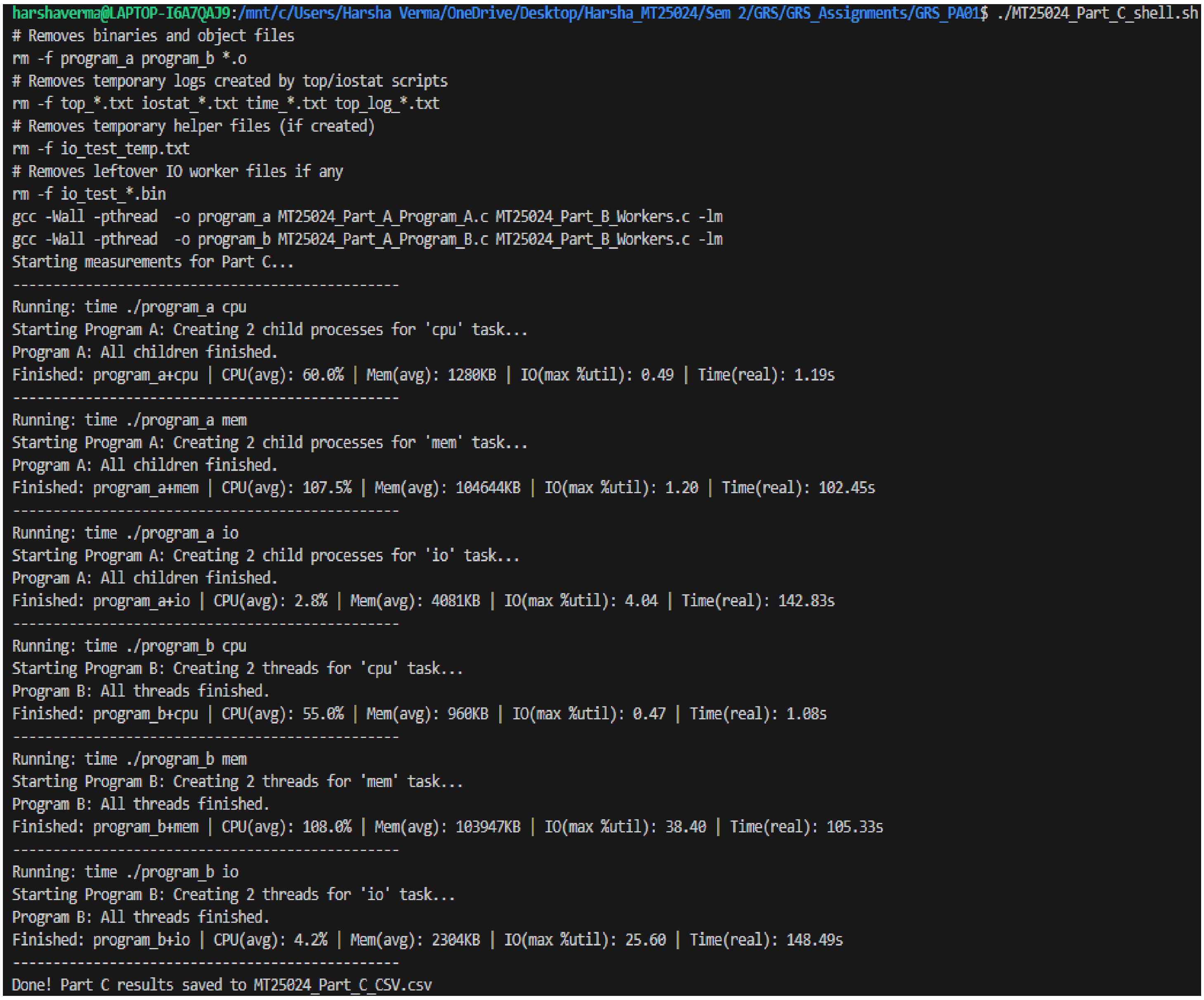
Name: Harsha Verma
Roll No: MT25024

Part C: Observation and Analysis:

1. Experimental Setup:

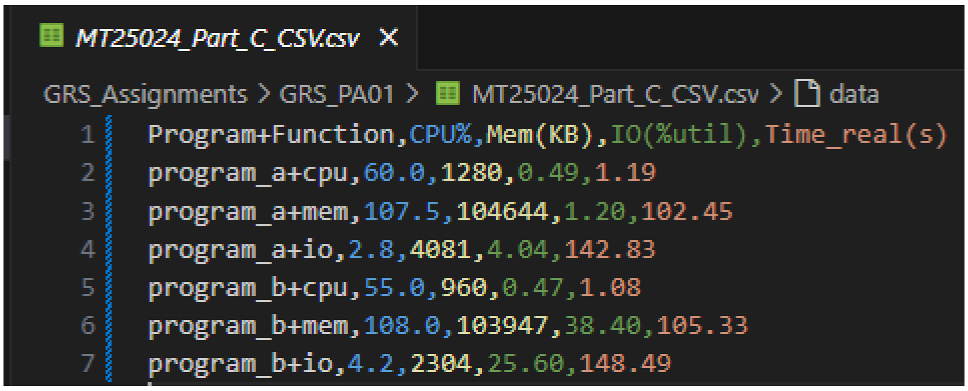
- Employed taskset in order to bind the running programs to a single CPU core.
- Gather CPU and Memory statistics data by running the top command
- Retrieved disk usage data via iostat
- Automated execution by bash script

2. Screenshots:



```
harshaverma@LAPTOP-I6A7QAJ9:/mnt/c/Users/Harsha Verma/OneDrive/Desktop/Harsha_MT25024/Sem 2/GRS/GRS_Assignments/GRS_PA01$ ./MT25024_Part_C_shell.sh
# Removes binaries and object files
rm -f program_a program_b *.o
# Removes temporary logs created by top/iostat scripts
rm -f top_*.txt iostat_*.txt time_*.txt top_log_*.txt
# Removes temporary helper files (if created)
rm -f io_test_temp.txt
# Removes leftover IO worker files if any
rm -f io_test_*.bin
gcc -Wall -pthread  -o program_a MT25024_Part_A_Program_A.c MT25024_Part_B_Workers.c -lm
gcc -Wall -pthread  -o program_b MT25024_Part_A_Program_B.c MT25024_Part_B_Workers.c -lm
Starting measurements for Part C...
------------------------------------------------
Running: time ./program_a cpu
Starting Program A: Creating 2 child processes for 'cpu' task...
Program A: All children finished.
Finished: program_a+cpu | CPU(avg): 60.0% | Mem(avg): 1280KB | IO(max %util): 0.49 | Time(real): 1.19s
------------------------------------------------
Running: time ./program_a mem
Starting Program A: Creating 2 child processes for 'mem' task...
Program A: All children finished.
Finished: program_a+mem | CPU(avg): 107.5% | Mem(avg): 104644KB | IO(max %util): 1.20 | Time(real): 102.45s
------------------------------------------------
Running: time ./program_a io
Starting Program A: Creating 2 child processes for 'io' task...
Program A: All children finished.
Finished: program_a+io | CPU(avg): 2.8% | Mem(avg): 4081KB | IO(max %util): 4.04 | Time(real): 142.83s
------------------------------------------------
Running: time ./program_b cpu
Starting Program B: Creating 2 threads for 'cpu' task...
Program B: All threads finished.
Finished: program_b+cpu | CPU(avg): 55.0% | Mem(avg): 960KB | IO(max %util): 0.47 | Time(real): 1.08s
------------------------------------------------
Running: time ./program_b mem
Starting Program B: Creating 2 threads for 'mem' task...
Program B: All threads finished.
Finished: program_b+mem | CPU(avg): 108.0% | Mem(avg): 103947KB | IO(max %util): 38.40 | Time(real): 105.33s
------------------------------------------------
Running: time ./program_b io
Starting Program B: Creating 2 threads for 'io' task...
Program B: All threads finished.
Finished: program_b+io | CPU(avg): 4.2% | Mem(avg): 2304KB | IO(max %util): 25.60 | Time(real): 148.49s
------------------------------------------------
Done! Part C results saved to MT25024_Part_C_CSV.csv
```

Figure 1: Terminal output showing automated execution and resource usage



MT25024_Part_C_CSV.csv ×

GRS_Assignments > GRS_PA01 > MT25024_Part_C_CSV.csv > data

```
Program+Function,CPU%,Mem(KB),IO(%util),Time_real(s)
program_a+cpu,60.0,1280,0.49,1.19
program_a+mem,107.5,104644,1.20,102.45
program_a+io,2.8,4081,4.04,142.83
program_b+cpu,55.0,960,0.47,1.08
program_b+mem,108.0,103947,38.40,105.33
program_b+io,4.2,2304,25.60,148.49
```

Figure 2: Generated CSV file containing summarized metrics

3. Summary Table:

Program + Function	CPU(%)	Memory (KB)	IO(%util)
Program A+CPU	60.0	1280	0.49
Program A+Mem	107.5	104644	1.20
Program A+IO	2.8	4081	4.04
Program B+CPU	55.0	960	0.47
Program B+Mem	108.0	103947	38.40
Program B+IO	4.2	2304	25.60

Table 1: Resource Utilization Summary

4. Analysis of Part C:

Six different combinations of program and workload: Program A (processes) and Program B (threads) with cpu, mem, and io worker functions following all the experimental setup.

4.1.CPU-intensive(cpu)

- Program A + cpu:
  CPU usage is 60%, while memory and I/O usage are very low.
- Program B + cpu:
  CPU usage is around 55%, lower than Program A,with similar low memory and io usage.

  Explanation:

  The cpu() worker function is engaged in constant mathematical calculation and therefore consumes large CPU resources but very minimal memory and disk I/O.

  Program A creates two child processes that are pinned to single CPU core due to taskset. Processes share CPU in a time slicing manner hence a relatively higher utilization ('top' aggregates CPU utilization across processes)

  Program B uses threads, and they all share the same address space and execution context.

  'top' reports CPU utilization at process level (program A has 2 child processes) hence the difference in cpu utilization in both programs.

4.2.Memory-intensive(mem)

- Program A + mem:
  Memory usage is close to 105MB, with CPU usage slightly above 100%.
- Program B + mem:
- Memory usage is similar(104MB), with CPU usage slightly above 100%.
  (Both exceed 100% CPU util because memory access still consumes CPU cycles, and top aggregates CPU time across concurrent workers during sampling.)

  Explanation:

  The mem() worker allocates a large memory buffer of 50 MB per worker and continuously accesses it. As both programs spawn two workers, the resulting memory usage is roughly doubled.

  Both processes and threads exhibit similar memory behaviour because of the fact that:

  In program A each process is allocated separate memory.

  In program B, even though the threads in a process share the address space, each thread allocates its own buffer independently.

  The CPU usage is still high as frequent memory access and cache misses have to be actively participated by the CPU. Disk I/O remains low because of memory operations.

