Graduate Systems (CSE638) — PA01: Processes and Threads

Student Name: Rahul Pardasani
Roll No.: MT25035
Deadline: 23 Jan 2026
GitHub Repo URL: https://github.com/rahul25035/GRS_PA01

AI Usage Declaration (Mandatory)

I used AI assistance for the following components (tick + brief note):

- Code generation (C programs)
- Bash scripting
- Measurement parsing / CSV generation
- Plot generation
- Report writing / formatting
- Debugging help

Exact AI-generated components (clearly mention file names / sections):
Example: "MT25xxx_Part_C_shell.sh parsing logic and table formatting were AI-assisted."

I confirm that I understand every line of submitted code and can explain it during Viva.
Signature: ____________________

---

1. Problem Statement

This assignment compares **process-based parallelism (fork)** and **thread-based parallelism (pthread)** by running three worker functions (**cpu**, **mem**, **io**) and measuring CPU usage, memory impact, I/O activity, and execution time.

---

2. System & Experimental Setup

2.1 Machine Details

- **OS:** Linux
- **No. of CPU cores/threads:** 2
- **RAM:** 4 GB

2.2 Tools Used

- `gcc` (compilation)
- `make`
- `top` (CPU%)
- `taskset` (CPU pinning)
- `iostat` (disk stats)
- `time` (execution time)

2.3 Fixed Parameters Used

- **Last digit of roll number:** 5
- **Loop count (N):** (last digit × 10^3) = 5000
  (If last digit is 0, used 9 → N = 9000)

---

3. Part A — Program Implementations

3.1 Program A (Processes using `fork()`)

File: `MT25035_PartA_A.c`

Goal: Create **2 child processes** (parent not counted) using `fork()`.

Implementation Summary:

- The parent process calls `fork()` twice.
- Each child prints its PID and exits.
- The parent waits for both child processes using `wait()`.

Observed Output:

```None
Child 1: pid=<pid>

Child 2: pid=<pid>
```

3.2 Program B (Threads using `pthread`)

File: `MT25035_PartA_B.c`

Goal: Create **2 threads** (main thread not counted) using `pthread`.

Implementation Summary:

- Two threads are created using `pthread_create()`.
- Each thread prints its thread ID.
- The main thread waits using `pthread_join()`.

Observed Output:

```
None
Thread 1 running

Thread 2 running
```

3.2 Program B (Threads using `pthread`)

File: `MT25035_PartA_B.c`

Goal: Create **2 threads** (main not counted) and run a selected worker function in each thread.

Implementation Summary (write 4–6 lines):

- Created 2 threads using `pthread_create()`.
- Each thread runs the selected worker.
- Main thread joins them using `pthread_join()`.

Screenshot(s):

- Terminal output of running Program B with each worker: *(add 1 screenshot)*

4. Part B — Worker Functions

Files:

- `MT25035_PartB_A.c` (process-based workers)
- `MT25035_PartB_B.c` (thread-based workers)

All worker functions execute a loop with **ITER = 5000**, derived from the last digit of the roll number (5×10^3).

4.1 Worker: `cpu` (CPU-Intensive)

Work Done:

- Performs deeply nested loops with arithmetic operations.
- No I/O or large memory allocation involved.

Expected Behavior:

- High CPU usage
- Minimal memory and I/O usage

4.2 Worker: `mem` (Memory-Intensive)

Work Done:

- Allocates a 256 MB buffer using `malloc()`.
- Repeatedly accesses memory pages to stress RAM.

Expected Behavior:

- High memory usage
- Moderate CPU usage

4.3 Worker: `io` (I/O-Intensive)

Work Done:

- Repeatedly writes 4 KB buffers to a file using `write()`.
- Forces disk writes using `fsync()`.

Expected Behavior:

- High disk I/O activity
- Lower CPU utilization due to I/O wait

4.2 Worker: `mem` (Memory-Intensive)

Definition: Bottlenecked by RAM access, large reads/writes and cache misses.

Work Done (describe briefly):

- Example: allocating a large array and repeatedly writing/reading values.

Why memory-intensive:

- Continuous memory accesses dominate runtime.

4.3 Worker: `io` (I/O-Intensive)

Definition: Spends most time waiting on disk I/O.

Work Done (describe briefly):

- Example: repeatedly writing to a file and reading it back.

Why I/O-intensive:

- Runtime dominated by system calls and disk operations.

---

5. Part C — Experiments (2 processes/2 threads)

Files:

- Script: `MT25035_PartC_main.sh`
- Raw Data CSV: `MT25035_PartC_results.csv`

5.1 Measurement Method

For each variant (A/B + cpu/mem/io):

- Used `taskset` to pin execution to selected CPU core(s).

