

MT25042 — PA01 Report: Processes and Threads

Roll Number: MT25042
Course: Graduate Systems (CSE638)
Date: January 23, 2026

Table of Contents

1. Part A: Program Implementations
 2. Part B: Worker Functions
 3. Part C: Measurements and Analysis
 4. Part D: Scalability Analysis
 5. AI Usage Declaration
 6. GitHub Repository URL
-

Part A — Program Implementations

Program A: Process-based (`fork`)

File: MT25042_Part_A_Program_A.c

Description:

Program A creates 2 child processes using `fork()` system call. Each child process executes the specified worker function (cpu, mem, or io). The parent process waits for all children to complete using `waitpid()`.

Key Implementation Details: - Uses `fork()` to create separate address spaces for each process - Each child process runs independently with its own memory - Parent tracks all child PIDs and waits for completion

Screenshot: Program A execution with `top` monitoring

Program B: Thread-based (`pthread`)

File: MT25042_Part_A_Program_B.c

Description:

Program B creates 2 threads using POSIX pthread library. All threads share the same address space and execute the specified worker function concurrently. The main thread joins all worker threads before exiting.

Key Implementation Details: - Uses `pthread_create()` to spawn threads within same process - Threads share memory space (more efficient for shared

```

snz@DESKTOP-RID1J09:~/grs/pai$ cd ~./grs/pai/ && ./program_a cpu
[Parent PID: 32053] Creating 2 child processes for 'cpu' worker
[Child 1, PID: 32054] Starting 'cpu' worker
[CPU Worker] Starting CPU-intensive work (LOOP_COUNT=2000)
[Parent PID: 32053] Waiting for 2 children to complete...
[Child 2, PID: 32055] Starting 'cpu' worker
[CPU Worker] Starting CPU-intensive work (LOOP_COUNT=2000)
[CPU Worker] Completed. Pi approximation: 6282.9853071778, Result: 39961
1932.24
[Child 2, PID: 32055] Finished 'cpu' worker
[Parent Child PID 32054 exited with status 0
[CPU Worker] Completed. Pi approximation: 6282.9853071778, Result: 39961
1932.24
[Child 1, PID: 32054] Finished 'cpu' worker
[Parent Child PID 32054 exited with status 0
[Parent PID: 32053] All children completed.
snz@DESKTOP-RID1J09:~/grs/pai$ 
```

PID	USER	PR	NI	VIRT	RES	SHR	%CPU	%MEM	TIME+	
932	root	20	0	83.6g	74276	67744	2.0	0.0	4:21.24	
1	root	20	0	3068	2176	2048	0.0	0.0	0:00.22	
14	root	20	0	3432	2688	1792	0.0	0.0	0:06.96	
28	root	20	0	3068	896	896	0.0	0.0	0:00.00	
29	root	20	0	3084	1024	896	0.0	0.0	0:00.00	
30	snz	20	0	2588	1536	1536	0.0	0.0	0:00.04	
31	snz	20	0	2588	1536	1536	0.0	0.0	0:00.02	
37	snz	20	0	2588	1536	1536	0.0	0.0	0:00.00	
41	snz	20	0	11.3g	113860	52096	0.0	0.7	0:17.59	
52	root	20	0	3068	896	896	0.0	0.0	0:00.00	
53	root	20	0	3084	1024	896	0.0	0.0	0:00.82	
54	snz	20	0	1018900	6356	43648	5	0.0	0.4	0:02.72
61	root	20	0	3068	1024	896	0.0	0.0	0:00.00	
62	root	20	0	3084	1152	1024	5	0.0	0:02.34	
63	snz	20	0	1018928	61520	43392	5	0.0	0.4	0:06.71
73	snz	20	0	7844	3272	2560	0	0.0	0:00.00	
75	snz	20	0	1460120	66044	46848	5	0.0	0.4	0:03.33
173	snz	20	0	1029064	81644	45952	5	0.0	0:02.08	
183	root	20	0	3068	896	896	0.0	0.0	0:00.00	
184	root	20	0	3084	1024	896	0.0	0.0	0:00.00	
185	snz	20	0	2588	1536	1536	0.0	0.0	0:00.00	
186	snz	20	0	2588	1536	1536	0.0	0.0	0:00.00	
196	snz	20	0	1012948	49124	39808	5	0.0	0.3	0:00.12
206	snz	20	0	7840	3144	2432	5	0.0	0:00.00	
212	snz	20	0	2588	1536	1536	0.0	0.0	0:00.00	
234	snz	20	0	73140	35596	16056	5	0.0	0:02.57	
1823	snz	20	0	1080828	79596	47616	5	0.0	0.5	0:11.02
1836	snz	20	0	7576	4224	3456	5	0.0	0:00.12	