4.3.I/O-intensive(io)

- Program A + io:
  CPU usage is very low(2.8%), memory usage is small(4081 KB), and disk utilization is moderate(4%).
- Program B + io:
  CPU usage is very low(4.2%),with slightly lower memory usage (2304 KB) due to shared address space, while disk utilization is higher (25.6%)

Explanation:

The io() worker repeatedly writes to a file. In disk I/O, the process/thread normally waits for the completion of I/O thus CPU usage remains very low. This suggests that the workload is I/O bound.

Program B shows higher disk utilisation likely because multiple threads issuing same io requests at the same time. This leads to higher io util observed by iostat.

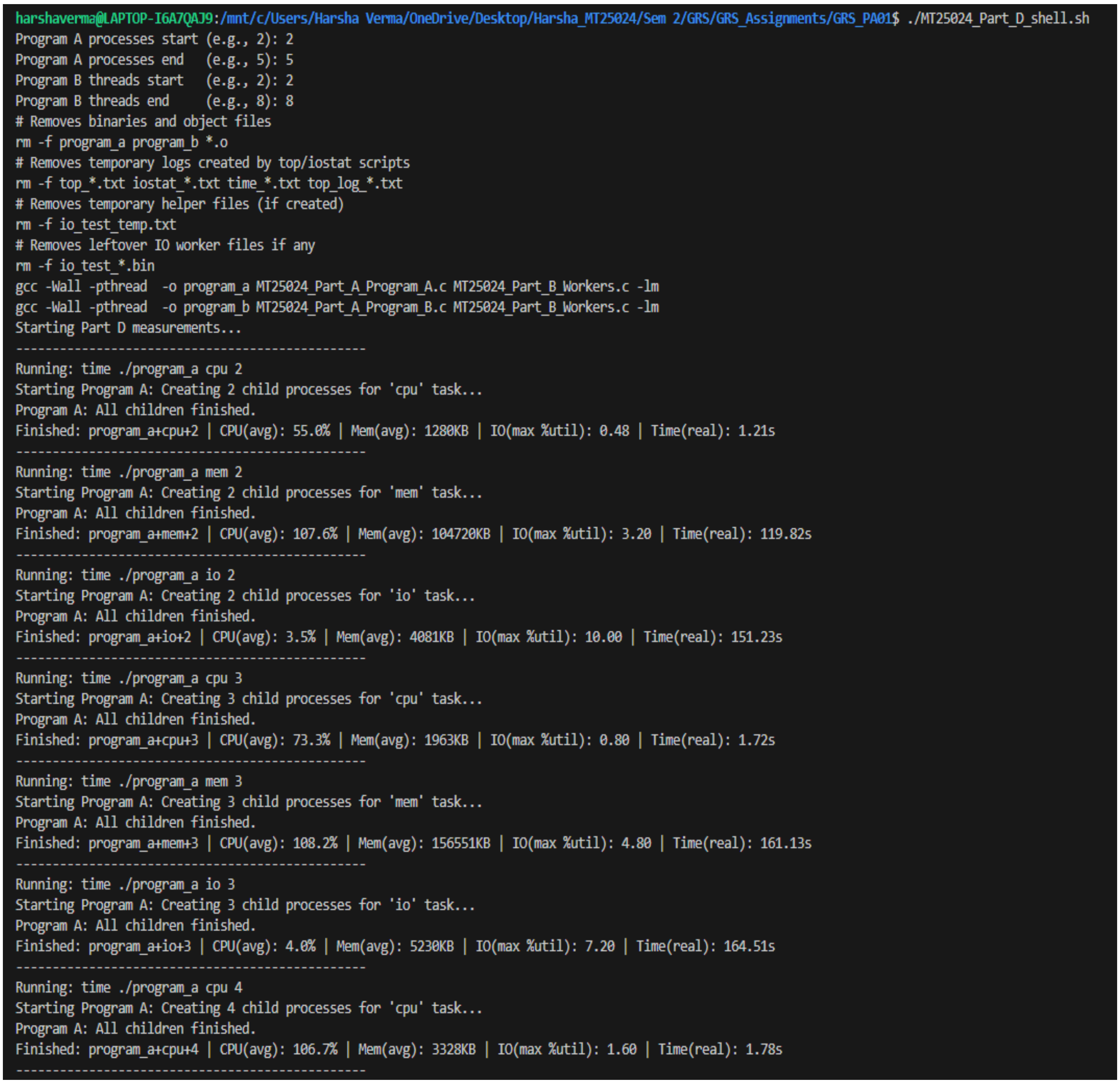
Threads exhibit somewhat lesser consumption of memory because of shared address space, whereas processes have separate memory overheads.

Analysis of Part D:

1. Introduction:

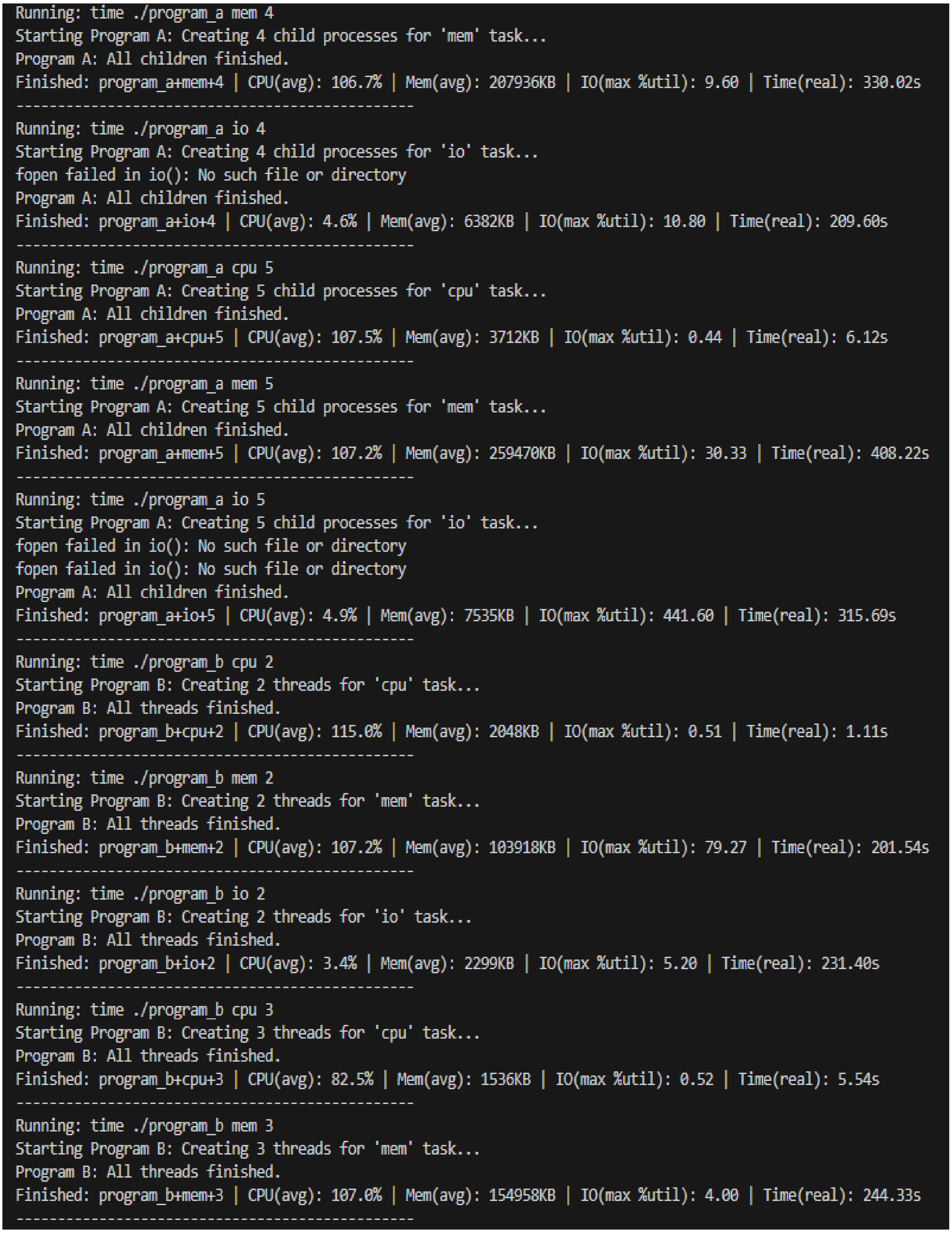
Part C was scaled up in part D. In part C we made use of Part_A_Program_A to create 2 child process and Part_A_Program_B to create 2 threads with the 3 worker functions. In part D we are scaling up the number of child process and threads with the 3 worker functions to study resource utilization.