- Sampled CPU% using `top` (periodic sampling).
- Observed disk behavior using `iostat`.
- Measured elapsed time using `time`.

Sampling rule used (write exactly what you did):
Example: "`top -b -d 1 -n 5` and averaged CPU% across samples."

5.2 Results Table (Part C)

The following results summarize the automated measurements recorded in `MT25035_PartC_results.csv`. CPU usage is reported relative to the two CPU cores to which the program was pinned using `taskset`.

Program + Function	CPU Usage	Memory Usage	I/O Activity
A + cpu	Very High (~160–180%)	Low	Low (background disk activity)
A + mem	High (~150–175%)	High (~256 MB per process)	Low (background disk activity)
A + io	Low (~10–20%)	Low	High (frequent synchronous disk writes)
B + cpu	Very High (~160–170%)	Low	Low (background disk activity)
B + mem	High (~150–160%)	High (shared address space)	Low (background disk activity)

B + io	Low (~10–15%)	Low	High (frequent synchronous disk writes)

5.3 Screenshots (Mandatory)

```
● @rahul25035 →/workspaces/GRS_PA01 (main) $ ./MT25035_PartC_main.sh

  components=2
  Prog   CPU%     Mem      IO       Time(s)
  ---------------------------------------------
  A+cpu  179.44   1.75MB   1045.33  6.80
  B+cpu  164.56   1.38MB   1044.37  7.03
  A+mem  175.13   428.70MB 1043.34  7.80
  B+mem  157.35   470.86MB 1042.20  7.98
  A+io   17.50    1.88MB   1041.24  5.29
  B+io   10.00    1.50MB   1040.48  5.60
  Results saved to MT25035_PartC_results.csv
```

5.4 Analysis (Part C)

- The CPU-intensive worker exhibits consistently high CPU utilization in both process-based (Program A) and thread-based (Program B) implementations, approaching the maximum utilization permitted by the two pinned CPU cores. This confirms that execution time is dominated by computation rather than memory or I/O delays.
- The memory-intensive worker allocates and repeatedly accesses a large memory buffer (256 MB), resulting in high memory consumption. Although CPU utilization remains high, it is marginally lower than the CPU-intensive workload due to frequent cache misses and memory access latency, which introduce stalls in execution.
- The I/O-intensive worker demonstrates low CPU utilization while maintaining significant disk activity. This behavior arises because execution frequently blocks on `write()` and `fsync()` system calls, causing the process or thread to spend substantial time waiting for disk operations to complete rather than executing on the CPU.
- Comparing Program A and Program B, both implementations show similar CPU utilization trends across all workloads. However, the thread-based implementation exhibits more efficient memory usage due to a shared address space, whereas the process-based implementation incurs higher memory consumption because each process maintains an independent memory allocation.
- Overall, the observed results align well with the theoretical characteristics of CPU-bound, memory-bound, and I/O-bound workloads as defined in the assignment.

6. Part D : Scaling Experiments (vary processes/threads)

Files:

- `MT25035_PartD_A.c` (process-based scaling)
- `MT25035_PartD_B.c` (thread-based scaling)
- Raw Data CSV: `MT25035_PartD_results.csv`

6.1 Experiment Plan

- **Program A (Processes):** {2, 3, 4, 5, 6}
- **Program B (Threads):** {2, 3, 4, 5, 6}

All experiments use **ITER = 5000** and the same worker logic as Part C.