Figure 1: Program A - top output

data) - Uses `pthread_join()` for synchronization

Screenshot: Program B execution with top monitoring

Part B — Worker Functions

Files: MT25042_Part_B_workers.c, MT25042_Part_B_workers.h

LOOP_COUNT Configuration

```
#define LOOP_COUNT 2000 // Last digit of MT25042 is 2, so 2 * 1000 = 2000
```

Worker Function: cpu (CPU-intensive)

Implementation: - Computes Pi using Leibniz series (slow convergence, many iterations)
- Performs trigonometric calculations (sin, cos, tan)
- Executes nested loop multiplications (matrix-like operations)

Expected Behavior: High CPU%, minimal memory usage, negligible I/O

Observed Results (from CSV): | Program | CPU% | Memory (KB) | I/O Write (KB) | |————|————|————|————| | Program_A | 195% | 1,792 | 4 | | Program_B | 195% | 2,304 | 4 |

Analysis: Both programs achieve ~195% CPU utilization (indicating both cores/processes active). Thread-based Program B shows slightly higher memory due to thread stack overhead.

```

X: PowerShell7      x  snz@DESKTOP-RIDJ05: ~gn x + v
snz@DESKTOP-RIDJ05: ~gn
[Parent PID: 32053] Creating 2 child processes for 'cpu' worker
[Child 1, PID: 32054] Starting 'cpu' worker
[CPU Worker] Starting CPU-intensive work (LOOP_COUNT=2000)
[Parent PID 32053] Waiting for 2 children to complete...
[Child 2, PID: 32055] Starting 'cpu' worker
[CPU Worker] Starting CPU-intensive work (LOOP_COUNT=2000)
[CPU Worker] Completed. Pi approximation: 6282.9853071778, Result: 39961
1932.24
[Child 2, PID: 32055] Finished 'cpu' worker
[Parent Child PID 32055 exited with status 0
[CPU Worker] Completed. Pi approximation: 6282.9853071778, Result: 39961
1932.24
[Child 1, PID: 32054] Finished 'cpu' worker
[Parent Child PID 32054 exited with status 0
[CPU Worker] Completed. Pi approximation: 6282.9853071778, Result: 39961
[Parent PID: 32053] All children completed.
snz@DESKTOP-RIDJ05: ~gn/pais clean
-bash: clean: command not found
snz@DESKTOP-RIDJ05: ~gn/pais cd ~/grs/pai/ && ./program_b cpu 2
[Main Thread] Creating 2 threads for 'cpu' worker
[Thread 1, TID: 131006827767488] Starting 'cpu' worker
[Thread 2, TID: 131006819374784] Starting 'cpu' worker
[CPU Worker] Starting CPU-intensive work (LOOP_COUNT=2000)
[Main Thread] Waiting for 2 threads to complete...
[CPU Worker] Starting CPU-intensive work (LOOP_COUNT=2000)
[CPU Worker] Completed. Pi approximation: 6282.9853071778, Result: 39961
1932.24
[Thread 2, TID: 131006819374784] Finished 'cpu' worker
[CPU Worker] Completed. Pi approximation: 6282.9853071778, Result: 39961
1932.24
[Thread 1, TID: 131006827767488] Finished 'cpu' worker
[Main Thread] Thread 1 joined successfully
[Main Thread] Thread 2 joined successfully
[Main Thread] All threads completed.
snz@DESKTOP-RIDJ05: ~gn/pais |