2. Screenshots:



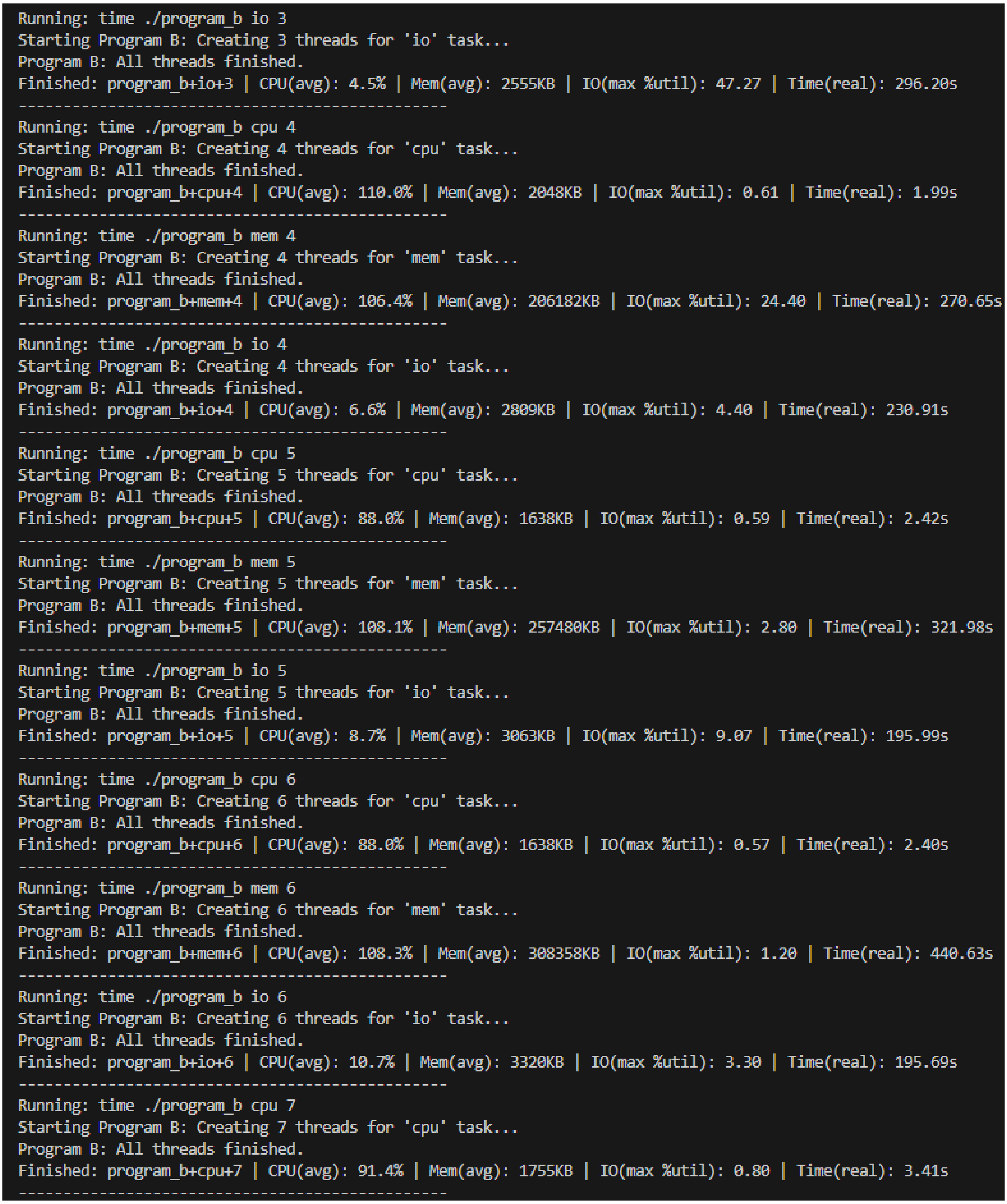
```
harshaverma@LAPTOP-I6A7QAJ9:/mnt/c/Users/Harsha Verma/OneDrive/Desktop/Harsha_MT25024/Sem 2/GRS/GRS_Assignments/GRS_PA01$ ./MT25024_Part_D_shell.sh
Program A processes start (e.g., 2): 2
Program A processes end   (e.g., 5): 5
Program B threads start   (e.g., 2): 2
Program B threads end     (e.g., 8): 8
# Removes binaries and object files
rm -f program_a program_b *.o
# Removes temporary logs created by top/iostat scripts
rm -f top_*.txt iostat_*.txt time_*.txt top_log_*.txt
# Removes temporary helper files (if created)
rm -f io_test_temp.txt
# Removes leftover IO worker files if any
rm -f io_test_*.bin
gcc -Wall -pthread  -o program_a MT25024_Part_A_Program_A.c MT25024_Part_B_Workers.c -lm
gcc -Wall -pthread  -o program_b MT25024_Part_A_Program_B.c MT25024_Part_B_Workers.c -lm
Starting Part D measurements...
------------------------------------------------
Running: time ./program_a cpu 2
Starting Program A: Creating 2 child processes for 'cpu' task...
Program A: All children finished.
Finished: program_a+cpu+2 | CPU(avg): 55.0% | Mem(avg): 1280KB | IO(max %util): 0.48 | Time(real): 1.21s
------------------------------------------------
Running: time ./program_a mem 2
Starting Program A: Creating 2 child processes for 'mem' task...
Program A: All children finished.
Finished: program_a+mem+2 | CPU(avg): 107.6% | Mem(avg): 104720KB | IO(max %util): 3.20 | Time(real): 119.82s
------------------------------------------------
Running: time ./program_a io 2
Starting Program A: Creating 2 child processes for 'io' task...
Program A: All children finished.
Finished: program_a+io+2 | CPU(avg): 3.5% | Mem(avg): 4081KB | IO(max %util): 10.00 | Time(real): 151.23s
------------------------------------------------
Running: time ./program_a cpu 3
Starting Program A: Creating 3 child processes for 'cpu' task...
Program A: All children finished.
Finished: program_a+cpu+3 | CPU(avg): 73.3% | Mem(avg): 1963KB | IO(max %util): 0.80 | Time(real): 1.72s
------------------------------------------------
Running: time ./program_a mem 3
Starting Program A: Creating 3 child processes for 'mem' task...
Program A: All children finished.
Finished: program_a+mem+3 | CPU(avg): 108.2% | Mem(avg): 156551KB | IO(max %util): 4.80 | Time(real): 161.13s
------------------------------------------------
Running: time ./program_a io 3
Starting Program A: Creating 3 child processes for 'io' task...
Program A: All children finished.
Finished: program_a+io+3 | CPU(avg): 4.0% | Mem(avg): 5230KB | IO(max %util): 7.20 | Time(real): 164.51s
------------------------------------------------
Running: time ./program_a cpu 4
Starting Program A: Creating 4 child processes for 'cpu' task...
Program A: All children finished.
Finished: program_a+cpu+4 | CPU(avg): 106.7% | Mem(avg): 3328KB | IO(max %util): 1.60 | Time(real): 1.78s
------------------------------------------------
```

Figure 3.1: Terminal output showing output of part d



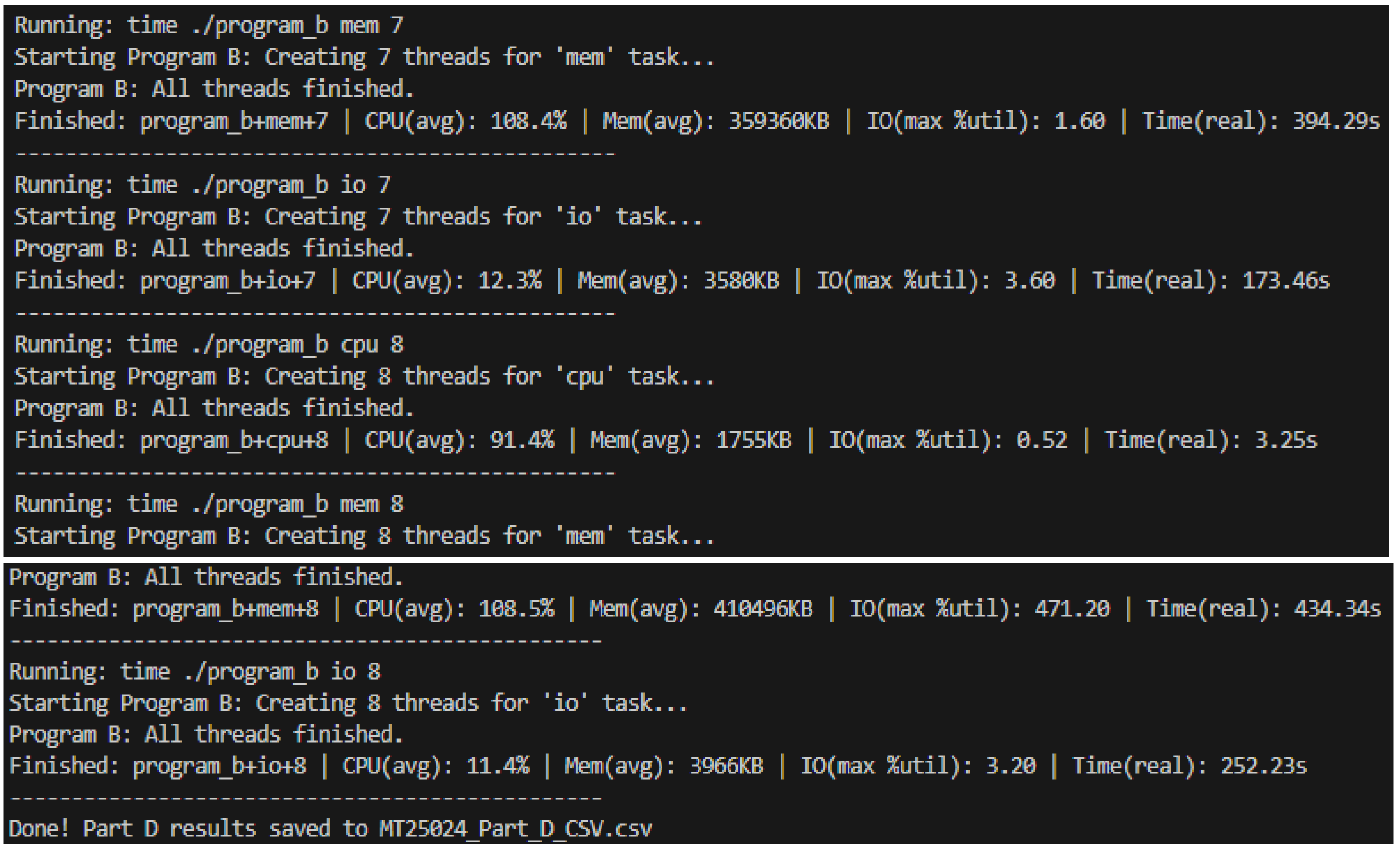
```
Running: time ./program_a mem 4
Starting Program A: Creating 4 child processes for 'mem' task...
Program A: All children finished.
Finished: program_a+mem+4 | CPU(avg): 106.7% | Mem(avg): 207936KB | IO(max %util): 9.60 | Time(real): 330.02s
------------------------------------------------
Running: time ./program_a io 4
Starting Program A: Creating 4 child processes for 'io' task...
fopen failed in io(): No such file or directory
Program A: All children finished.
Finished: program_a+io+4 | CPU(avg): 4.6% | Mem(avg): 6382KB | IO(max %util): 10.80 | Time(real): 209.60s
------------------------------------------------
Running: time ./program_a cpu 5
Starting Program A: Creating 5 child processes for 'cpu' task...
Program A: All children finished.
Finished: program_a+cpu+5 | CPU(avg): 107.5% | Mem(avg): 3712KB | IO(max %util): 0.44 | Time(real): 6.12s
------------------------------------------------
Running: time ./program_a mem 5
Starting Program A: Creating 5 child processes for 'mem' task...
Program A: All children finished.
Finished: program_a+mem+5 | CPU(avg): 107.2% | Mem(avg): 259470KB | IO(max %util): 30.33 | Time(real): 408.22s
------------------------------------------------
Running: time ./program_a io 5
Starting Program A: Creating 5 child processes for 'io' task...
fopen failed in io(): No such file or directory
fopen failed in io(): No such file or directory
Program A: All children finished.
Finished: program_a+io+5 | CPU(avg): 4.9% | Mem(avg): 7535KB | IO(max %util): 441.60 | Time(real): 315.69s
------------------------------------------------
Running: time ./program_b cpu 2
Starting Program B: Creating 2 threads for 'cpu' task...
Program B: All threads finished.
Finished: program_b+cpu+2 | CPU(avg): 115.0% | Mem(avg): 2048KB | IO(max %util): 0.51 | Time(real): 1.11s
------------------------------------------------
Running: time ./program_b mem 2
Starting Program B: Creating 2 threads for 'mem' task...
Program B: All threads finished.
Finished: program_b+mem+2 | CPU(avg): 107.2% | Mem(avg): 103918KB | IO(max %util): 79.27 | Time(real): 201.54s
------------------------------------------------
Running: time ./program_b io 2
Starting Program B: Creating 2 threads for 'io' task...
Program B: All threads finished.
Finished: program_b+io+2 | CPU(avg): 3.4% | Mem(avg): 2299KB | IO(max %util): 5.20 | Time(real): 231.40s
------------------------------------------------
Running: time ./program_b cpu 3
Starting Program B: Creating 3 threads for 'cpu' task...
Program B: All threads finished.
Finished: program_b+cpu+3 | CPU(avg): 82.5% | Mem(avg): 1536KB | IO(max %util): 0.52 | Time(real): 5.54s
------------------------------------------------
Running: time ./program_b mem 3
Starting Program B: Creating 3 threads for 'mem' task...
Program B: All threads finished.
Finished: program_b+mem+3 | CPU(avg): 107.0% | Mem(avg): 154958KB | IO(max %util): 4.00 | Time(real): 244.33s
------------------------------------------------
```