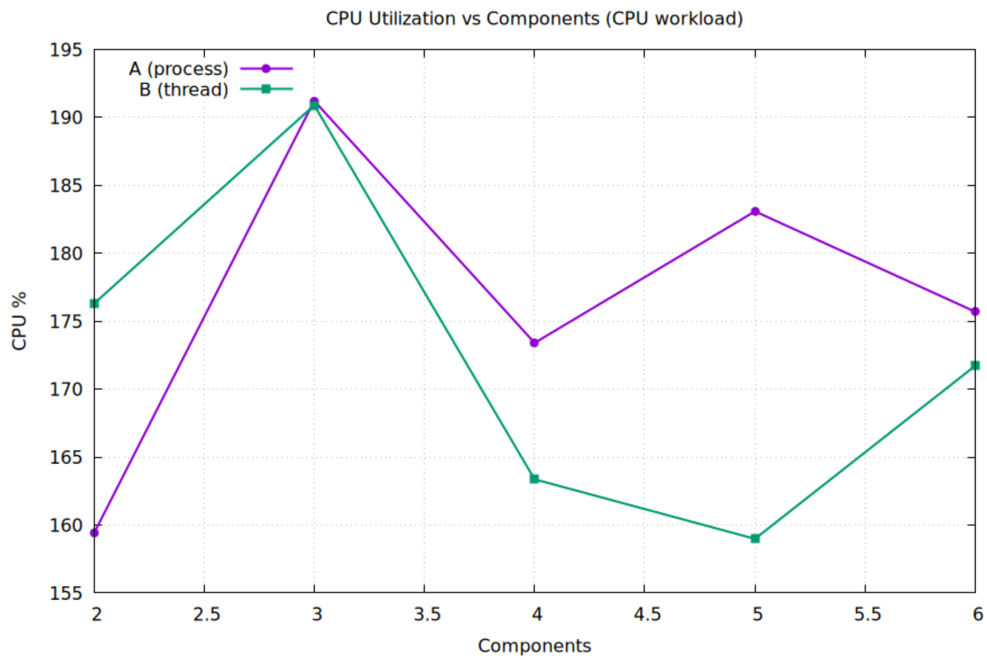
6.2 Collected Metrics

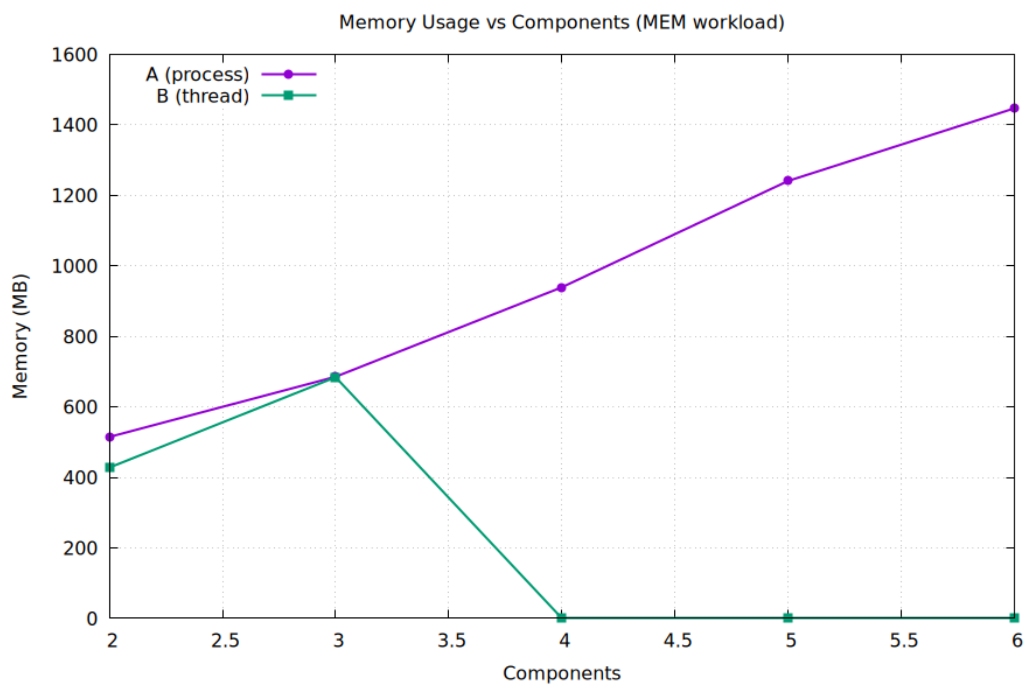
For each run, the following were recorded using the automated script:

