LLM inference API with batched GPU serving

front: what to ask, what to design · 55 minutes · one problem

1 REQUIREMENTS

0–8 min · ask, then write sentences

- **Ask:** fixed batch contract or continuous batching? May I autoscale, rate limit, cache, or is this the dispatch layer only?
- **Ask:** chat (TTFT matters) or offline? Client waits or polls? Scale today and at 10×? Which models, how many versions live?
- **A user can:** send a prompt and get tokens back, streaming or in one reply · choose a model and version, old ones included · paid and free tiers, paid first · one API across all replicas.
- **Technical challenges** (say this, not "non-functional"): GPUs near 100% busy · tail latency bounded, P95 `500 ms` · cold start of inactive models · isolation, no noisy neighbour · availability over consistency · metrics.
- **Anchors:** `100` and `10,000` req/s · batch ≤100 · `100 ms` per batch fixed · GPU boot is minutes · weights ≈ `1 TB` per frontier model.

2 CORE ENTITIES

8–12 min · nouns, then fields

- **Request**(id, tenant_id, model, version, prompt, max_tokens, stream, idempotency_key, state)
- **Model**(name) · **ModelVersion**(model, version, min_gpus, weights_ref, state active|inactive)
- **Replica**(id, model_version, gpu_ids, state warming|ready|draining, queue) — hardware running loaded weights
- **Tenant / Limit**(tenant_id, tier, rpm, tpm)
- **Batch** is a property of the replica's queue, not an entity. Isolation is a policy on the queues.
- Say aloud: model = blueprint (version + weights on storage); replica = GPUs with the weights loaded; each version needs a minimum GPU count.

3 API

12–17 min · one endpoint per requirement

- `POST /v1/messages` {model, version?, prompt, max_tokens, stream} → 200 body, or SSE token stream; `429` + retry_after when shedding
- `GET /v1/requests/{id}` → state, tokens so far (polling clients)
- `GET /v1/models` → versions, state, warm replica count
- Internal: `POST /replicas/{id}/load` {model_version} · `/drain` · `/warm`
- Headers: tenant from auth token, never the body; idempotency key on every POST.

4 DATA FLOW

17–22 min · walk one request through

- auth + tenant limit at the gateway → **router** picks the pool by model+version (no live replica → cold-start path) →
- **per-tenant queue** (paid first, fairness) → **per-replica batcher**: flush at N or at `50 ms`, whichever first → replica runs prefill then decode →
- finished sequences **leave the batch early** → response service streams tokens → metrics: queue depth, batch fill, TTFT, P99, GPU util.
- Record→batch map kept so a dead replica costs one retry, never a duplicate answer.

5 HIGH-LEVEL DESIGN

22–38 min · boxes, one layer per row

- Row 1: clients → API gateway → router → model registry.
- Row 2: queue-depth monitor · per-tenant priority queues · per-replica batcher · scheduler/autoscaler.
- Row 3: replicas (N GPUs + loaded weights + KV cache) per model version; response service on the left.
- Row 4: metrics · warm pool (hardware only) · weights store (NVMe/object, fast load).
- Solid = request path 1–6; dashed = control, cold start, failure. Colour by layer, legend in the corner.
- Capacity out loud: `100 ÷ 0.1 s = 1,000 req/s per GPU` → 10 GPUs floor at 10k → ~18 at 65% + spares. Budget: 50 batcher + 100 GPU + network, keep ~200 ms for one retry.

