LLM INFERENCE, MEASURED · PART FIVE

vLLM tuning cheat sheet

Six settings, what each one is really a knob on, and which way to move it. Defaults move between releases — run `vllm --version` before trusting any of these.

v0.27.1 PINNED

THE SIX SETTINGS

▼ lower it ▲ raise it ● leave it, but check something

`--max-num-seqs`

`1024 / 256` by card

A ceiling on the batch, not the batch you get.

- ▼ Steadier per-token speed for those already in. Costs throughput, and lengthens the queue: the scheduler stops admitting at the ceiling.
- ▲ Rarely moves anything: capacity binds first. Exception, short contexts on a 256-default card.

`--gpu-memory-utilization`

`0.92`

The share of the card vLLM will touch at all. Weights, activation peak and graph buffers come out of *inside* it.

- ▲ 0.95 is the cheapest capacity there is, though nothing like the largest.
- ● You court fragmentation or a co-tenant, not a long request: one that cannot get blocks is preempted, not crashed.

`--max-num-batched-tokens`

`16,384 / 8,192 / 2,048` by card and launch

The per-step budget. Requests already generating come off the top, one token each.

- ▼ Protects per-token speed for those already generating. Costs the newcomer's first token.
- ▲ Does the reverse. No setting is correct. Floor: at least `--max-num-seqs`.

`--enable-prefix-caching`

`on`

Reuse, exact-match from the first token. Pays on a shared system prompt or a multi-turn chat, nothing on traffic that varies at the front.

- ● You do not turn this on. Check nobody turned it off.
- ● Check the hit rate before trusting a benchmark: warm and cold measure two different products.

`--kv-cache-dtype fp8`

`off`

Cache bytes. Halving them roughly doubles the conversations that fit.

- ▲ Take it for the capacity.
- ● Then two checks: the backend line, because one without FP8 support falls back quietly; and your own eval, because the published accuracy evidence never touched the cache.

`--enforce-eager`

`off`

Compile and CUDA graph capture. Fast start, slower decode.

- ● Development only. A surprising amount of "vLLM is slow" is this, left on after debugging.
- ▲ One exception: where capacity binds, the graph buffers it skips go back to the KV cache. Measure both ways.

THE TWO DEFAULTS THAT DEPEND ON YOUR CARD

CARD	--max-num-seqs	TOKENS, OFFLINE	…API SERVER
B200	1024	16,384	**8,192**
H100	1024	16,384	**8,192**
H200	1024	16,384	**8,192**
A100	256	8,192	**2,048**
L40S	256	8,192	**2,048**

Two tiers, not five cards: device memory, with one exception. Under 70 GiB you get the smaller default, and the A100 gets it too, by name. Most people are serving, so most are on the shaded column and do not know it. Both token columns double if you run `--performance-mode throughput` and have not set them yourself.

Which wall are you against?

Put `vllm:num_requests_running` against part one's ridge point for your card — **296** on an H100, **153** on an A100, **419** on an L40S — and watch `vllm:num_preemptions_total`.

CAPACITY Preemptions climbing under load you used to handle. The memory fence and the 8-bit cache are where the room is.

LATENCY Far under the ridge, no preemptions, and still missing your target. Capacity is fine, so the concurrency ceiling and the token budget are the pair to move.

NEITHER Far under the ridge, no preemptions, and goodput where you want it. The defaults are already the right answer for you.

WHAT TO MEASURE — ALL FOUR, TOGETHER

Time to first token
TTFT · prompt reading and queueing

Time per output token
TPOT · memory bandwidth and batch size

Throughput
the aggregate

Goodput
only requests that met your target — the only one that maps to money

Three of the four trade against each other, so a number quoted alone says nothing. A server at full throughput and 40% goodput is failing while looking busy.

HOW TO SWEEP IT YOURSELF

1. **Pin your request shapes first.** Prompt lengths, output lengths, and how much your prompts share a prefix. If you cannot state those three, that is the task, not the tuning.
2. **Then one setting at a time**, in this order: `--gpu-memory-utilization`, `--max-num-seqs`, `--max-num-batched-tokens`.
3. **Re-read the KV cache line** in the startup log afterwards: the token budget sets the activation peak, and that peak comes out of the fence you set first.
4. **Record p99 next to the median**, and read where a curve bends rather than its best point.
5. **Expect most of it to do nothing.** Write down what did not move, or you will sweep it again in six months.

Sources. The per-card numbers are `get_batch_defaults` in `vllm/engine/arg_utils.py`; the 0.92 fence is the field default in `vllm/config/cache.py`. Both read at tag `v0.27.1`, the current release at the time of writing. Nothing here is measured on my own hardware and nothing is estimated. From *Tuning vLLM: What Every Setting Does to the Arithmetic*, part five of *LLM Inference, Measured*, by Satsawat Natakarnkitkul · satsawat.ai