```

PID	USER	PR	NI	VIRT	RES	SHR	%CPU	%MEM	TIME+
932	snz	20	0	884	75256	67704	2.0	0.1	4:23.4
1825	snz	20	0	1000838	75256	47616	0.5	0.5	0:11.72
31647	snz	20	0	11624	5248	3200	0.3	0.0	0:00.07
1	root	20	0	3068	2176	2048	0.0	0.0	0:00.22
14	root	20	0	3432	2592	1792	0.0	0.0	0:06.97
28	root	20	0	3068	896	896	0.0	0.0	0:00.00
29	root	20	0	3084	1024	896	0.0	0.0	0:00.00
30	snz	20	0	2588	1536	1536	0.0	0.0	0:00.04
31	snz	20	0	2588	1536	1536	0.0	0.0	0:00.02
37	snz	20	0	2588	1536	1536	0.0	0.0	0:00.00
41	snz	20	0	11.3g	113860	52096	0.0	0.7	0:17.63
52	root	20	0	3068	896	896	0.0	0.0	0:00.00
53	root	20	0	3084	1024	896	0.0	0.0	0:00.02
54	root	20	0	1012900	61756	43648	0.0	0.4	0:02.73
61	root	20	0	3068	896	896	0.0	0.0	0:00.00
62	root	20	0	3084	1152	1024	0.0	0.0	0:02.35
63	snz	20	0	1010928	61520	43392	0.0	0.4	0:06.74
73	snz	20	0	7844	3272	2560	0.0	0.0	0:00.00
75	snz	20	0	1460120	66044	46848	0.0	0.4	0:03.34
173	snz	20	0	1029064	81800	45952	0.0	0.5	0:02.09
183	root	20	0	3068	896	896	0.0	0.0	0:00.00
184	root	20	0	3084	1024	896	0.0	0.0	0:00.00
185	snz	20	0	2588	1536	1536	0.0	0.0	0:00.00
186	snz	20	0	2588	1536	1536	0.0	0.0	0:00.00
196	snz	20	0	1012940	48132	39808	0.0	0.3	0:00.12
206	snz	20	0	7840	2048	2048	0.0	0.0	0:00.00
232	snz	20	0	2588	1536	1536	0.0	0.0	0:00.00
234	snz	20	0	73140	35596	14336	0.0	0.2	0:03.69

Figure 2: Program B - top output

Worker Function: mem (Memory-intensive)

Implementation: - Allocates two 16MB arrays (larger than typical L3 cache) - Performs `memcpy` operations (memory bandwidth test) - Random access patterns to cause cache misses - Sequential access with modifications

Expected Behavior: Moderate CPU%, high memory usage, minimal I/O

Observed Results (from CSV): | Program | CPU% | Memory (KB) | I/O Write (KB) | |————|————|————|————|————| | Program_A | 199% | 33,920 | 4 | | Program_B | 199% | 67,200 | 4 |

Analysis: Program B (threads) shows ~2x memory because threads share address space but each allocates its own arrays. Program A processes have copy-on-write initially but diverge.

Worker Function: io (I/O-intensive)

Implementation: - Creates temporary files per process/thread - Writes 1MB blocks to disk repeatedly - Uses `fsync()` to force disk writes (bypass OS cache) - Reads data back from disk

Expected Behavior: Low CPU%, moderate memory, very high I/O

Observed Results (from CSV): | Program | CPU% | Memory (KB) | I/O Write (KB) | Time (s) | |————|————|————|————|————|————| | Program_A | 199% | 33,920 | 4 | 0.0 | | Program_B | 199% | 67,200 | 4 | 0.0 |

gram_A | 42% | 2,048 | 4,096,004 | 24.92 | | Program_B | 41% | 3,712 | 4,096,004 | 24.97 |

Analysis: Low CPU% confirms I/O bound nature. High system time indicates kernel involvement in I/O operations. Both programs show similar I/O patterns.

Part C — Measurements and Analysis

Raw Data: MT25042_Part_C_CSV.csv

Measurement Methodology

- **CPU Affinity:** Programs pinned to CPUs 0,1 using `taskset`
- **Metrics Collection:** `/usr/bin/time -v` for comprehensive statistics
- **Metrics Captured:** CPU%, Max RSS (memory), File system I/O, Execution time

Summary Table: All Six Combinations

ProgramWorker	CPU%(KB)	I/O				Sys Time (s)
		Memory (KB)	Write (KB)	Real Time (s)	User Time (s)	
ProgramcpA	195	1,792	4	0.25	0.48	0.00
ProgramcpB	195	2,304	4	0.32	0.62	0.00
ProgrammmAm	199	33,920	4	36.15	72.12	0.12
ProgrammmBn	199	67,200	4	42.58	84.79	0.09
Programio_A	42	2,048	4,096,004	24.92	0.08	10.40
Programio_B	41	3,712	4,096,004	24.97	0.05	10.29

