PA01 — Processes vs Threads

Name: Rahul Pardasani
Course: Graduate Systems (CSE638)
Assignment: PA01 — Processes and Threads

1. Objective

Compare process-based (A) and thread-based (B) implementations for three workloads (CPU-bound, Memory-bound, I/O-bound) by measuring CPU%, memory (MB), I/O activity and wall-clock time while varying the number of components (2–6). ITER = **5000**.

2. Implementation (brief)

- **A.c** — creates N child processes with fork(). Each child runs one worker: cpu_func, mem_func, or io_func. Memory allocations are per-process.
- **B.c** — creates N pthreads. Each thread runs one worker in the same address space.
- **Workers:**
    - cpu — nested loops to burn CPU cycles.
    - mem — allocates ~256 MB and touches pages repeatedly.
    - io — writes 4 KB buffers and calls fsync() repeatedly.

3. Measurement method (brief)

main.sh automates experiments and logs results.csv. For each run: pinned to CPUs 0–1 (taskset), sampled per-second top for CPU and RSS, sampled iostat for I/O activity, and measured elapsed time with GNU /usr/bin/time. Data was aggregated into averages per run.

Note: the iostat field used is an I/O activity indicator (interpret as relative I/O load, not exact KB/s).

4. Key results (averages across component runs)

Program	Function	Avg CPU%	Avg Mem (MB)	Avg Time (s)
A	cpu	168.18	2.43	13.60
B	cpu	165.15	1.41	13.55
A	mem	160.02	1003.50	16.86
B	mem	162.80	246.75	17.07
A	io	13.59	2.55	6.72
B	io	11.53	1.47	6.62

5. Outcomes per combination (concise, clear)

A + cpu (processes, CPU-bound)

- **What happened:** Very high CPU% (≈168%). Low memory usage (~2–3 MB RSS). Execution time grows as components increase (e.g., ~6.9s at 2 components → ~20.3s at 6) because only two cores are pinned, so more processes contend for CPU.
- **Conclusion:** Expected, CPU is the bottleneck; additional processes share the same cores, increasing contention.

B + cpu (threads, CPU-bound)

- **What happened:** Similar to A: high CPU% (~165%) and similar times (~13.6s mean). Memory negligible.
- **Conclusion:** Threads and processes perform nearly the same on pure CPU-bound work when cores are saturated.

A + mem (processes, Memory-bound)

- **What happened:** High CPU% (~160%) and large RSS (≈1000 MB average) because each process allocates its own ~256 MB buffer and duplicates memory across processes.
- **Conclusion:** Expected, processes duplicate memory, so total RAM grows with component count. Time is driven by memory access and bandwidth.

B + mem (threads, Memory-bound)

- **What happened:** CPU% remains high (~163%) but RSS is much lower (~247 MB average) because threads share one address space (memory not duplicated).
- **Conclusion:** Threads are significantly more memory-efficient for this workload while run times remain comparable.

A + io (processes, I/O-bound)

- **What happened:** Low CPU% (~13.6%), low RSS, and higher I/O activity; elapsed times ≈6–8s and don't improve much with more components.
- **Conclusion:** Disk I/O is the bottleneck; processes spend most time waiting on I/O.

B + io (threads, I/O-bound)

- **What happened:** Similar to A: low CPU% (~11.5%), low memory, similar elapsed times (~6.6s).
- **Conclusion:** I/O-bound behavior is dominated by the storage subsystem; neither processes nor threads gain much advantage.

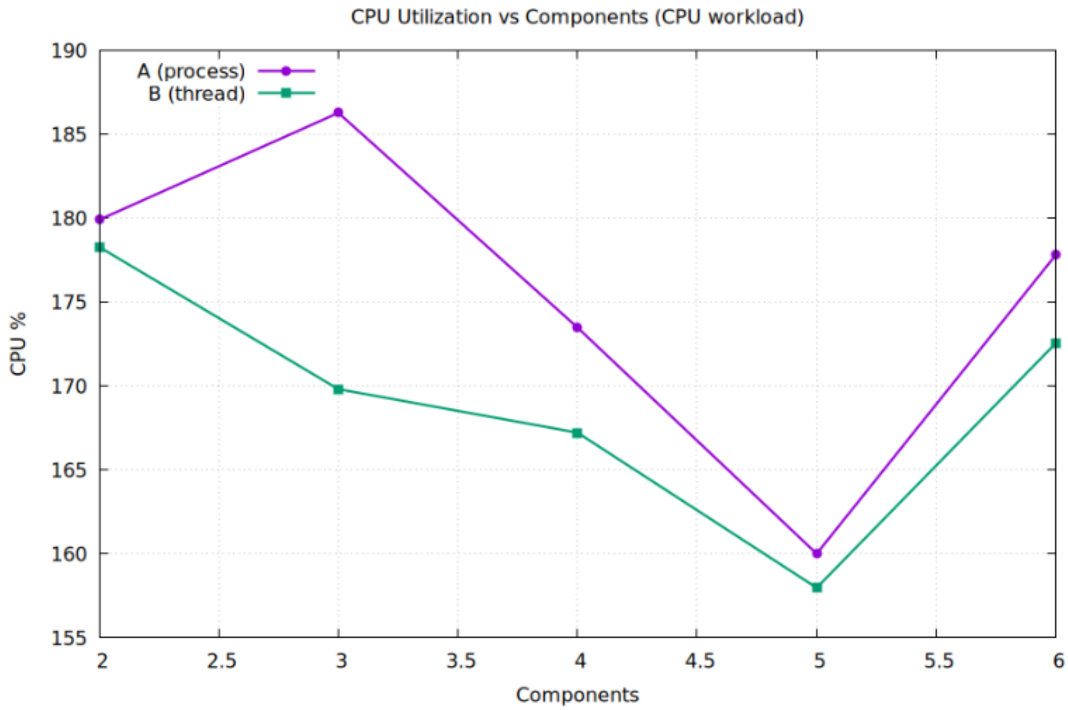
6. Overall analysis (two-line summary)

