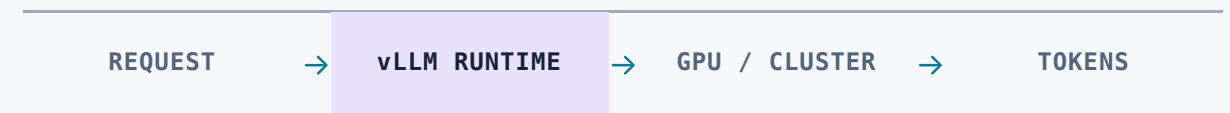


vLLM 101

Architecture, Challenges, and Scaling

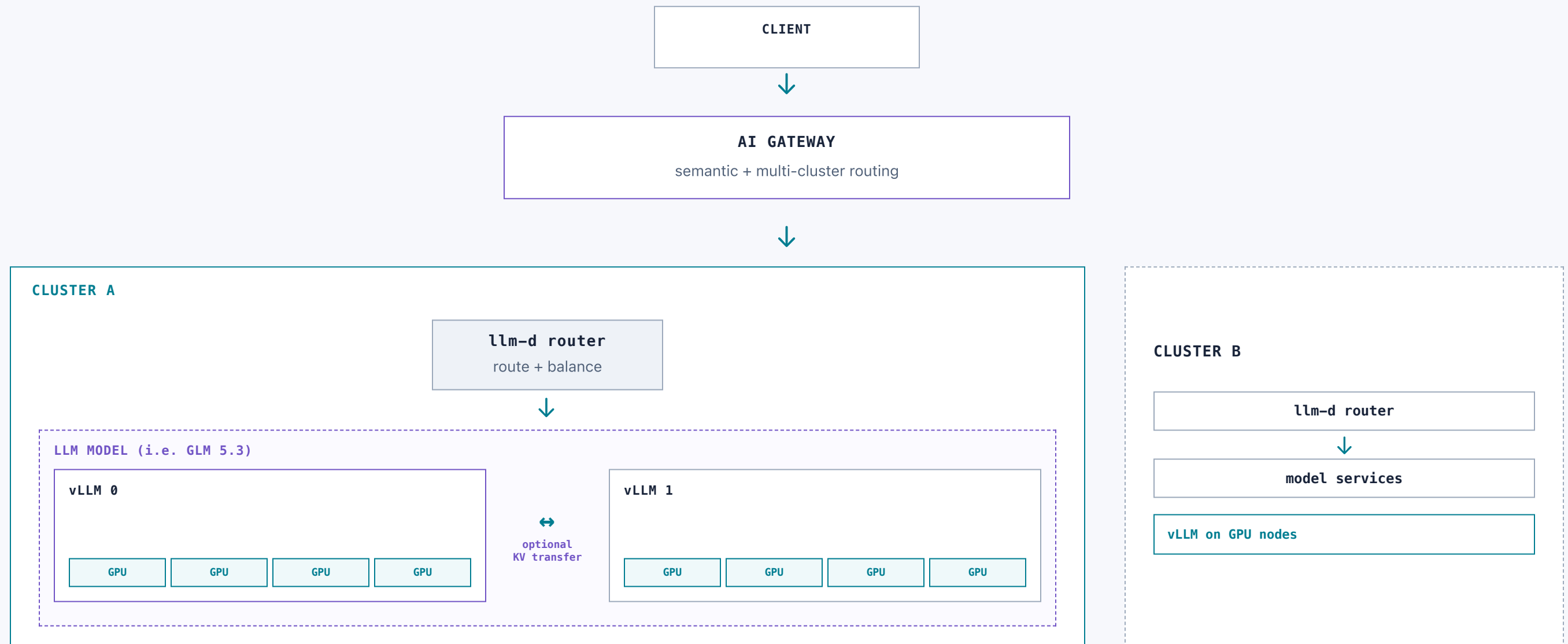
Daniele Zonca · Distinguished Engineer & OpenShift AI Chief Architect



We are going to follow one request—and see why serving it gets complicated fast.

