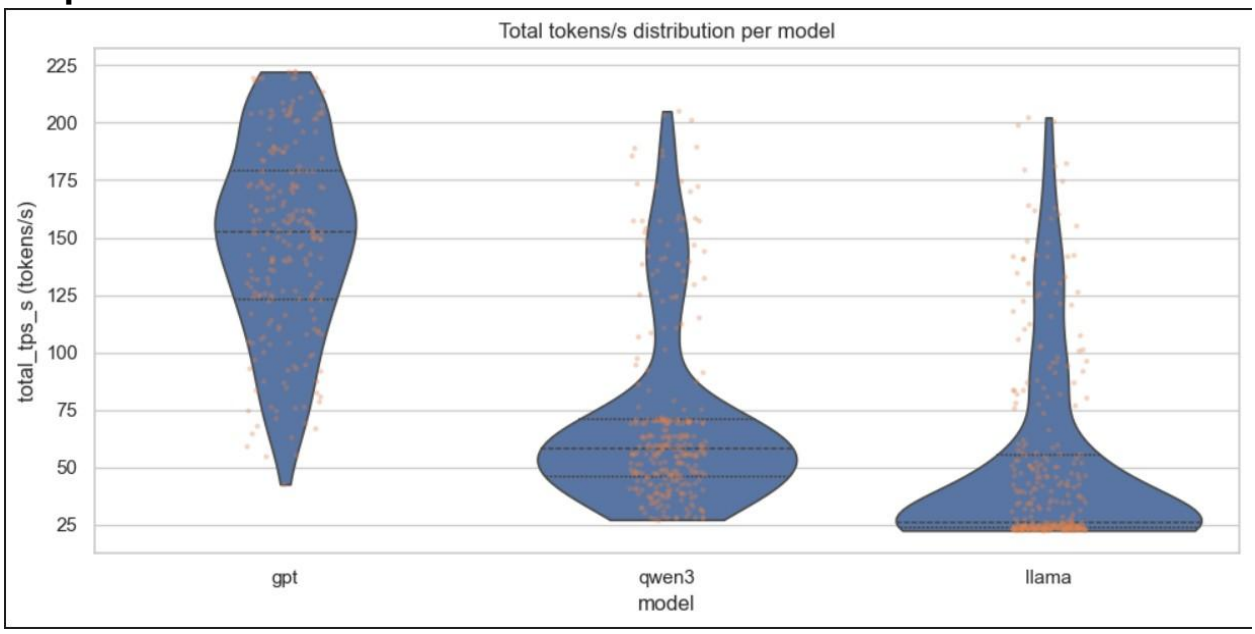
Graph Analysis and Optimization Directions

Graph A : Token/s Distribution



Graph analysis

- GPT: very high throughput.
- Qwen3: medium typical throughput, with a few spikes.
- Llama: low typical throughput, with a long tail (rare spikes).

Likely vulnerabilities

1 **Instability / high variance (especially Qwen3 & Llama)**

When a model is often low and sometimes very high, it usually indicates that performance depends heavily on external conditions (batching, queueing, sequence lengths, contention).