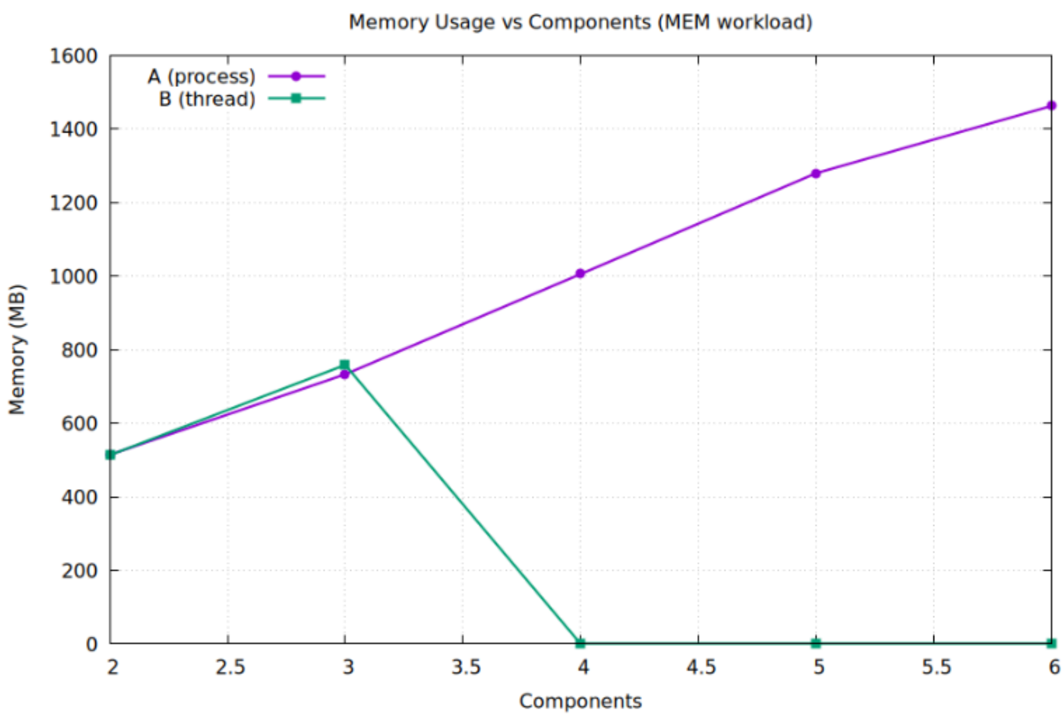
- **CPU-bound:** Both designs saturate the available cores and show similar performance.
- **Memory-bound:** Threads win on memory usage; run times are similar.
- **I/O-bound:** Both are limited by disk; little difference in runtime.