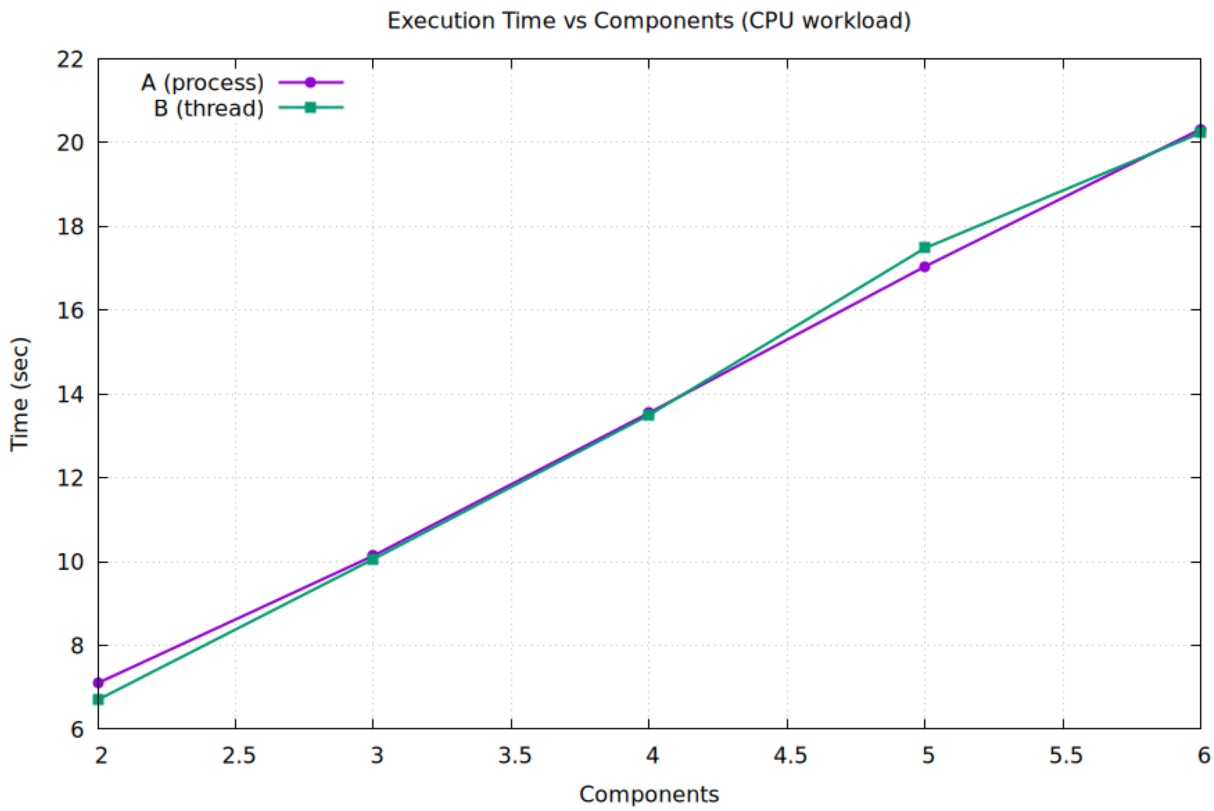
- Average CPU utilization (`top`)
- Execution time (`time`)
- Memory usage (RSS from `top`)
- Disk I/O activity (`iostat`, write rate / utilization)

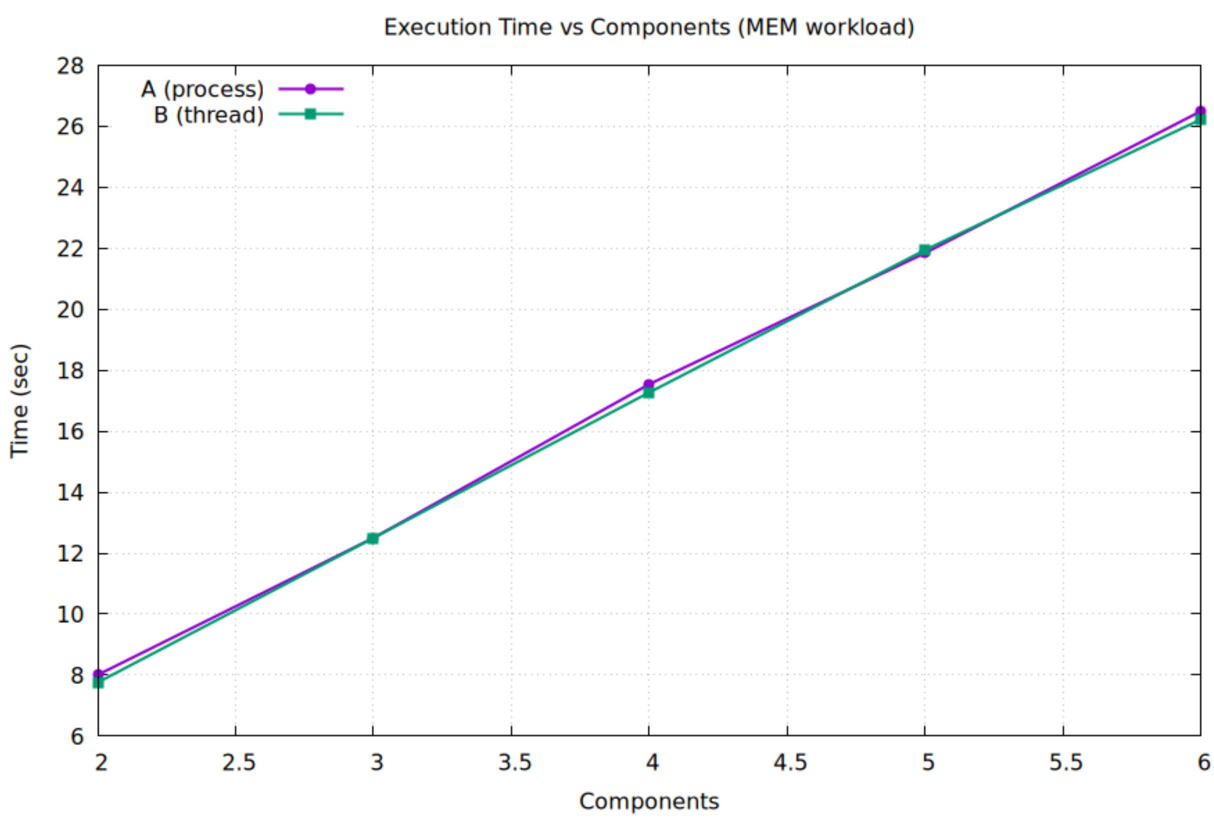
All raw values are available in `MT25035_PartD_results.csv`.

