Graduate Systems (CSE638)

Assignment – PA-01 (Processes and threads)

Name: Yash verma (MT25092)

Question A/Part A: Process and Thread Creation

1. Running Program A:
   . /ProgramA cpu 2

Output:

```
techg@YASH:/mnt/c/Users/techg/OneDrive - Manipal University Jaipur/Des
ktop/CLone/GRS_PA01$ ./programA cpu 2
Program A (Processes)
Worker: cpu | Processes: 2 | Loops: 2000
```

2. Running Program B:
   . /ProgramB cpu 2

Output:

```
techg@YASH:/mnt/c/Users/techg/OneDrive - Manipal University Jaipur/Des
ktop/CLone/GRS_PA01$ ./programB cpu 2
Program B (Threads)
Worker: cpu | Threads: 2 | Loops: 2000
```

Analysis of QA :

1. **Program A** uses **fork()** to create child processes.
2. Each processes has a separate address space.
3. **Program B** uses pthread_Create() to create threads.
4. Threads share memory, leading to lower overhead.

Question B/Part B: Worker Functions:

1. Running workers:

   ./program cpu 6

Output:

```
techg@YASH:/mnt/c/Users/techg/OneDrive - Manipal University Jaipur/Des
ktop/CLone/GRS_PA01$ ./programB cpu 6
Program B (Threads)
Worker: cpu | Threads: 6 | Loops: 2000
```

./programB mem 6

Output:

```
techg@YASH:/mnt/c/Users/techg/OneDrive - Manipal University Jaipur/Des
ktop/CLone/GRS_PA01$ ./programB mem 6
Program B (Threads)
Worker: mem | Threads: 6 | Loops: 2000
```

./program io 6

Output:

```
techg@YASH:/mnt/c/Users/techg/OneDrive - Manipal University Jaipur/Des
ktop/CLone/GRS_PA01$ ./programB io 6
Program B (Threads)
Worker: io | Threads: 6 | Loops: 2000
```

Analysis:

1. **CPU worker:**
   (a) Heavy floating point operations.
   (b) It keeps the CPU busy.
2. **Memory worker:**
   (a) Repeated access to large memory.
   (b) Causes cache misses and memory traffic.
3. **IO worker:**
   (a) Writes a disk using fsync.
   (b) CPU often idle, waiting for disk.

Question C/Part C : Measurement using top, iostat, time

Running shell script for part C:

```
techg@YASH:/mnt/c/Users/techg/OneDrive - Manipal University Jaipur/Des
ktop/CLone/GRS_PA01$ chmod +x MT25092_Part_C_shell.sh
techg@YASH:/mnt/c/Users/techg/OneDrive - Manipal University Jaipur/Des
ktop/CLone/GRS_PA01$ ./MT25092_Part_C_shell.sh
Running A + cpu
Running A + mem
Running A + io
Running B + cpu
Running B + mem
Running B + io
Saved to MT25092_Part_C_CSV.csv
```

Output:

```
Program,Worker,CPU_Percent,Disk_TPS,Time_Seconds
A,cpu,47,1.68,0.01
A,mem,99,0.38,40.82
A,io,5,0.58,5.01
B,cpu,50,0.38,0.01
B,mem,99,0.38,33.29
B,io,5,0.37,4.91
```

Analysis for part C:

Comparison:

Variant	Observation
A+ CPU	**Moderate CPU %, lessor execution**
A + MEM	**High CPU, longer execution**
A+ IO	**Lower CPU utilization**
B + CPU	**Moderate CPU percentage, faster execution**
B + MEM	**Higher CPU utilization, Higher time**
B + IO	**Similar to Process behavior**

Observation:

1. Threads perform better due to shared memory.
2. Processes incur context switch overhead.

Question D/Part D: Scaling with Processes and Threads

1. Running the D part:
   Chmod + x MT25092_Part_D_shell.sh

On terminal/Ubuntu:

```
techg@YASH:~$ chmod +x MT25092_Part_D_shell.sh
```

2. **Output in CSV file:**

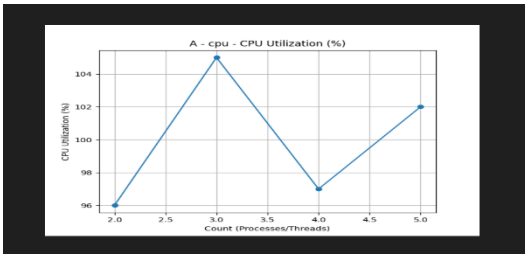
```csv
Program,Worker,Count,CPU_Percent,Disk_TPS,Time_Seconds
A,cpu,2,96,0.78,9.50
A,cpu,3,105,0.48,11.30
A,cpu,4,97,0.47,14.91
A,cpu,5,102,0.47,14.93
A,mem,2,100,0.45,36.37
A,mem,3,100,0.44,52.06
A,mem,4,99,0.42,66.65
A,mem,5,99,0.40,83.01
A,io,2,4,0.50,6.02
A,io,3,4,0.40,6.24
A,io,4,4,0.40,9.96
A,io,5,5,0.39,9.92
B,cpu,2,97,0.39,7.03
B,cpu,3,97,0.39,11.55
B,cpu,4,103,0.58,13.98
B,cpu,5,97,0.38,18.35
B,cpu,6,100,0.38,20.09
B,cpu,7,98,0.37,23.52
B,cpu,8,101,0.37,26.59
B,mem,2,100,0.36,37.26
B,mem,3,98,0.85,53.26
B,mem,4,101,0.34,65.84
B,mem,5,99,0.83,83.09
B,mem,6,100,0.31,103.59
B,mem,7,100,0.30,121.26
B,mem,8,99,0.28,135.87
B,io,2,5,0.28,6.10
B,io,3,4,0.28,6.65
B,io,4,5,0.28,8.68
B,io,5,4,0.28,11.86
B,io,6,5,0.28,12.33
B,io,7,4,0.28,16.27
B,io,8,4,0.57,15.31