Figure 3.2: Terminal output showing output of part d



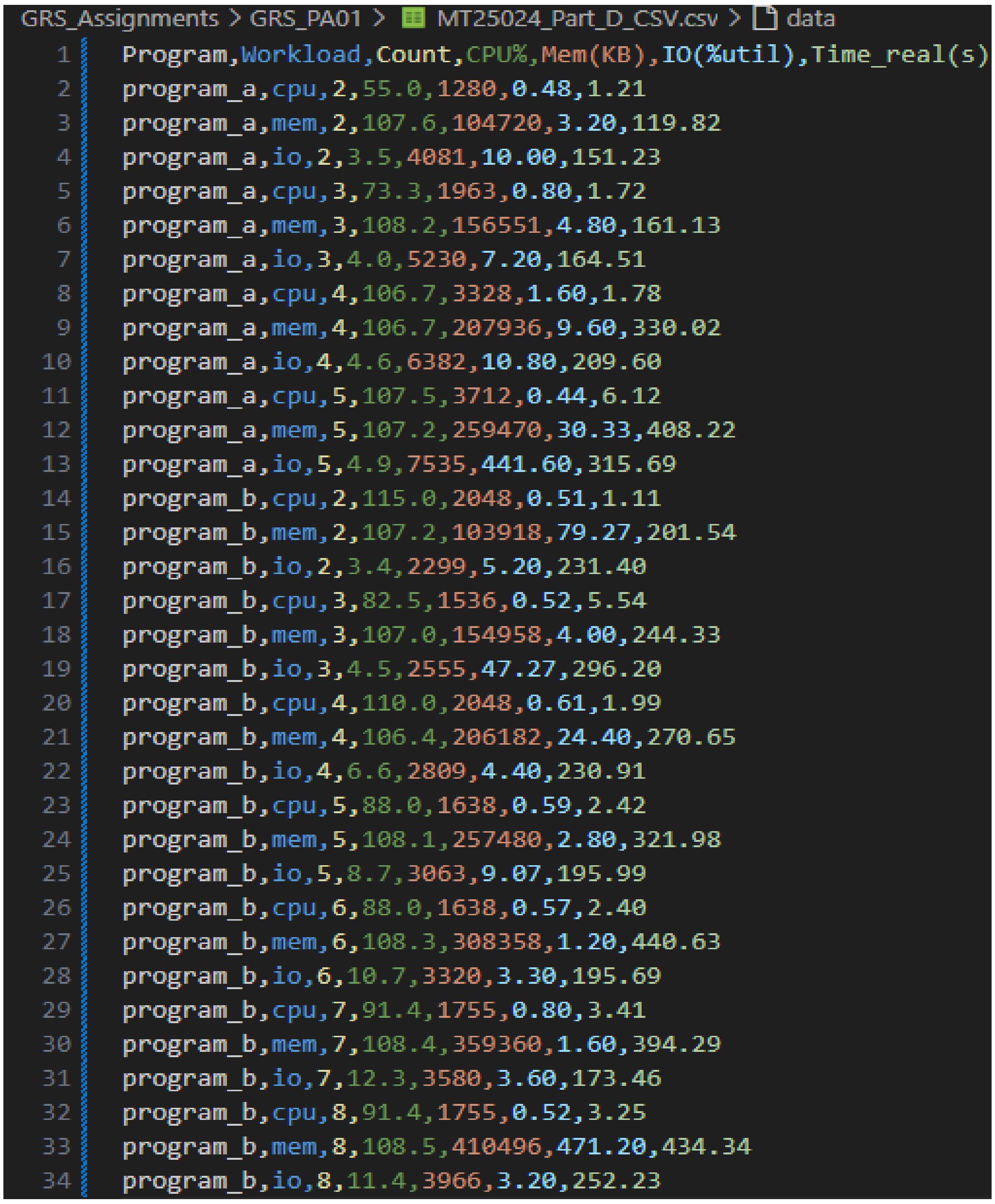
```
Running: time ./program_b io 3
Starting Program B: Creating 3 threads for 'io' task...
Program B: All threads finished.
Finished: program_b+io+3 | CPU(avg): 4.5% | Mem(avg): 2555KB | IO(max %util): 47.27 | Time(real): 296.20s
------------------------------------------------
Running: time ./program_b cpu 4
Starting Program B: Creating 4 threads for 'cpu' task...
Program B: All threads finished.
Finished: program_b+cpu+4 | CPU(avg): 110.0% | Mem(avg): 2048KB | IO(max %util): 0.61 | Time(real): 1.99s
------------------------------------------------
Running: time ./program_b mem 4
Starting Program B: Creating 4 threads for 'mem' task...
Program B: All threads finished.
Finished: program_b+mem+4 | CPU(avg): 106.4% | Mem(avg): 206182KB | IO(max %util): 24.40 | Time(real): 270.65s
------------------------------------------------
Running: time ./program_b io 4
Starting Program B: Creating 4 threads for 'io' task...
Program B: All threads finished.
Finished: program_b+io+4 | CPU(avg): 6.6% | Mem(avg): 2809KB | IO(max %util): 4.40 | Time(real): 230.91s
------------------------------------------------
Running: time ./program_b cpu 5
Starting Program B: Creating 5 threads for 'cpu' task...
Program B: All threads finished.
Finished: program_b+cpu+5 | CPU(avg): 88.0% | Mem(avg): 1638KB | IO(max %util): 0.59 | Time(real): 2.42s
------------------------------------------------
Running: time ./program_b mem 5
Starting Program B: Creating 5 threads for 'mem' task...
Program B: All threads finished.
Finished: program_b+mem+5 | CPU(avg): 108.1% | Mem(avg): 257480KB | IO(max %util): 2.80 | Time(real): 321.98s
------------------------------------------------
Running: time ./program_b io 5
Starting Program B: Creating 5 threads for 'io' task...
Program B: All threads finished.
Finished: program_b+io+5 | CPU(avg): 8.7% | Mem(avg): 3063KB | IO(max %util): 9.07 | Time(real): 195.99s
------------------------------------------------
Running: time ./program_b cpu 6
Starting Program B: Creating 6 threads for 'cpu' task...
Program B: All threads finished.
Finished: program_b+cpu+6 | CPU(avg): 88.0% | Mem(avg): 1638KB | IO(max %util): 0.57 | Time(real): 2.40s
------------------------------------------------
Running: time ./program_b mem 6
Starting Program B: Creating 6 threads for 'mem' task...
Program B: All threads finished.
Finished: program_b+mem+6 | CPU(avg): 108.3% | Mem(avg): 308358KB | IO(max %util): 1.20 | Time(real): 440.63s
------------------------------------------------
Running: time ./program_b io 6
Starting Program B: Creating 6 threads for 'io' task...
Program B: All threads finished.
Finished: program_b+io+6 | CPU(avg): 10.7% | Mem(avg): 3320KB | IO(max %util): 3.30 | Time(real): 195.69s
------------------------------------------------
Running: time ./program_b cpu 7
Starting Program B: Creating 7 threads for 'cpu' task...
Program B: All threads finished.
Finished: program_b+cpu+7 | CPU(avg): 91.4% | Mem(avg): 1755KB | IO(max %util): 0.80 | Time(real): 3.41s
------------------------------------------------
```

Figure 3.3: Terminal output showing output of part d



```
Running: time ./program_b mem 7
Starting Program B: Creating 7 threads for 'mem' task...
Program B: All threads finished.
Finished: program_b+mem+7 | CPU(avg): 108.4% | Mem(avg): 359360KB | IO(max %util): 1.60 | Time(real): 394.29s
----------------------------------------------
Running: time ./program_b io 7
Starting Program B: Creating 7 threads for 'io' task...
Program B: All threads finished.
Finished: program_b+io+7 | CPU(avg): 12.3% | Mem(avg): 3580KB | IO(max %util): 3.60 | Time(real): 173.46s
----------------------------------------------
Running: time ./program_b cpu 8
Starting Program B: Creating 8 threads for 'cpu' task...
Program B: All threads finished.
Finished: program_b+cpu+8 | CPU(avg): 91.4% | Mem(avg): 1755KB | IO(max %util): 0.52 | Time(real): 3.25s
----------------------------------------------
Running: time ./program_b mem 8
Starting Program B: Creating 8 threads for 'mem' task...
Program B: All threads finished.
Finished: program_b+mem+8 | CPU(avg): 108.5% | Mem(avg): 410496KB | IO(max %util): 471.20 | Time(real): 434.34s
----------------------------------------------
Running: time ./program_b io 8
Starting Program B: Creating 8 threads for 'io' task...
Program B: All threads finished.
Finished: program_b+io+8 | CPU(avg): 11.4% | Mem(avg): 3966KB | IO(max %util): 3.20 | Time(real): 252.23s
----------------------------------------------
Done! Part D results saved to MT25024_Part_D_CSV.csv
```

Figure 3.4: Terminal output showing output of part d



GRS_Assignments > GRS_PA01 > MT25024_Part_D_CSV.csv > data

```
Program,Workload,Count,CPU%,Mem(KB),IO(%util),Time_real(s)
program_a,cpu,2,55.0,1280,0.48,1.21
program_a,mem,2,107.6,104720,3.20,119.82
program_a,io,2,3.5,4081,10.00,151.23
program_a,cpu,3,73.3,1963,0.80,1.72
program_a,mem,3,108.2,156551,4.80,161.13
program_a,io,3,4.0,5230,7.20,164.51
program_a,cpu,4,106.7,3328,1.60,1.78
program_a,mem,4,106.7,207936,9.60,330.02
program_a,io,4,4.6,6382,10.80,209.60
program_a,cpu,5,107.5,3712,0.44,6.12
program_a,mem,5,107.2,259470,30.33,408.22
program_a,io,5,4.9,7535,441.60,315.69
program_b,cpu,2,115.0,2048,0.51,1.11
program_b,mem,2,107.2,103918,79.27,201.54
program_b,io,2,3.4,2299,5.20,231.40
program_b,cpu,3,82.5,1536,0.52,5.54
program_b,mem,3,107.0,154958,4.00,244.33
program_b,io,3,4.5,2555,47.27,296.20
program_b,cpu,4,110.0,2048,0.61,1.99
program_b,mem,4,106.4,206182,24.40,270.65
program_b,io,4,6.6,2809,4.40,230.91
program_b,cpu,5,88.0,1638,0.59,2.42
program_b,mem,5,108.1,257480,2.80,321.98
program_b,io,5,8.7,3063,9.07,195.99
program_b,cpu,6,88.0,1638,0.57,2.40
program_b,mem,6,108.3,308358,1.20,440.63
program_b,io,6,10.7,3320,3.30,195.69
program_b,cpu,7,91.4,1755,0.80,3.41
program_b,mem,7,108.4,359360,1.60,394.29
program_b,io,7,12.3,3580,3.60,173.46
program_b,cpu,8,91.4,1755,0.52,3.25
program_b,mem,8,108.5,410496,471.20,434.34
program_b,io,8,11.4,3966,3.20,252.23
```