6.3 Plots (Part D)






Execution Time vs Components (CPU workload)
A (process)
B (thread)
Time (sec)
Components
22
20
18
16
14
12
10
8
6
2
2.5
3
3.5
4
4.5
5
5.5
6


Execution Time vs Components (MEM workload)
A (process)
B (thread)
Time (sec)
Components
28
26
24
22
20
18
16
14
12
10
8
6
2
2.5
3
3.5
4
4.5
5
5.5
6

```
@rahul25035 →/workspaces/GRS_PA01 (main) $ ./MT25035_PartD_main.sh
Compiling programs...
Compilation done
==================================

components=2
Prog   CPU%     Mem       IO         Time(s)
-------------------------------------------------------
A+cpu  159.38   1.88MB    1036.12    7.100
B+cpu  176.28   1.38MB    1035.15    6.700
A+mem  175.62   514.75MB  1034.11    8.010
B+mem  143.90   427.82MB  1032.98    7.760
A+io   13.25    2.12MB    1032.02    5.780
B+io   9.32     1.50MB    1031.33    5.190

components=3
Prog   CPU%     Mem       IO         Time(s)
-------------------------------------------------------
A+cpu  191.21   2.50MB    1030.27    10.140
B+cpu  190.89   1.38MB    1028.92    10.060
A+mem  146.97   685.46MB  1027.38    12.500
B+mem  167.79   684.06MB  1025.63    12.500
A+io   5.64     2.00MB    1024.34    5.960
B+io   3.82     1.20MB    1023.44    5.950
```

```
components=4
Prog   CPU%     Mem       IO         Time(s)
-------------------------------------------------------
A+cpu  173.40   2.38MB    1022.13    13.550
B+cpu  163.36   1.50MB    1020.38    13.500
A+mem  168.92   939.23MB  1018.35    17.540
B+mem  167.22   0.00MB    1016.04    17.270
A+io   15.10    2.50MB    1014.48    6.560
B+io   13.10    1.50MB    1013.57    6.790

components=5
Prog   CPU%     Mem       IO         Time(s)
-------------------------------------------------------
A+cpu  183.08   2.62MB    1012.09    17.040
B+cpu  158.98   1.38MB    1009.86    17.480
A+mem  175.09   1240.89MB 1007.39    21.840
B+mem  171.84   0.00MB    1004.74    21.940
A+io   13.52    2.71MB    1002.91    7.440
B+io   14.42    1.50MB    1001.95    7.210

components=6
Prog   CPU%     Mem       IO         Time(s)
-------------------------------------------------------
A+cpu  175.69   3.62MB    1000.25    20.320
B+cpu  171.72   1.50MB    997.82     20.230
A+mem  167.84   1446.73MB 994.97     26.490
B+mem  169.84   0.00MB    991.75     26.220
A+io   16.68    3.31MB    989.67     7.800
B+io   18.20    1.35MB    988.66     7.690
```

6.4 Observations & Discussion (Part D)

- For the **CPU-intensive worker**, CPU utilization increases with more processes/threads until it saturates the available 2 CPU cores.
- Beyond this point, adding more processes or threads does not improve performance due to context-switching overhead.
- The **memory-intensive worker** shows increasing memory pressure as concurrency increases, with threads being slightly more memory-efficient due to shared address space.
- The **I/O-intensive worker** saturates disk bandwidth early; increasing concurrency mainly increases I/O wait rather than throughput.
- Overall, **threads scale better for CPU-bound workloads** on this system, while **process-based execution incurs higher overhead** at larger scales.