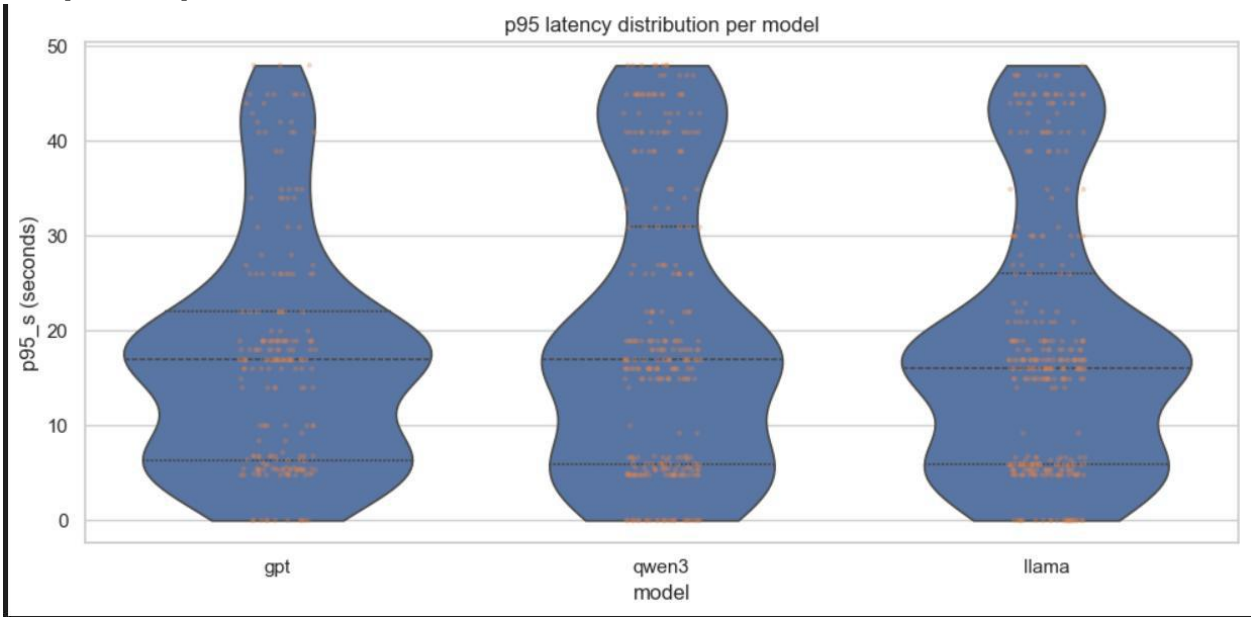
2 **Sensitivity to load conditions**

"Long tails" mean the model can fall into a degraded regime (saturated GPU, scheduler waiting, less favorable batching).

Plausible explanations

- **Variable output length**: observed tokens/s changes (longer decode means more stable; very short means noisy).
- **Batch size** changes: some moments benefit from a perfect batch, others do not.
- **Warm-up**: slower start, then stable regime.

Graph B : p95 Distribution



Graph analysis

- Typical p95 15-20s on all 3 models.
- Presence of a low mode (5-7s) and a high mode (40-45s).

Likely vulnerabilities

1 **Queueing / congestion (p95 goes very high)**

A p95 at 40-45s is typical of a queue or a bottleneck (requests are waiting).

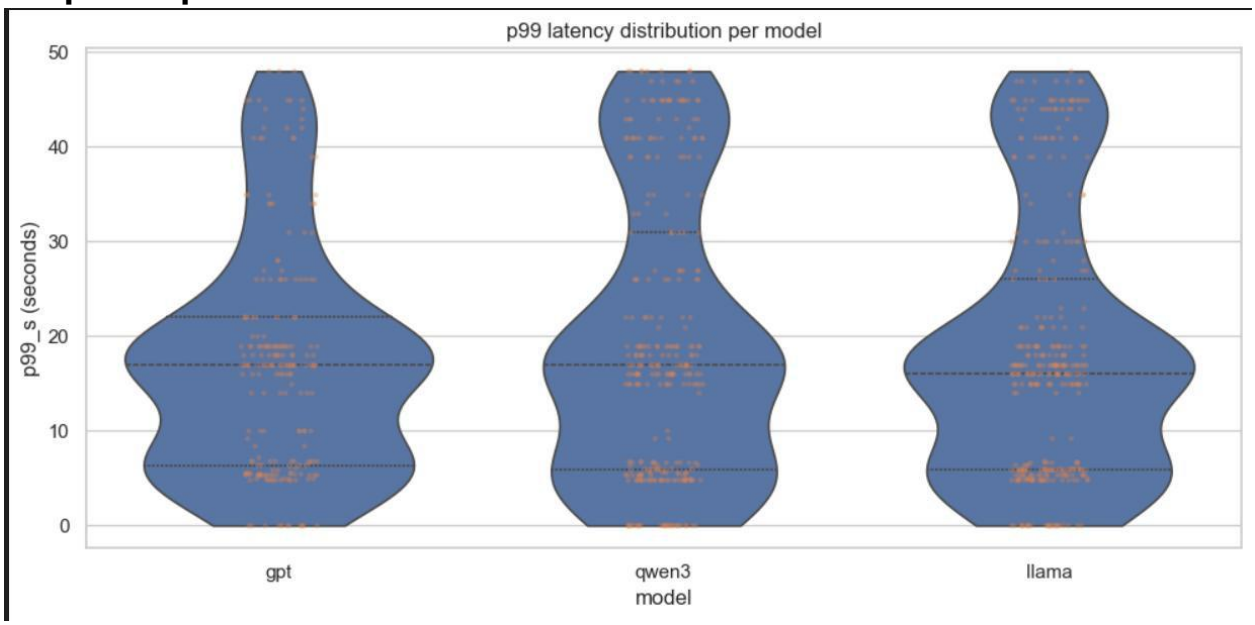
2 **Multiple regimes**

The system switches between “normal” and “stress” (modes visible on the violin plot).

Plausible explanations

- **GPU saturation** means *num_requests_waiting* would increase.
- **Prompt length / output length variability**: some requests are much more expensive.
- **Batching:** under load, a request may wait for the next batch.

Graph C : p99 Distribution



Graph analysis

- p99 is very close to p95, with the same spikes up to ~45–48s.