Figure 4: Generated CSV file containing summarized metrics

3. Analysis and discussion:

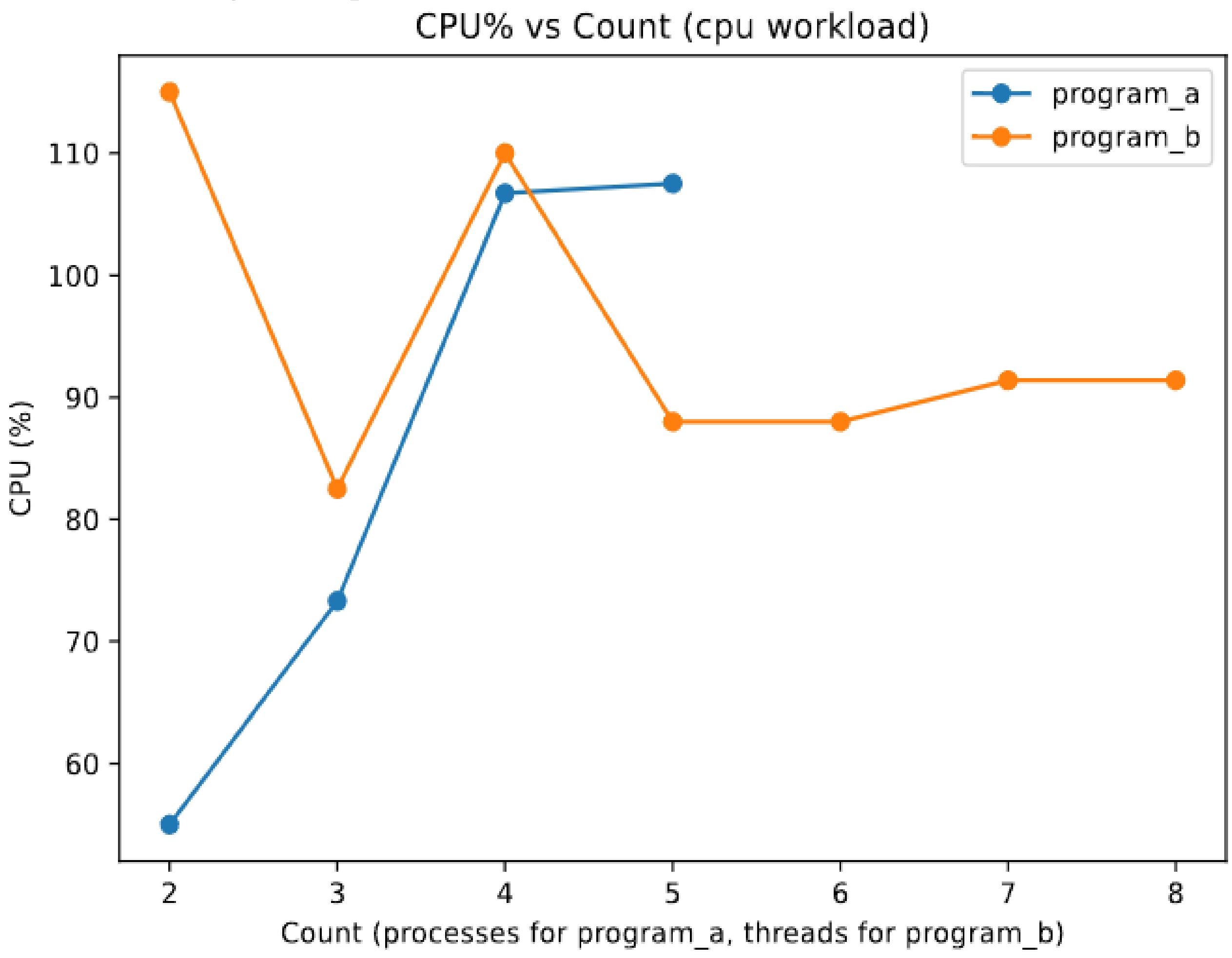
3.1 CPU Scaling:

Summary Table from CSV:

Here count is the number of child processes in case of program_a and number of threads in case of program_b

Program	Workload	Count	CPU%	Mem(KB)	IO(%util)
Program_a	cpu	2	55.0	1280	0.48
Program_a	cpu	3	73.3	1963	0.80
Program_a	cpu	4	106.7	3329	1.60
Program_a	cpu	5	107.5	3712	0.44
Program_b	cpu	2	115.0	2048	0.51
Program_b	cpu	3	82.5	1536	0.52
Program_b	cpu	4	110	2048	0.61
Program_b	cpu	5	88.0	1638	0.59
Program_b	cpu	6	88.0	1638	0.57
Program_b	cpu	7	91.4	1755	0.80
Program_b	cpu	8	91.4	1755	0.52

3.1.1 CPU Scaling under cpu workload:



Plot 1: CPU% vs Count under CPU workload

Program A (Processes)

From CPU workload table and CPU% vs count plot the following trends were observed:

CPU utilization increases consistently from 55% at 2 processes to 107% at 4–5 processes. This increase reflects better CPU saturation, where more child processes are fighting for the same core. Beyond 4 processes, CPU usage saturates close to 100–110% (showing that this core is fully utilized). Additional processes beyond this do not offer substantive returns, while they do add context-switching overhead.

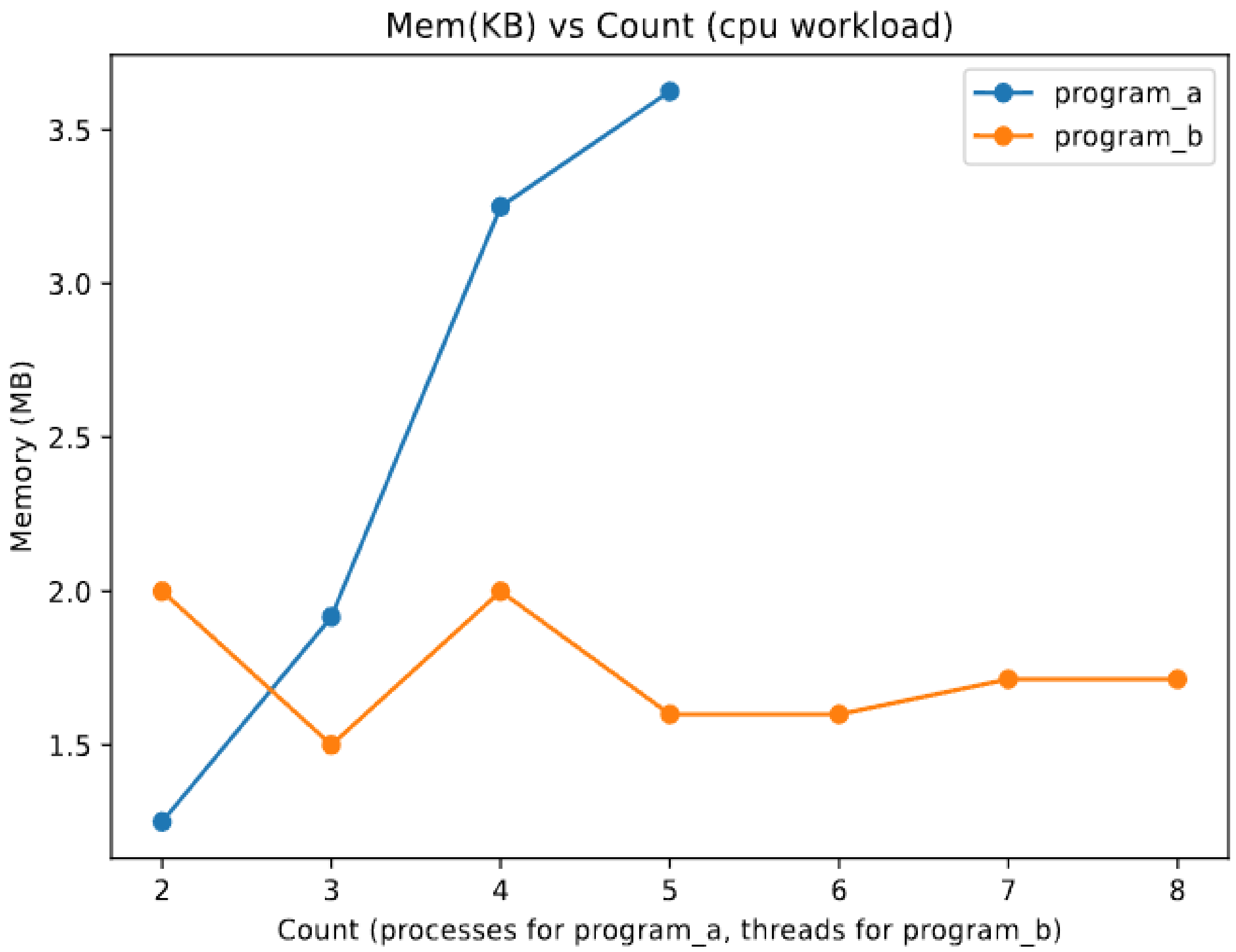
Program B (Threads)

CPU utilization begins high (115% with 2 threads), decreases when 3 threads are present, and varies between 88-92% for increased numbers of threads. Compared to a process, a thread shares the same address space, thus having fewer context switches as well as increased clash for shared execution resources. The absence of monotonic scaling also hints at the thread scheduling on a single core exhibiting timing and synchronization influences. The CPU does not become more utilized when the count of threads is elevated beyond a certain extent because the maximum limits apply to each and every one.

Conclusion:

This plot shows that for CPU-bound applications on a single core, increasing the number of processes or threads beyond a small threshold yields CPU saturation without performance improvement. It shows the importance of making parallelism consistent with the available hardware resources.

3.1.2 Memory Scaling under cpu workload:



Plot 2: Memory (KB) vs Count under CPU workload

Program A (Processes)

The memory usage increases monotonically with an increase in processes. Every new procedure introduces its distinct address space along with its procedure stacks, heaps, and kernel structures.As a result, memory usage grows almost linearly with the number of processes.