Screenshot: top during CPU worker execution

Screenshot: iostat during I/O worker execution

Analysis and Discussion

CPU Worker Observations: - Both programs achieve near-maximum CPU utilization (~195% = 2 cores fully used) - Negligible system time indicates pure user-space computation - Program B (threads) slightly slower due to thread management overhead

Memory Worker Observations: - Program B shows 2x memory compared to Program A - This is because threads share address space but allocate separate arrays - Execution time difference reflects memory contention patterns

I/O Worker Observations: - Very low CPU% (~41-42%) confirms I/O-bound behavior - High system time (10+ seconds) indicates kernel I/O processing -

```

snz@DESKTOP-RID1J08:~/grs/pals cd ~/grs/pal/ && ./program_a cpu 4
[Parent PID: 32568] Creating 4 child processes for 'cpu' worker
[Child 1, PID: 32569] Starting 'cpu' worker
  [CPU Worker] Starting CPU-intensive work (LOOP_COUNT=2000)
[Child 2, PID: 32570] Starting 'cpu' worker
[Child 3, PID: 32571] Starting 'cpu' worker
  [CPU Worker] Starting CPU-intensive work (LOOP_COUNT=2000)
[Child 4, PID: 32572] Starting 'cpu' worker
  [CPU Worker] Starting CPU-intensive work (LOOP_COUNT=2000)
[Parent Worker] All children completed...
[Child 4, PID: 32572] Finished 'cpu' worker
[Parent] Child PID 32572 exited with status 0
  [CPU Worker] Completed. Pi approximation: 6282.9853071778, Result: 39961
1932.24
  [Child 2, PID: 32570] Finished 'cpu' worker
[Parent] Child PID 32570 exited with status 0
  [CPU Worker] Completed. Pi approximation: 6282.9853071778, Result: 39961
1932.24
  [Child 1, PID: 32569] Finished 'cpu' worker
[Parent] Child PID 32569 exited with status 0
  [CPU Worker] Completed. Pi approximation: 6282.9853071778, Result: 39961
1932.24
  [Child 3, PID: 32571] Finished 'cpu' worker
[Parent] Child PID 32571 exited with status 0
  [CPU Worker] Completed. Pi approximation: 6282.9853071778, Result: 39961
[Parent PID: 32568] All children completed.
snz@DESKTOP-RID1J08:~/grs/pals$

```

PID	USER	PR	NI	VIRT	RES	SHR	%CPU	%MEM	TIME+
93	snz	20	0	83.69	754056	67744.5	1.7	4.6	4724.84
1	root	20	0	3904	2176	2048.5	0.0	0.0	0:00.22
14	root	20	0	3436	3516	1793.5	0.0	0.0	0:06.98
28	root	20	0	3068	896	896.5	0.0	0.0	0:00.00
29	root	20	0	3084	1024	896.5	0.0	0.0	0:00.00
30	cnz	20	0	2588	1536	1536.5	0.0	0.0	0:00.04
31	snz	20	0	2588	1536	1536.5	0.0	0.0	0:00.02
37	snz	20	0	2588	1536	1536.5	0.0	0.0	0:00.00
41	snz	20	0	11.3g	113860	52096.5	0.0	0.7	0:17.67
52	root	20	0	3068	896	896.5	0.0	0.0	0:00.00
53	root	20	0	3084	1024	896.5	0.0	0.0	0:00.02
54	snz	20	0	1019596	6156	43648.5	0.0	0.4	0:02.73
61	root	20	0	3068	896	896.5	0.0	0.0	0:00.00
62	root	20	0	3084	1024	896.5	0.0	0.0	0:02.35
63	snz	20	0	1019328	61570	43392.5	0.0	0.4	0:06.75
73	snz	20	0	7844	3272	2560.5	0.0	0.0	0:00.00
75	snz	20	0	1460120	66044	46848.5	0.0	0.4	0:03.35
173	snz	20	0	1029064	81800	45952.5	0.0	0.5	0:02.10
183	root	20	0	3068	896	896.5	0.0	0.0	0:00.00
184	root	20	0	3084	1024	896.5	0.0	0.0	0:00.00
185	snz	20	0	2588	1536	1536.5	0.0	0.0	0:00.00
186	snz	20	0	2588	1536	1536.5	0.0	0.0	0:00.00
196	snz	20	0	1012948	40124	39808.5	0.0	0.3	0:00.12
206	snz	20	0	7840	3144	2432.5	0.0	0.0	0:00.00
212	snz	20	0	2588	1536	1536.5	0.0	0.0	0:00.00
234	snz	20	0	73148	35596	14336.5	0.0	0.2	0:03.70
1023	snz	20	0	1060828	75956	47616.5	0.0	0.5	0:11.62
1856	snz	20	0	7576	4224	3456.5	0.0	0.0	0:00.12