Likely vulnerabilities

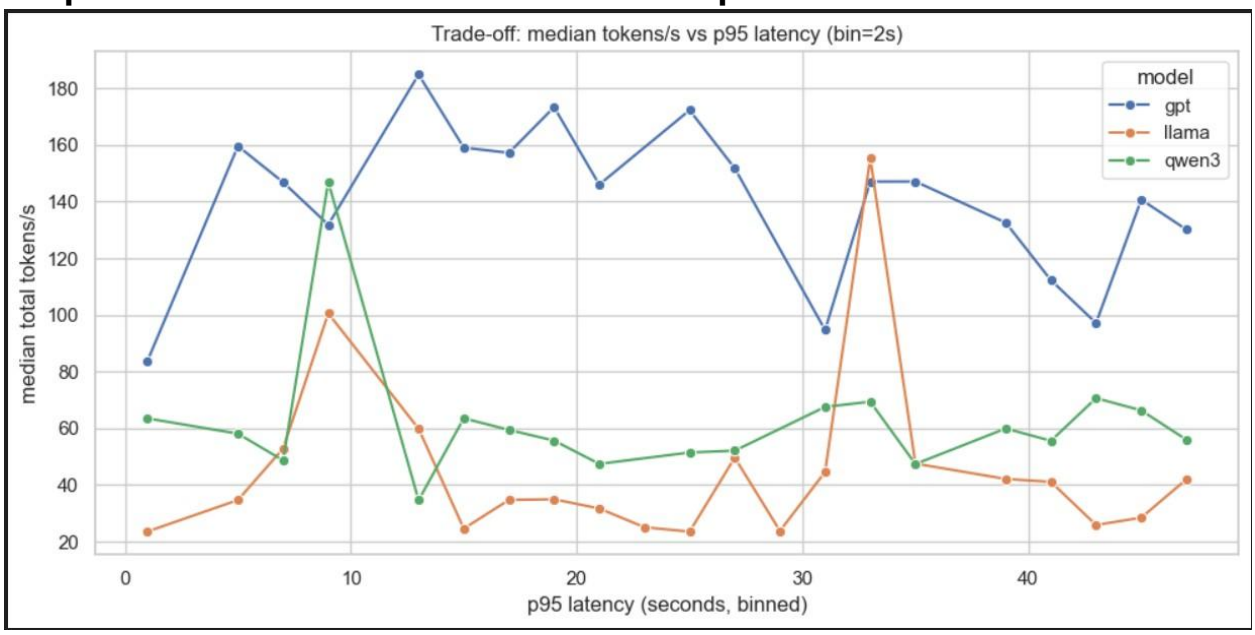
High and persistent worst-case

High p99 means a few requests regularly experience extreme delays.

Plausible explanations

- Head-of-line blocking: large requests slow down others (batch / scheduler).

Graph D : Trade-off: median tokens/s vs p95



Graph analysis

- GPT keeps high throughput even when p95 increases.
- Qwen3 is stable but low.
- Llama is very low with one isolated peak.

Likely vulnerabilities

1 **Llama: frequent “degraded” regime**

Low tokens/s across almost all bins, inefficiency risk (heavy model, config/quantization).

2 **Qwen3: performance ceiling**

It does not exploit favorable regimes as much as GPT (may be limited by config: KV cache, tensor parallel, quantization).

Plausible explanations

- GPT might be served on a more optimized setup / more GPUs / better parallelization.
- Qwen3/Llama may be more sensitive to length (heavy prefill).

Parameters to tune per model (vLLM optimization hypotheses)

Important: these settings are hypothetical. The goal is to target the observed symptoms: queueing, variability, throughput ceiling, and tail latency.

1) GPT (high throughput but p95/p99 latency sometimes high)

Goal: reduce tail latency without killing throughput

- Reduce the queue (if load varies):

  --max-num-seqs (lower if too much concurrency to have fewer requests waiting).

  --max-num-batched-tokens (tune: too high can increase latency for short requests).

- Stabilize prefill:

  --enable-chunked-prefill (better responsiveness with long prompts).

- Limit context length if possible:

  --max-model-len (if you don’t need a huge context).

- Tune GPU memory:

  --gpu-memory-utilization (too aggressive can create memory).

2) Qwen3 (medium throughput and spikes)

Goal: increase typical throughput and reduce variance

- Improve batching efficiency (if many small requests):

  --max-num-batched-tokens (increase gradually).

  --max-num-seqs (increase if GPU under-utilized, decrease if queueing).

- Optimize multi-GPU parallelization (if the model is very large):

  --tensor-parallel-size (TP): test values that better saturate GPUs.

  --pipeline-parallel-size (PP): if the model is huge and TP alone is not enough.

- KV cache / memory :

  --kv-cache-dtype (FP8/FP16 depending on support) to reduce memory and increase capacity.

  --gpu-memory-utilization.

- Reduce prefill overhead:

  --enable-chunked-prefill (if prompts are long).

  --enable-prefix-caching

3) Llama (low typical throughput + rare spikes → frequent “degraded” regime)

Goal: get out of the low regime and improve the “typical” level

- Check/tune quantization:

  --quantization (AWQ/GPTQ depending on the build): can significantly increase throughput, sometimes at a small quality cost.

- Parallelism:

  --tensor-parallel-size / --pipeline-parallel-size

- Batching:

  --max-num-batched-tokens (increase if Llama never benefits from good batches).

  --max-num-seqs (tune to avoid exploding p95/p99).

- Reduce latency variability:

  --enable-chunked-prefill

  --max-model-len (if unnecessarily large, it costs memory and can hurt performance).

- Memory/KV :

  --gpu-memory-utilization and --kv-cache-dtype to avoid situations where the cache is too constrained