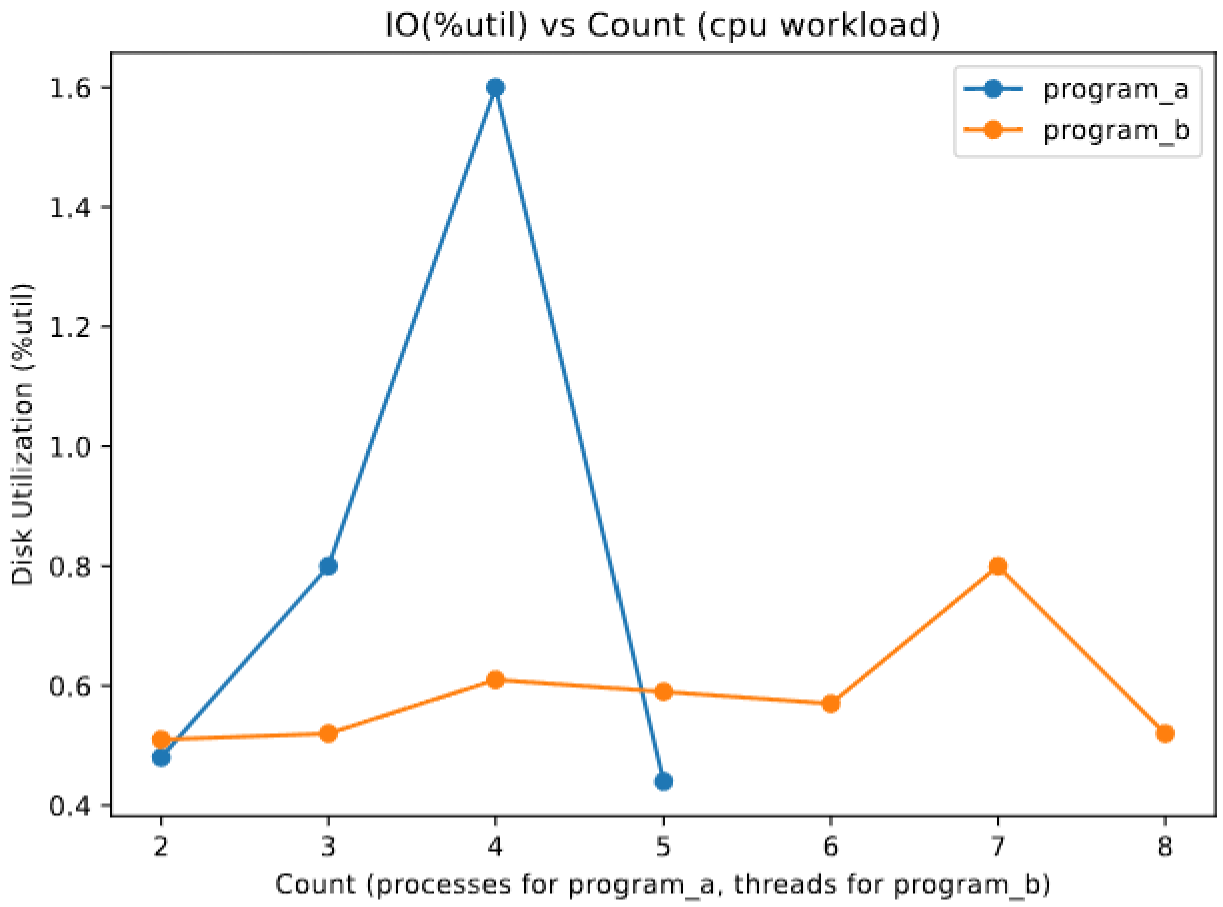
Program B (Threads)

Memory usage is fairly consistent across different thread counts, with just minor variation. Threads have the common address space, code segment, and global data. The minor variations are due to allocator behavior and granularity of measurement in top.

Conclusion:

For a CPU-bound application, an increased number of processes results in considerable memory growth, whereas an increased number of threads does not affect memory usage in a significant manner. This makes threads more appropriate for highly concurrent CPU-intensive programs, especially in situations where memory usage needs to be saved.

3.1.3 IO(% util) scaling under cpu workload:



Plot 3: IO (% util) vs Count under CPU workload

Program A (Processes)

I/O utilization rises slightly between 2 to 4 processes, which is a peak of 1.6%, before decreasing significantly at 5 processes. The small spike may be attributed to: inherent measurement noise of iostat example - process startup, termination overhead, etc. Since CPU-intensive processes do not normally wait on disk access, the variations do not measure real I/O blocking:

Program B (Threads)

I/O utilization is steady and low, running at a small range of percentages around 0.5 to 0.8%. Threads share their access to the file descriptor and share the context of the process. This minimizes additional disk access. Minor increases in the higher thread count numbers should be due to scheduler and system activity rather than workload-driven I/O.

Conclusion:

Under a CPU-intensive workload, disk I/O utilization remains negligible regardless of concurrency level. Any observed fluctuations are artifacts of system overhead and measurement granularity (iostat averages disk activity over that interval) rather than actual I/O demand.

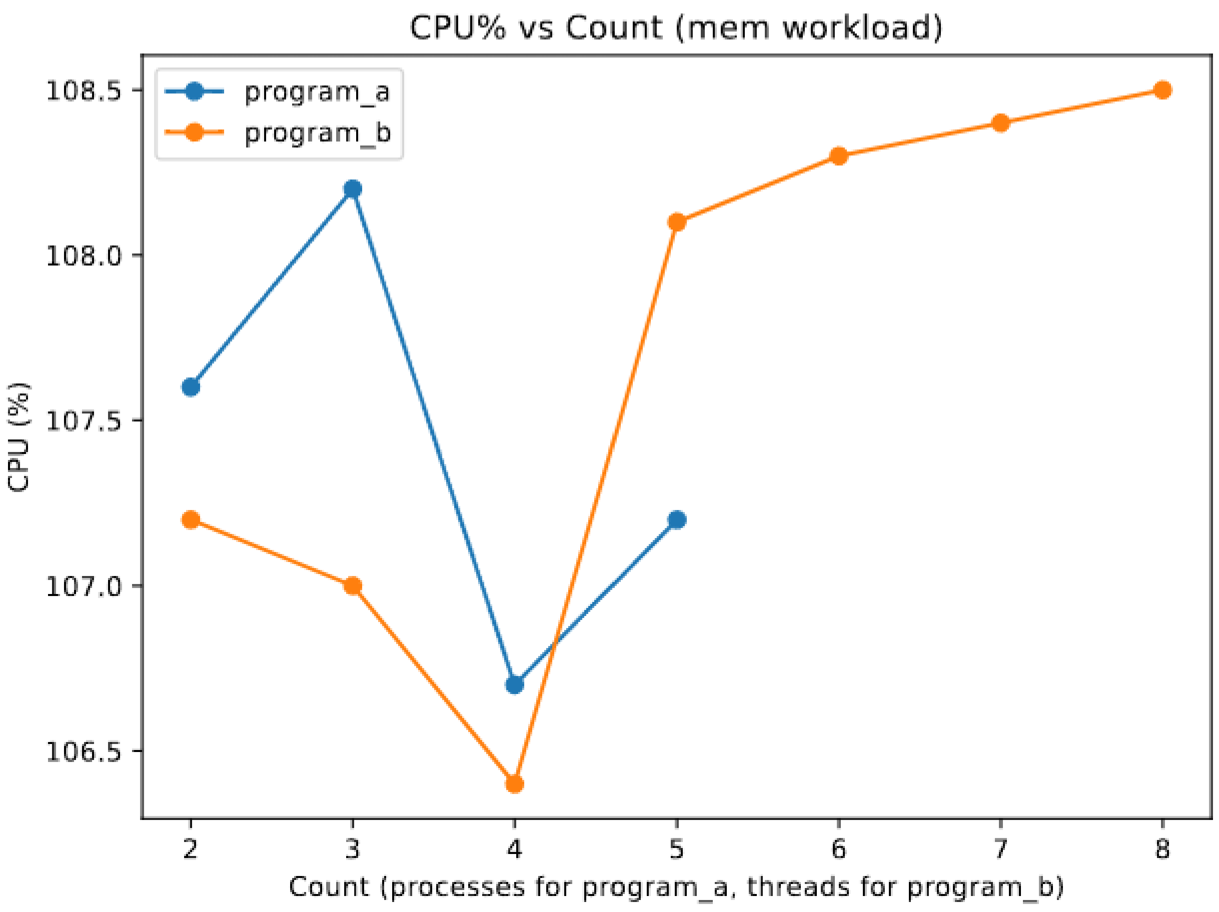
3.2 Memory Scaling:

Summary Table from CSV:

Here count is the number of child processes in case of program_a and number of threads in case of program_b

Program	Workload	Count	CPU%	Mem(KB)	IO(%util)
Program_a	Mem	2	107.6	104720	3.20
Program_a	Mem	3	108.2	156551	4.80
Program_a	Mem	4	106.7	207936	9.60
Program_a	Mem	5	107.2	259470	30.33
Program_b	Mem	2	107.2	103918	79.27
Program_b	Mem	3	107.0	154958	4.00
Program_b	Mem	4	106.4	206182	24.40
Program_b	Mem	5	108.1	257480	2.80
Program_b	Mem	6	108.3	308358	1.20
Program_b	Mem	7	108.4	359360	1.60
Program_b	Mem	8	108.5	410496	471.20

3.2.1 CPU Scaling under mem workload:



Plot 4: CPU % vs Count under CPU workload

Program A (Processes)

The CPU utilization fluctuates in a small range 106-108%. This fluctuates in line with the number of running processes, even though the system consumes considerable memory, with the CPU always working on accessing memory buffers, managing cache misses, and handling address translation operations. The small dip seen at 4 processes is likely a scheduling phenomenon, not an actual reduction in workload intensity. The CPU utilization is near saturation across all samples and thus likely indicates that we are using a fully utilized pinned core.

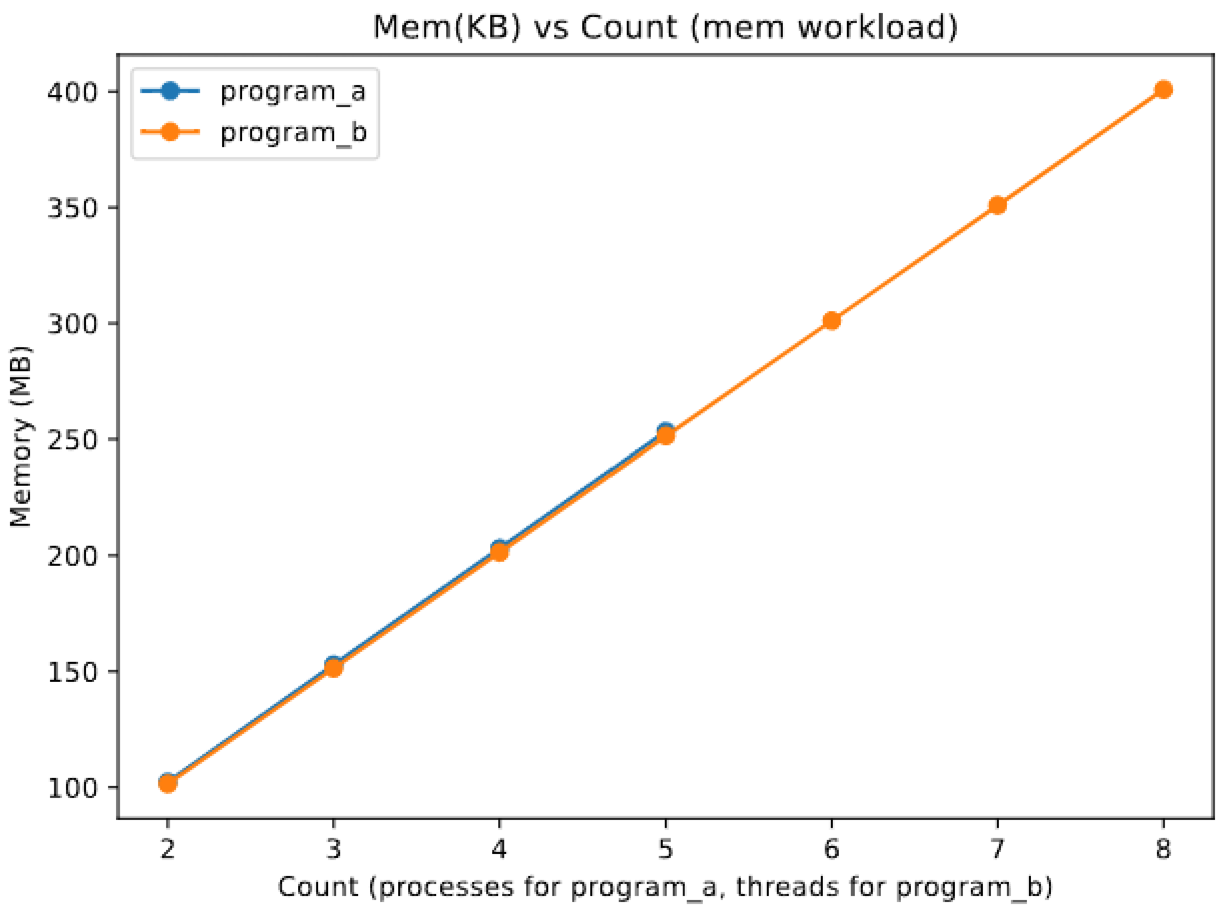
Program B (Threads)