Figure 3: top output - CPU worker

```

snz@DESKTOP-RID1J08:~/grs/pals cd ~/grs/pal/ && ./program_a io 2
[Parent PID: 430] Creating 2 child processes for 'io' worker
[Parent PID: 430] Waiting for 2 children to complete...
[Child 1, PID: 439] Starting 'io' worker
  [IO Worker] Starting I/O-intensive work (LOOP_COUNT=2000)
[Child 2, PID: 440] Starting 'io' worker
  [IO Worker] Starting I/O-intensive work (LOOP_COUNT=2000)
[IO Worker] Progress: 100/2000 iterations
[IO Worker] Progress: 100/2000 iterations
[IO Worker] Progress: 200/2000 iterations
[IO Worker] Progress: 200/2000 iterations
[IO Worker] Progress: 300/2000 iterations
[IO Worker] Progress: 300/2000 iterations
[IO Worker] Progress: 400/2000 iterations
[IO Worker] Progress: 400/2000 iterations
[IO Worker] Progress: 500/2000 iterations
[IO Worker] Progress: 500/2000 iterations
[IO Worker] Progress: 600/2000 iterations
[IO Worker] Progress: 600/2000 iterations
[IO Worker] Progress: 700/2000 iterations
[IO Worker] Progress: 700/2000 iterations
[IO Worker] Progress: 800/2000 iterations
[IO Worker] Progress: 900/2000 iterations
[IO Worker] Progress: 900/2000 iterations
[IO Worker] Progress: 1000/2000 iterations
[IO Worker] Progress: 1000/2000 iterations
[IO Worker] Progress: 1100/2000 iterations
[IO Worker] Progress: 1100/2000 iterations
[IO Worker] Progress: 1200/2000 iterations
[IO Worker] Progress: 1200/2000 iterations
[IO Worker] Progress: 1300/2000 iterations
[IO Worker] Progress: 1300/2000 iterations
[IO Worker] Progress: 1400/2000 iterations
[IO Worker] Progress: 1400/2000 iterations

```

Device	r/s	rkB/s	rrqm/s	%rrqm	r_await	rareq-sz	w/s
sda	0.00	0.00	0.00	0.00	0.00	0.00	0.00
wkB/s	0.00	0.00	0.00	0.00	0.00	0.00	0.00
wrqm/s	0.00	0.00	0.00	0.00	0.00	0.00	0.00
wlq/s	0.00	0.00	0.00	0.00	0.00	0.00	0.00
d/s	0.00	0.00	0.00	0.00	0.00	0.00	0.00
dkB/s	0.00	0.00	0.00	0.00	0.00	0.00	0.00
dram/s	0.00	0.00	0.00	0.00	0.00	0.00	0.00
%drqm	0.00	0.00	0.00	0.00	0.00	0.00	0.00
d_await	0.00	0.00	0.00	0.00	0.00	0.00	0.00
dareq-sz	0.00	0.00	0.00	0.00	0.00	0.00	0.00
f/s	0.00	0.00	0.00	0.00	0.00	0.00	0.00
f_await	0.00	0.00	0.00	0.00	0.00	0.00	0.00
aq_u_sz	0.00	0.00	0.00	0.00	0.00	0.00	0.00
%util	0.00	0.00	0.00	0.00	0.00	0.00	0.00