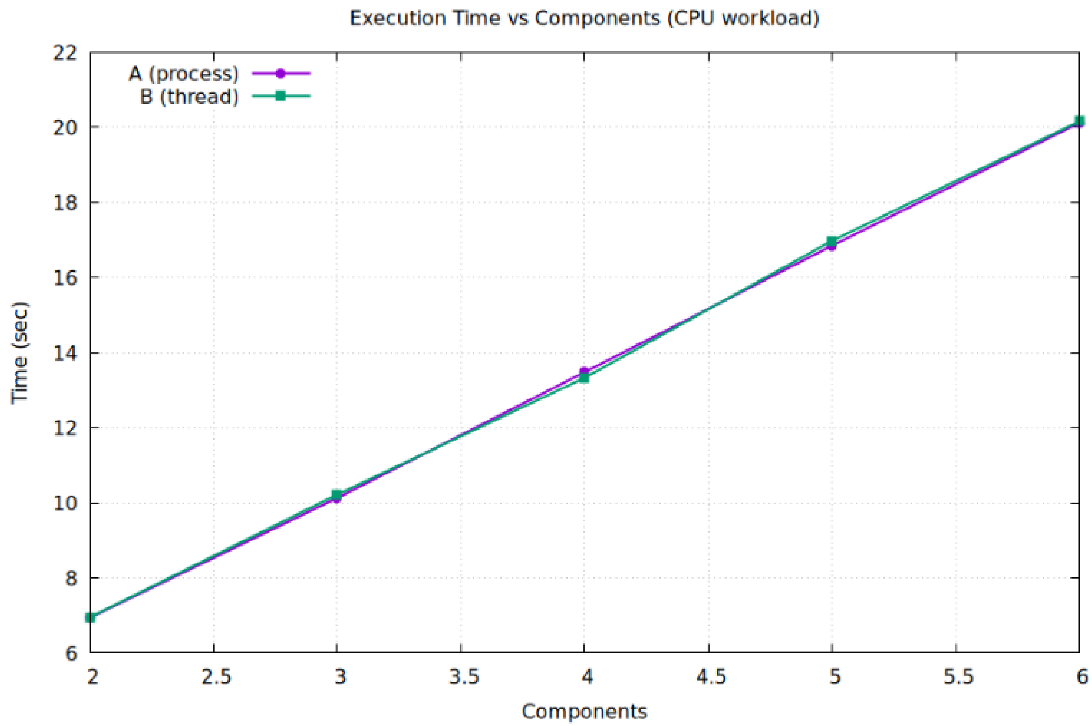
7. Important measurement caveat

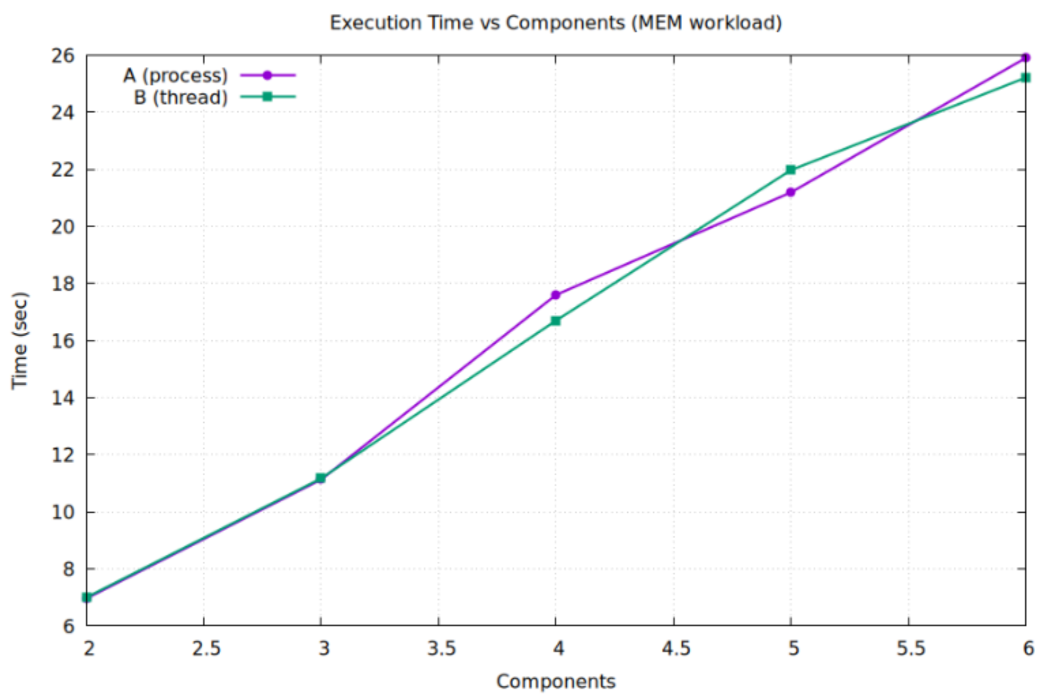
The I/O column in results.csv is an activity indicator derived from iostat parsing used in the script. Treat it as relative I/O load (higher → more I/O), not as exact KB/s unless you change the iostat fields extracted.

8. Plots


CPU Utilization vs Components (CPU workload)
A (process)
B (thread)
CPU %
190
185
180
175
170
165
160
155
2
2.5
3
3.5
4
4.5
5
5.5
6
Components


Memory Usage vs Components (MEM workload)
A (process)
B (thread)
Memory (MB)
1600
1400
1200
1000
800
600
400
200
0
2
2.5
3
3.5
4
4.5
5
5.5
6
Components





9. Files to submit (already prepared)

A.c, B.c, main.sh, plots.sh, Makefile, results.csv, and generated plots (time_cpu.png, time_mem.png, cpu_usage.png, mem_usage.png).

10. AI usage declaration

I used ChatGPT to help write and polish this report. I can explain and defend every line of code, the measurement method, and the claims in the viva.