6 DEEP DIVES

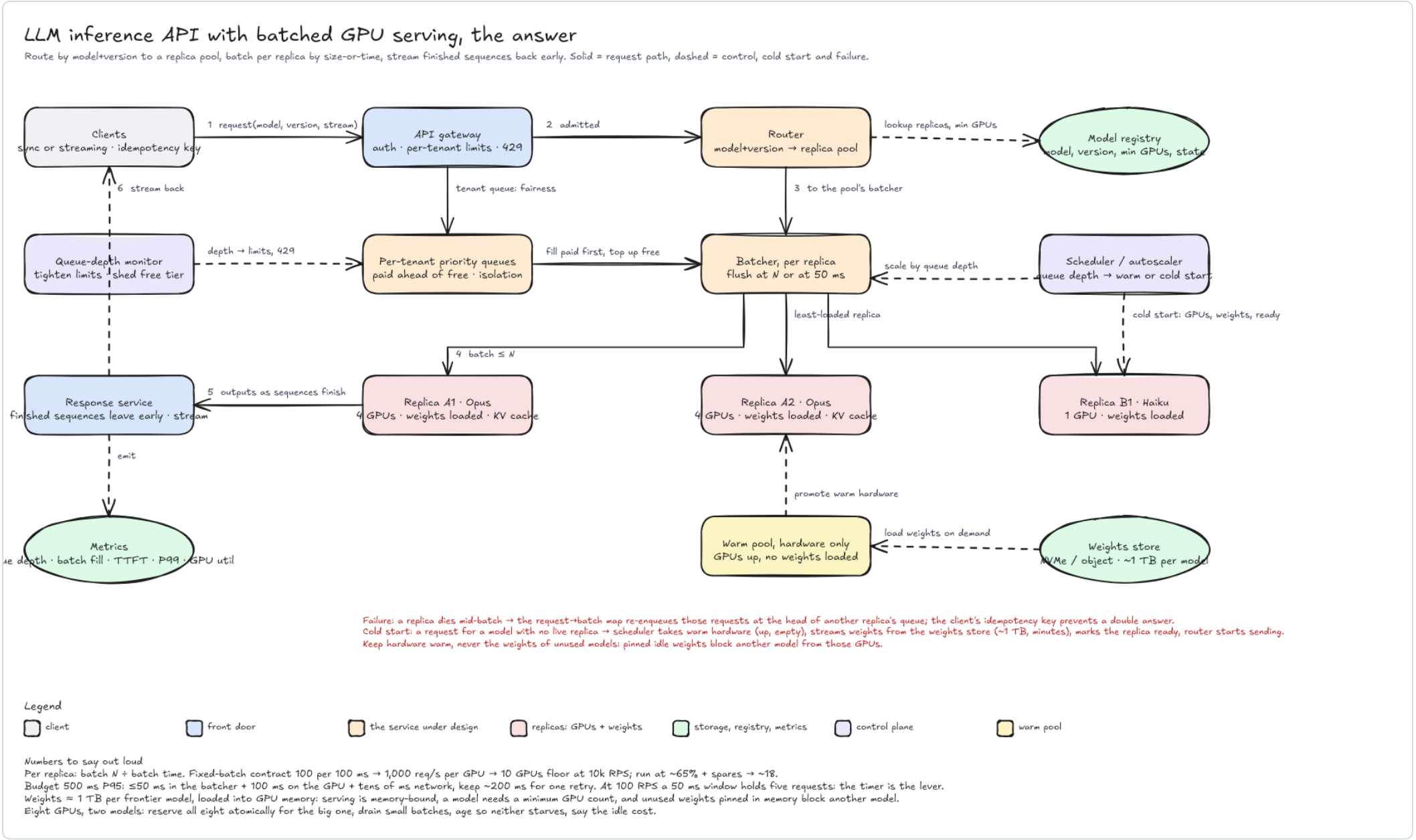
38–50 min · offer, then follow the interviewer

- **Batcher constants** from the budget: timer is the lever at low load, size at high load.
- **Cold start:** warm hardware, never idle weights pinned; scheduler assigns GPUs, streams weights, marks ready; router starts sending. Minutes, so plan for it.
- **Overload:** queue depth is the truth; tighten per-tier limits, shed free, 429 + retry_after. Not a static token bucket.
- **Isolation:** per-tenant queues, quotas, max tokens per request; one giant request cannot stall the batch.
- **Replica dies mid-batch:** map re-enqueues at head; idempotency key.
- **8 GPUs, two models:** reserve all 8 atomically, drain small batches, age so neither starves, say the idle cost.
- **Cache:** only if the hit rate pays; prefix KV cache for shared system prompts is the one that usually does.
- Bow at 50: decisions, what you would revisit, what you cut. Stop.

Sources: 1point3acres SD Q1 card and Aug–Sep 2026 reports · coach feedback 26 Sep 2026 · rik-kisnah.ai/teach/gpu-ai/design-an-inference-api

The board at minute forty

back: the answer, as it should look on the shared screen



Interactive: link.excalidraw.com/readonly/ZFowRfbEdBwmqn3Nhpa6 · source: 2026-09-26-inference-api-answer-board.excalidraw

back