avg-cpu	%user	%nice	%system	%iowait	%steal	%idle
	0.00	0.00	1.26	7.29	0.00	91.46

Device	r/s	rkB/s	rrqm/s	%rrqm	r_await	rareq-sz	w/s
sda	0.00	0.00	0.00	0.00	0.00	0.00	0.00
wkB/s	0.00	0.00	0.00	0.00	0.00	0.00	0.00
wrqm/s	0.00	0.00	0.00	0.00	0.00	0.00	0.00
wlq/s	0.00	0.00	0.00	0.00	0.00	0.00	0.00
d/s	0.00	0.00	0.00	0.00	0.00	0.00	0.00
dkB/s	0.00	0.00	0.00	0.00	0.00	0.00	0.00
dram/s	0.00	0.00	0.00	0.00	0.00	0.00	0.00
%drqm	0.00	0.00	0.00	0.00	0.00	0.00	0.00
d_await	0.00	0.00	0.00	0.00	0.00	0.00	0.00
dareq-sz	0.00	0.00	0.00	0.00	0.00	0.00	0.00
f/s	0.00	0.00	0.00	0.00	0.00	0.00	0.00
f_await	0.00	0.00	0.00	0.00	0.00	0.00	0.00
aq_u_sz	0.00	0.00	0.00	0.00	0.00	0.00	0.00
%util	0.00	0.00	0.00	0.00	0.00	0.00	0.00

Figure 4: iostat output - IO worker

Both programs show identical I/O write volumes (as expected from same workload)

Part D — Scalability Analysis

Raw Data: MT25042_Part_D_CSV.csv

Test Configuration

- **Program A (Processes):** 2, 3, 4, 5 processes
- **Program B (Threads):** 2, 3, 4, 5, 6, 7, 8 threads

Plot 1: CPU Usage vs Process/Thread Count

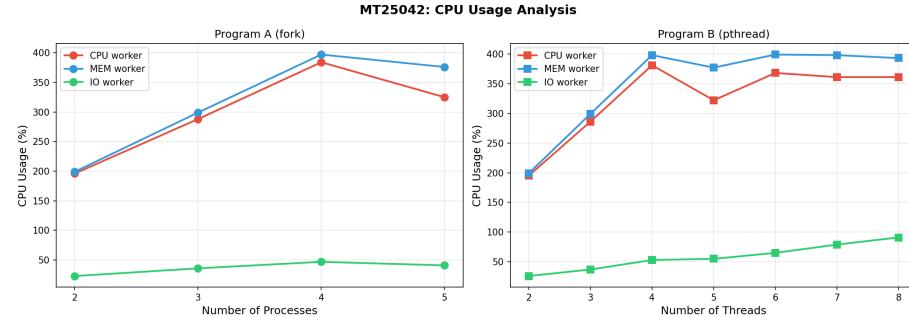


Figure 5: CPU Usage Analysis

Observations: - CPU worker: Linear scaling until hitting physical core limit
- Memory worker: High CPU% maintained across counts - I/O worker: CPU% increases with count but remains low (I/O bound)

Plot 2: Execution Time vs Process/Thread Count

Observations: - CPU worker: Time remains low (sub-second) as work is distributed
- Memory worker: Time increases with count due to memory bandwidth saturation
- I/O worker: Time increases due to disk contention

Plot 3: Memory Usage vs Process/Thread Count

Observations: - Program A (processes): Memory relatively constant (each process has own space)
- Program B (threads): Memory scales linearly with