```

Analysis:

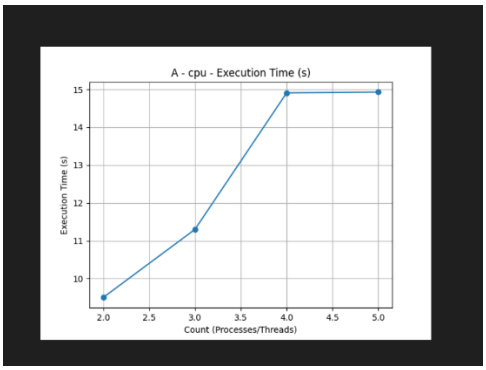
1. **CPU saturates as count increases.**
2. **After certain threads, no performance gain.**
3. **More context switching overhead.**
4. **Disk becomes bottleneck for IO workload.**

3. **Plots of different variants with different counts.**

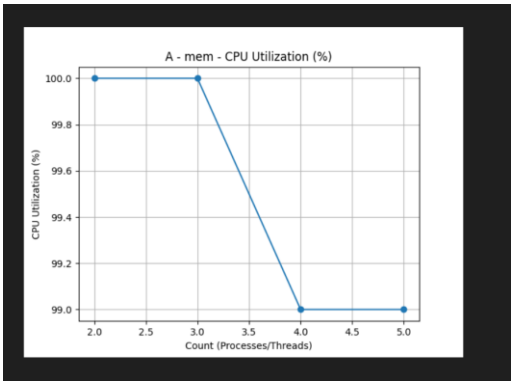
1. **A + CPU count -2**



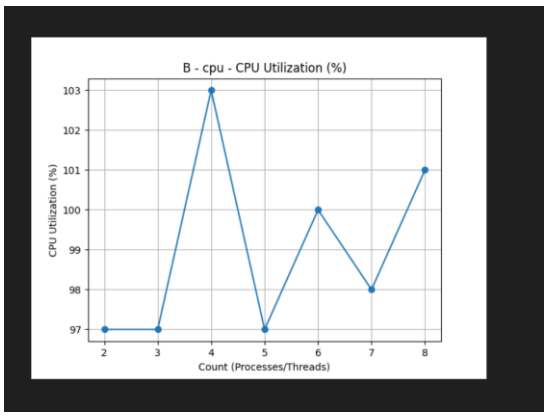
2. A + Execution time:



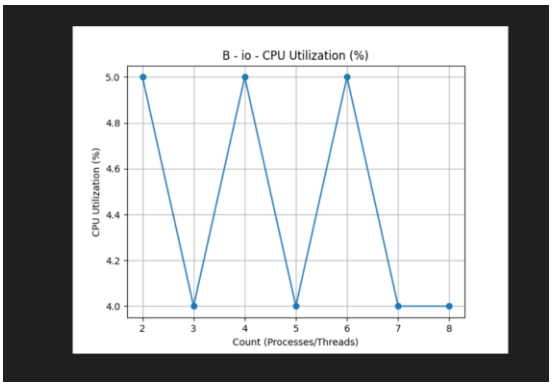
3. A + memo (CPU Utilization)



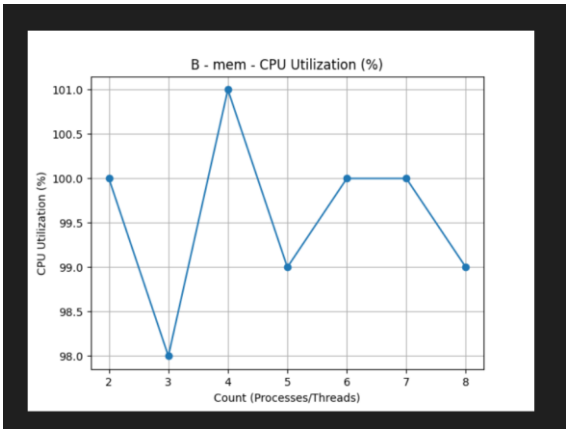
4. B + CPU



5. **B + IO**



6. **B + mem**



Note: I have uploaded all the plots in plot folder under the GRS_PA01

AI Declaration:

1. **Code structure:** I have modified the structure using LLM like in MT25092_Part_A_Program_A and MT25092_Part_A_Program_B how the loops will work and how the result will be printed I have taken the help from LLM.

2. **Bash files:** In shell files like MT25092_Part_C_shell.sh and MT25092_Part_D_shell.sh I generate the logic from LLM like how will the CPU percentage, Disk_TPS and Time seconds it will generate.

3. **Python file:** Taken the logic from LLM for file MT25092_Part_D_Plots.py how the plots will be generated from MT25092_Part_D_CSV.csv.

Make file, MT2092_Part_A_Program_A, MT25092_Part_A_Program_B, report, readme is generated by me.

Github repo: **https://github.com/Vyash2002/GRS_PA01.git**