A similar pattern could be noticed for threads. CPU utilization stays close to 107–108.5% and increases slightly as the thread count rises. Threads share the same address space therefore, there is less memory management overhead compared to processes. This gradual increase in CPU usage reflects added pressure on memory access and cache handling due to an increase in the number of threads operating at the same time.

Conclusion:

In the memory-intensive case, it can be seen that the CPU usage remains high and near saturation in both processes and threads. The rate at which CPU usage increases with a higher count of execution entities remains almost same since this is a single core and not because of CPU capability itself.

3.2.2 Mem Scaling under mem workload:



Plot 5: Memory (KB) vs Count under Mem workload

Program A (Processes)

It can be seen that memory consumption rises linearly as the number of running processes increases. This is due to the allocation of a large memory buffer per process under the mem() task, and each process having its own virtual address space. Hence, the sum of memory consumption rises linearly.

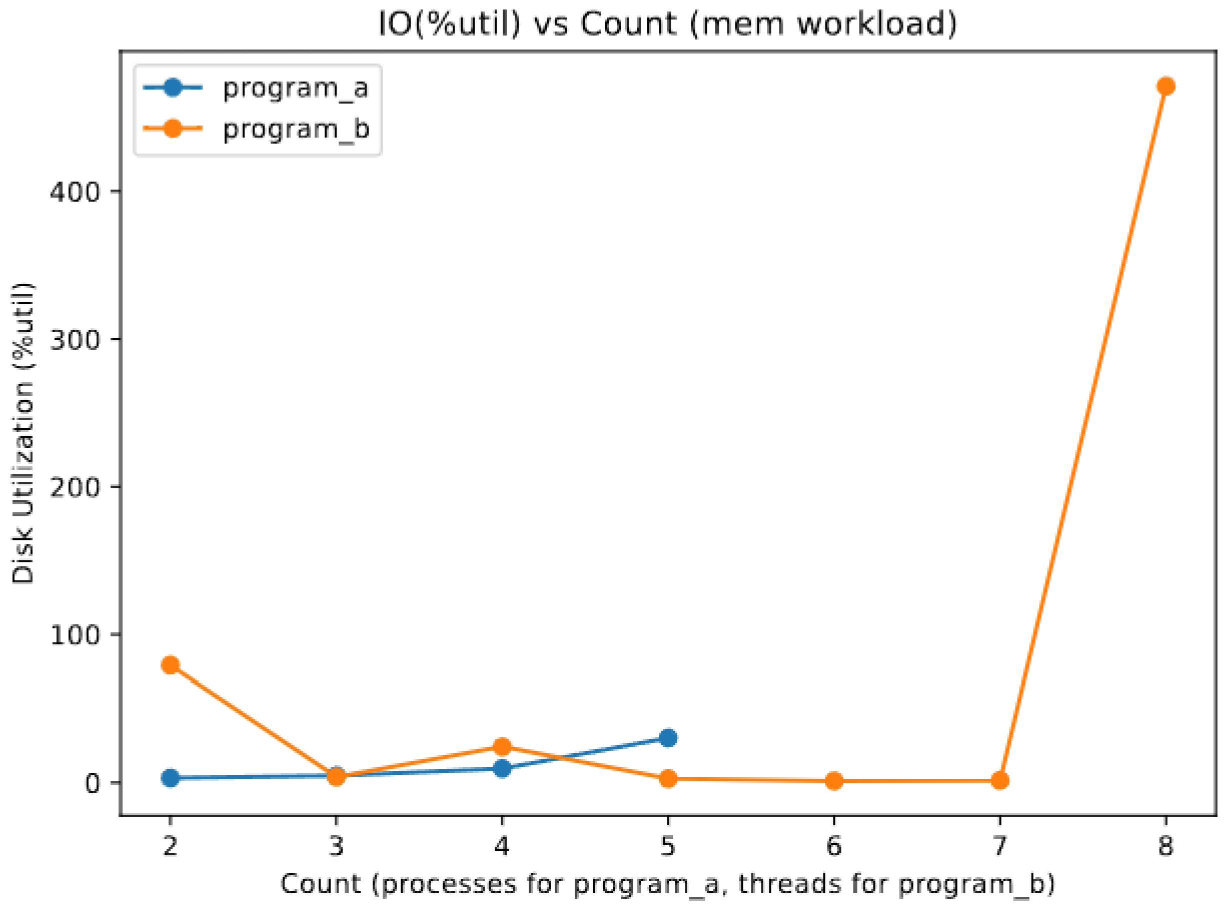
Program B (Threads)

Similarly, there is an increase at a constant rate for memory usage with an increase in the number of threads. Although threads have the same address space, during execution of memory workload, every thread uses memory buffers of its own, i.e., memory usage increases with an increase in the number of threads.

Conclusion:

The memory requirement for threads is very similar in terms of processes, for the same number of processes, showing that we are largely governed by memory allocation for each worker in a workload.

3.2.3 IO Scaling under mem workload:



Plot 6: IO (%util) vs Count under Mem workload

Program A (Processes)

The disk utilized for Program A maintains a relatively low disk utilization at smaller counts but increases as the number of processes increases, reaching 30% at 5 processes. The memory workload does not directly utilize disk operations but consistently makes allocations to large memory buffers in RAM. This trend of increasing usage of the disk might be caused by several reasons, primarily related to memory usage. This includes memory allocation to pages, reclaiming memory from pages, and kernel background activities, rather than any workload-driven I/O to files. This is especially true when the memory usage is high, as there would be more processes.

Program B (Threads)

As can be determined in program B, it demonstrates irregular disk usage. It, however, demonstrates an enormous spike in I/O disk usage at an 8-thread count. This can not be interpreted that the memory workload results in disk I/O, but rather is likely caused by OS-level memory pressure, that is, additional concurrent allocation of memory by multiple threads can cause delayed write-back, flushing, etc (threads have the same address space).

Conclusion:

Under the memory-intensive workload, disk I/O is not a primary performance factor. The observed disk utilization, including the large spike at higher thread counts, is driven by operating system memory management behavior rather than by explicit I/O operations performed by the workload.

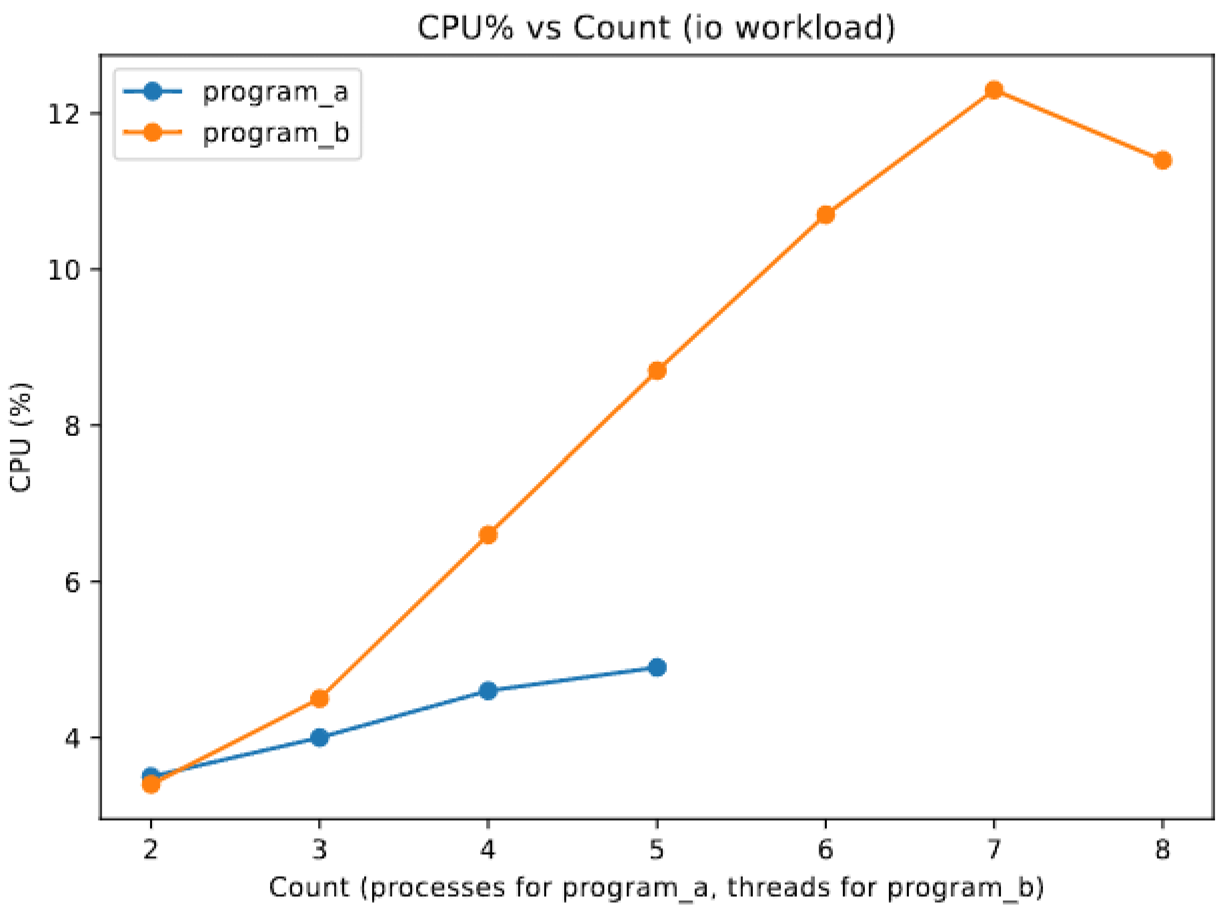
3.3 IO Scaling:

Summary Table from CSV:

Here count is the number of child processes in case of program_a and number of threads in case of program_b

Program	Workload	Count	CPU%	Mem(KB)	IO(%util)
Program_a	IO	2	3.5	4081	10.00
Program_a	IO	3	4.0	5230	7.20
Program_a	IO	4	4.6	6382	10.80
Program_a	IO	5	4.9	7535	441.60
Program_b	IO	2	3.4	2299	5.20
Program_b	IO	3	4.5	2555	47.27
Program_b	IO	4	6.6	2809	4.40
Program_b	IO	5	8.7	3063	9.07
Program_b	IO	6	10.7	3320	3.30
Program_b	IO	7	12.3	3580	3.60
Program_b	IO	8	11.4	3966	3.20

3.3.1 CPU Scaling under IO workload:



Plot 7: CPU % vs Count under IO workload

Program A (Processes)

The percentage utilization for CPU by Program A remains very low. From 3.5 for 2 processes, it gradually increases to 4.9 for 5 processes. Due to the I/O-intensive nature of programs, most of the processes will spend most of their computation time waiting for I/O operations to finish. A gradual increase in CPU utilization also takes place due to increased overhead for scheduling processes. However, no major bottlenecks occur.