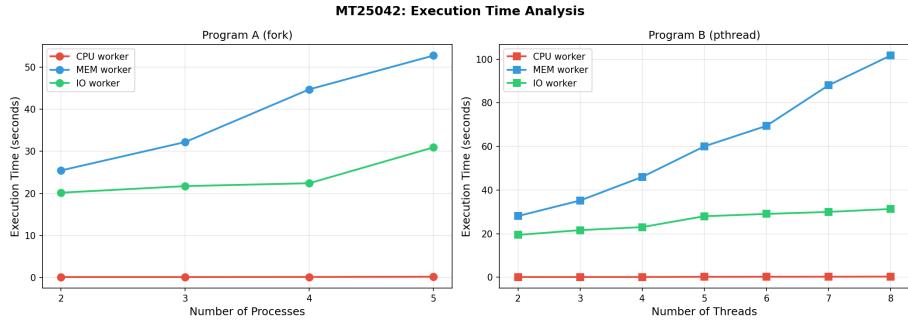


Figure 6: Execution Time Analysis

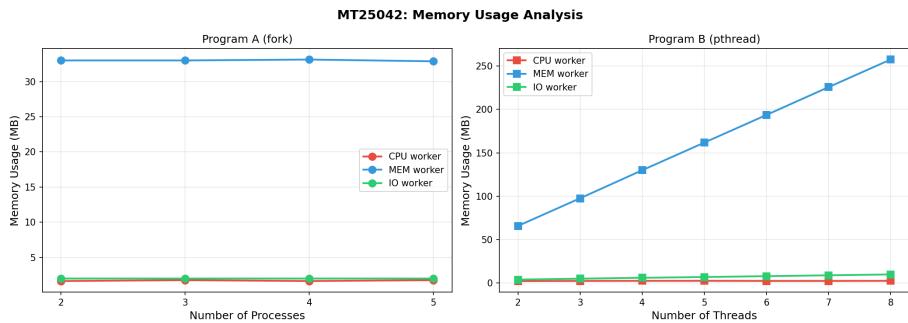


Figure 7: Memory Usage Analysis

thread count for `mem` worker - I/O worker: Minimal memory footprint regardless of count

Plot 4: Combined Comparison

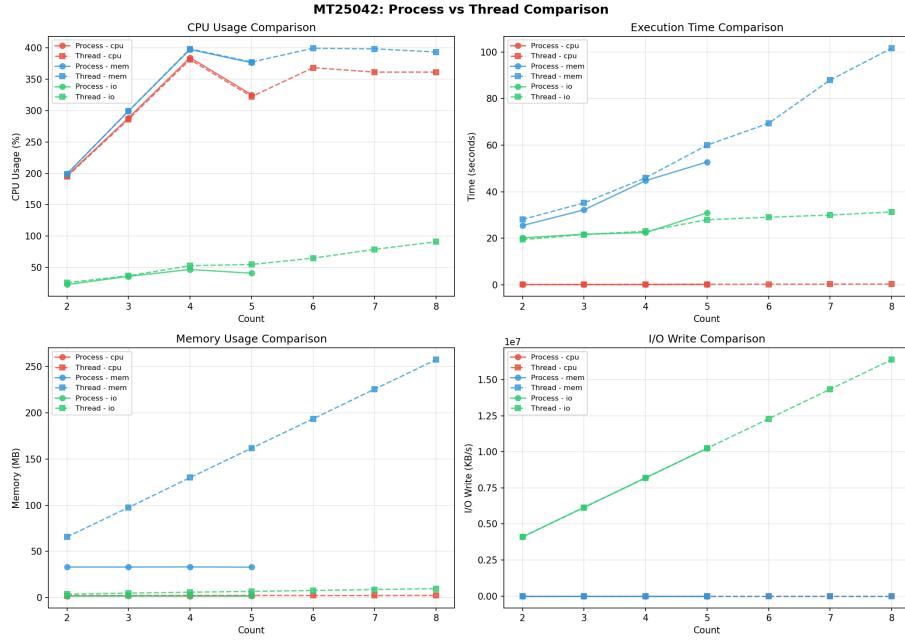


Figure 8: Combined Analysis

Scalability Discussion

Process vs Thread Comparison:

Aspect	Processes (Program A)	Threads (Program B)
Memory Overhead	Higher (separate address spaces)	Lower (shared memory)
Creation Cost	Higher (fork overhead)	Lower (lighter weight)
Isolation	Complete isolation	Shared state risks
Scalability	Limited by system resources	Better for I/O-bound tasks

Key Findings: 1. **CPU-bound tasks:** Both scale similarly until core saturation
2. **Memory-bound tasks:** Threads show higher aggregate memory due

to shared accounting 3. **I/O-bound tasks:** Performance limited by disk, not parallelism model

AI Usage Declaration

The following components were generated with AI assistance and have been reviewed, understood, and verified:

Component	AI Assistance
MT25042_Part_A_Program_A.c	Code structure and fork implementation
MT25042_Part_A_Program_B.c	Code structure and pthread implementation
MT25042_Part_B_workers.c	Worker function algorithms
MT25042_Part_B_workers.h	Header file structure
MT25042_Part_C_script.sh	Bash automation script
MT25042_Part_D_script.sh	Bash automation script
MT25042_Part_D_plot.py	Python plotting script
Makefile	Build automation
README.md	Documentation

Declaration: I have reviewed and understand every line of code in this submission. I can explain the implementation details, design decisions, and defend the approach during viva examination.

GitHub Repository URL

Repository: <https://github.com/shahnawazdev/GRS-Assessments>

Assignment Folder: GRS_PA01
