Project 03 — Systematic Profiling of an LLM Inference Server (CPU-first)

Profile an LLM inference stack as a **server workload** under realistic resource constraints (VM + CPU). Your goal is to connect user-visible metrics (TTFT, p99 latency, tokens/s) to OS mechanisms (memory behavior, scheduling, IO) and to validate mitigations.

Team size **1–2**, duration **10 weeks**.

> ***Baseline expectation (this course): MS level.***

Scope / constraints

- **CPU-first** (do not assume GPUs).
- Must run on **laptop + Ubuntu VM**.
- Choose an inference implementation you can run locally, e.g.:
  - `llama.cpp` (recommended)
  - `ollama` (CPU mode if feasible)

Metrics (required)

Measure at least:

- **TTFT** (time to first token)
- **tokens/s** (steady-state generation)
- **p95/p99 request latency** (for a fixed prompt+generation length)

Also record:

- RSS / memory footprint
- major/minor faults (or approximations via `/proc` + `pidstat`)
- context switches / CPU utilization

Experiment variables (pick at least 3)

Examples:

- context length (short vs long prompts)
- output length
- concurrency (1 vs N requests)
- thread count
- model size / quantization (if available)
- cold start vs warm start (first request vs steady state)

Required OS/resource experiments

You must include:

1. **cgroup v2 CPU quota** experiment: show how throttling changes p99/TTFT and why.
2. **cgroup v2 memory limit** experiment: show behavior under tight memory (reclaim, faults, possible OOM) and why.

Evidence standard (MS baseline)

For **each** interference experiment and mitigation, you must provide:

1. **Two independent pieces of evidence (cross-layer observations)**
    - At least one must come from the **application/user space** (end-to-end latency percentiles, error rate, throughput, hop-level latency, etc.), and another from the **OS/resource control/K8s control plane** (cgroup/PSI/pidstat/iostat/kubectl events, etc.).
    - "Independent" means the two pieces should not both come from the same tool or the same type of metric (for example, two top screenshots do not count).
    - Examples:
        - App latency histogram/logs + cgroup v2 cpu.stat / memory.current
        - PSI (/proc/pressure/*) + iostat -x / pidstat
        - K8s events (kubectl describe pod) + cgroup stats
2. **One exclusionary control (controlled variable / counterexample)**
    - For the most likely alternative explanation, provide evidence showing that it is not the primary cause.
    - Examples
        - Stable iostat → not an I/O bottleneck
        - No throttling in cpu.stat → not a CPU quota issue
        - Low CPU PSI / low CPU usage in pidstat → not CPU contention
3. **Before/after percentiles + corresponding mechanism-level metric changes**
    - Report p50/p95/p99 (or at least p50/p99).
    - Also show one mechanism-level metric moved consistently with your explanation (e.g., `throttled_usec`, PSI, `await`, `oom_kill`).

Perf PMU counters may not work in VMs; don't depend on hardware cache events.

Mitigations (required)

Provide at least **two** mitigations, at least one in each category:

- **system-level:** warmup strategy, pinning/affinity, cgroup tuning, avoiding swapping, preloading model files
- **application-level:** admission control, batching (if possible), prompt constraints, caching

Validate mitigations with before/after plots and an explanation tied to mechanisms.