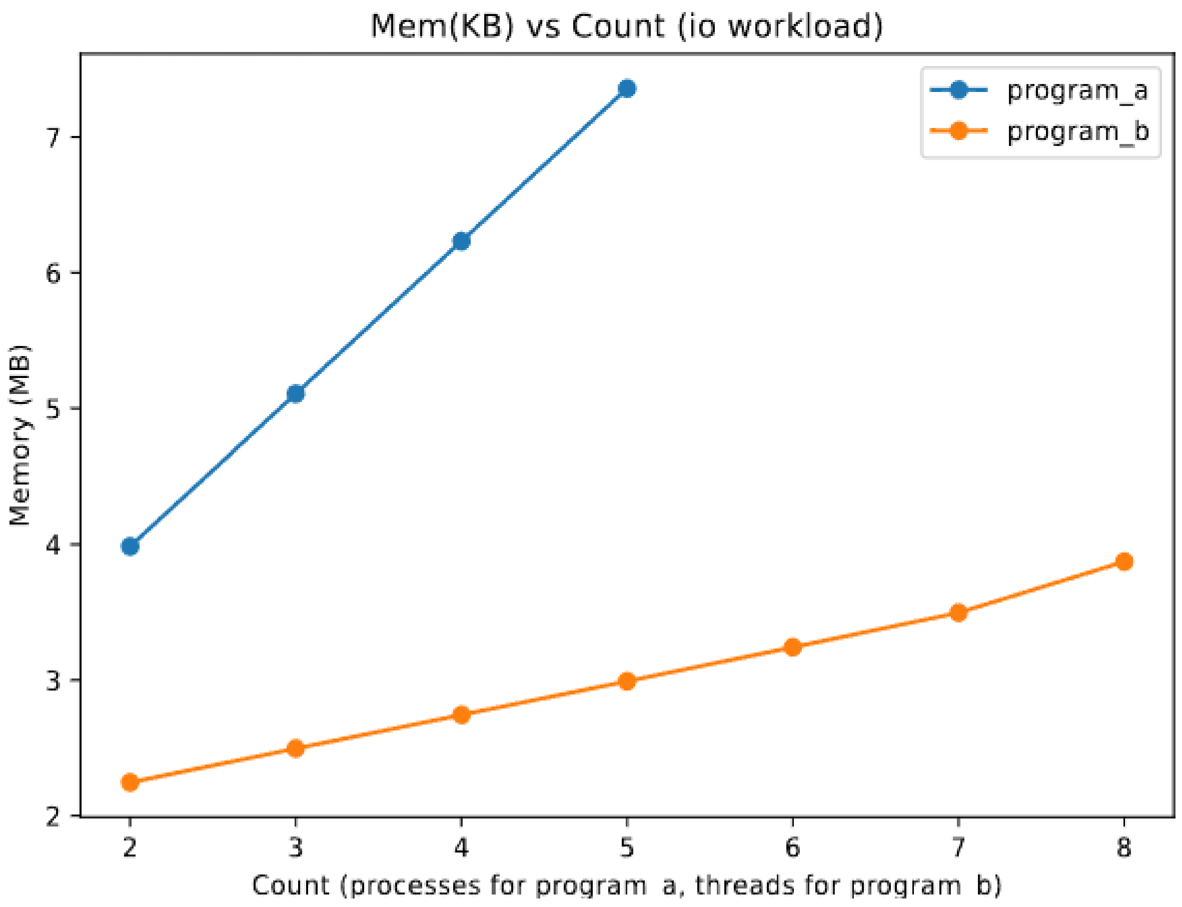
Program B (Threads)

For Program B, CPU utilization increases more noticeably as the number of threads grows. The increase was from 3.4% at 2 threads to approximately 12.3% at 7 threads before slightly going down for 8 threads. Threads submit I/O requests in parallel and share all of the process context thus CPU involvement increases for managing the many I/O requests as well as performing synchronizations. The more threads there are, the more the CPU is consumed in these activities. The slight drop at higher thread counts indicates the saturation of the I/O subsystem-adding more threads beyond that doesn't increase the effective parallelism, but instead introduces additional wait cycles.

Conclusion:

CPU utilization under the I/O-intensive workload remains low for both processes and threads, as expected since the workload is I/O-bound. At larger counts, threads incur higher CPU usage than processes due to increased coordination and synchronization overhead but in all cases, the CPU is not the limiting resource (Increasing CPU availability does not reduce execution time).

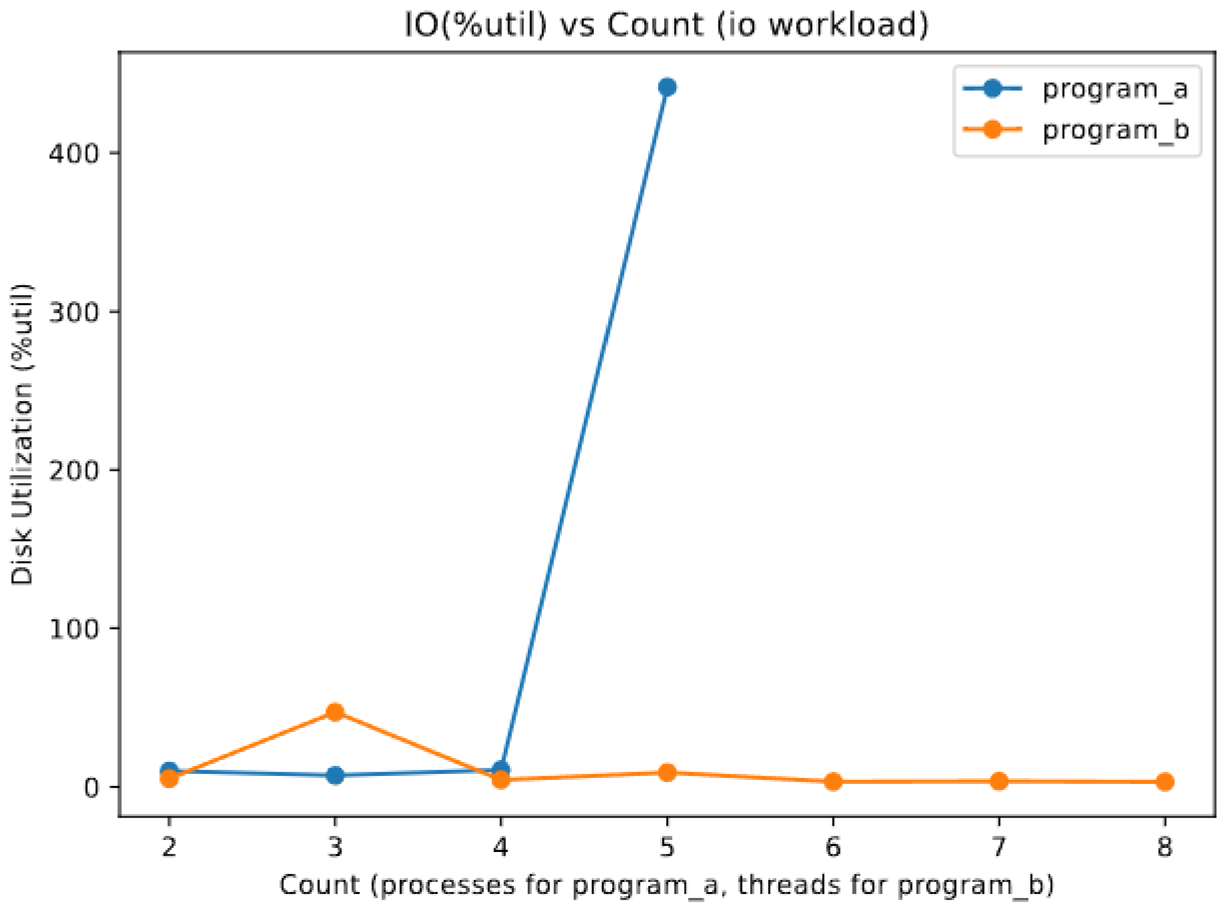
3.3.2 Mem Scaling under io workload:



Plot 8: Memory (KB) vs Count under io workload

Under an I/O-intensive workload, memory consumption grows with increasing concurrency for both processes and threads. However, threads use memory more efficiently than processes, making them better suited for highly concurrent I/O-bound applications where memory overhead needs to be minimized.

3.3.3 IO Scaling under io workload:



Plot 9: IO (%util) vs Count under io workload

Program A (Processes)

Disk Utilization stays steady for lower numbers of processes. However, it produces a sharp spike at 5 processes with a high value. This implies that the disk would be more heavily utilized whenever there are concurrent processes containing their respective I/O requests. The sharp spike implies that the disk would act as a limiting resource with the CPU remaining idle.

Program B (Threads)

Disk utilization for threads still appears to be low, although this figure increases substantially with 3 threads, after which it remains low for larger thread counts. Because threads share the address space it helps to control too many disk queues, hence smooth disk utilization compared to the use of processes.

Conclusion:

With I/O-intensive workload, the usage of the disks continues to grow considerably depending on the number of running processes, and this usage can grow to the extent that the disks can get saturated. Threads show very stable I/O access due to resource-sharing; however, in this case, the resource being limited is the disk, not the CPU.

AI Usage Declaration:

AI tools were used for assistance in understanding concepts, debugging issues, refining explanations, and supporting experimental analysis. All code was written by me, thoroughly reviewed, and fully understood. Experimental execution, debugging, and performance analysis were conducted by me with AI used as a supportive aid.

GitHub Repository Link: https://github.com/theharshaverma/GRS_Assignments.git