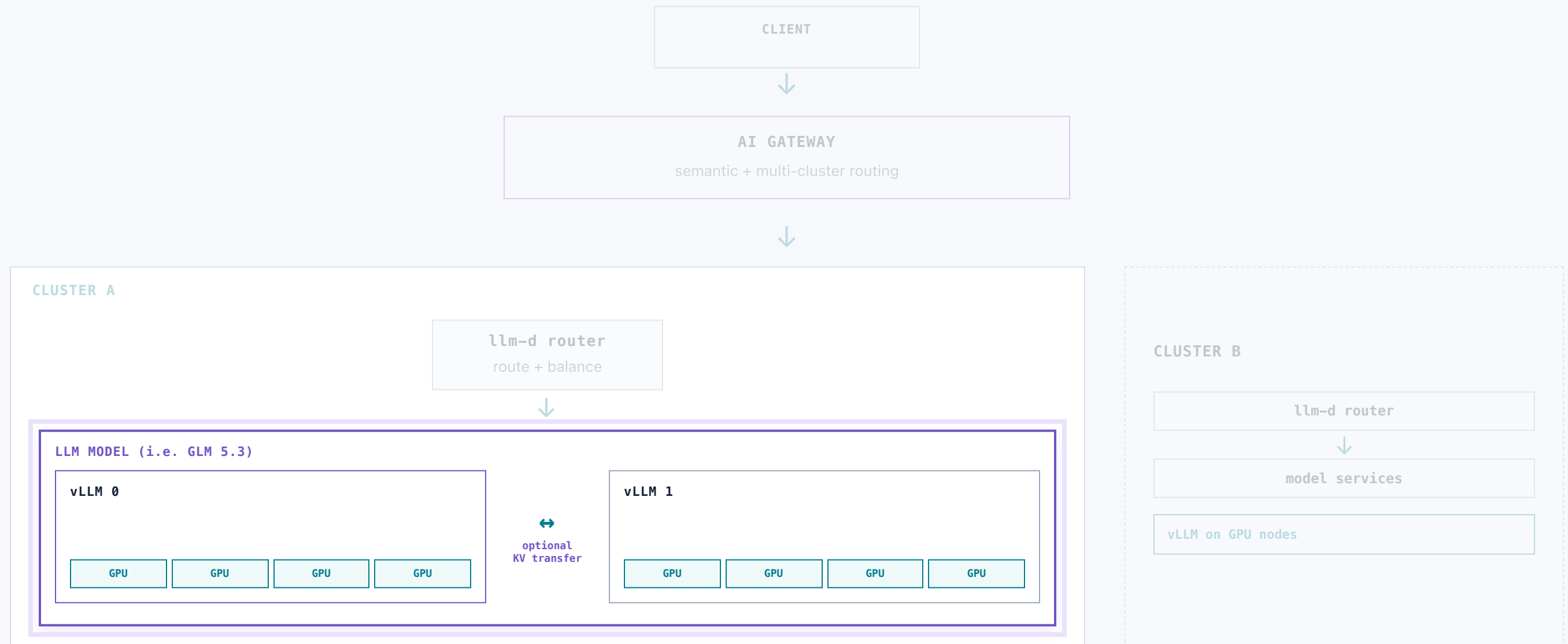
The serving system.

We will zoom from GPU execution to vLLM, model services, and global routing.



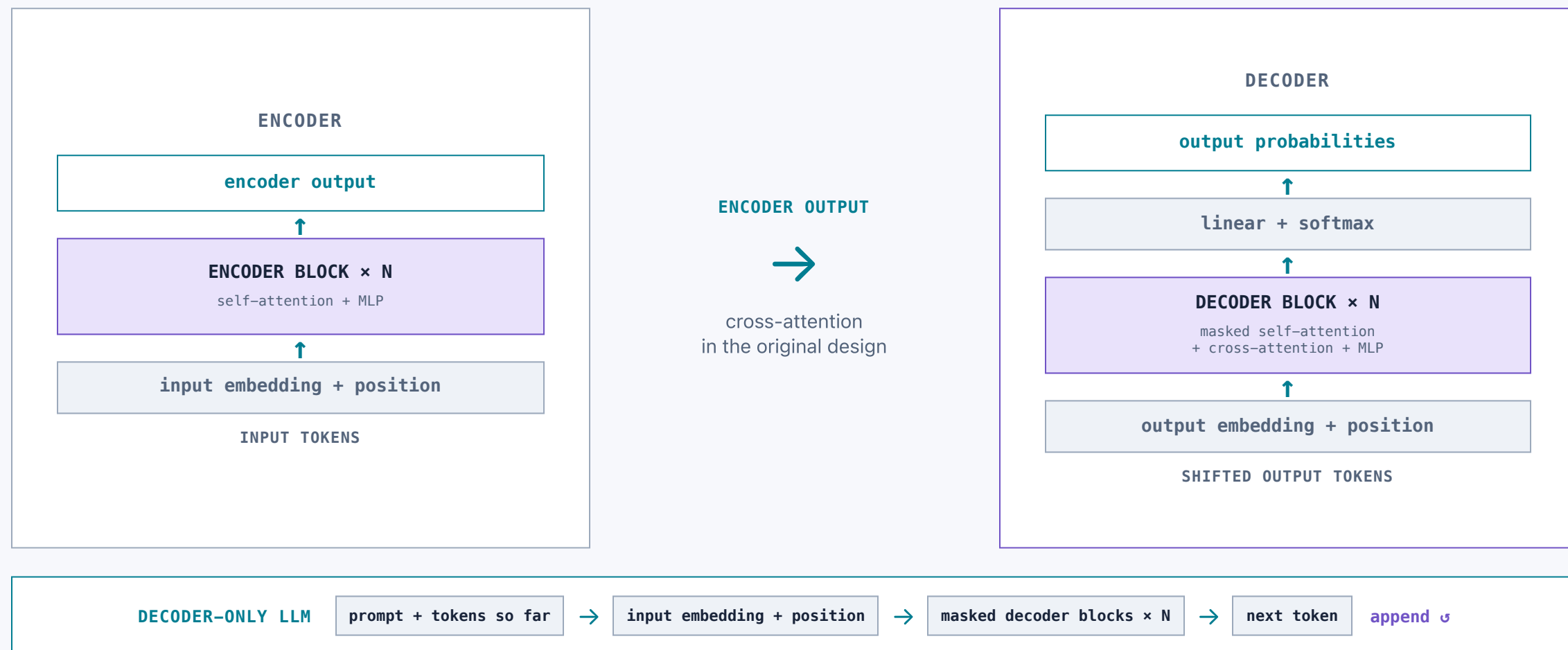
Start inside the model.

We begin with the LLM model—then follow its state outward.



Transformer architecture.

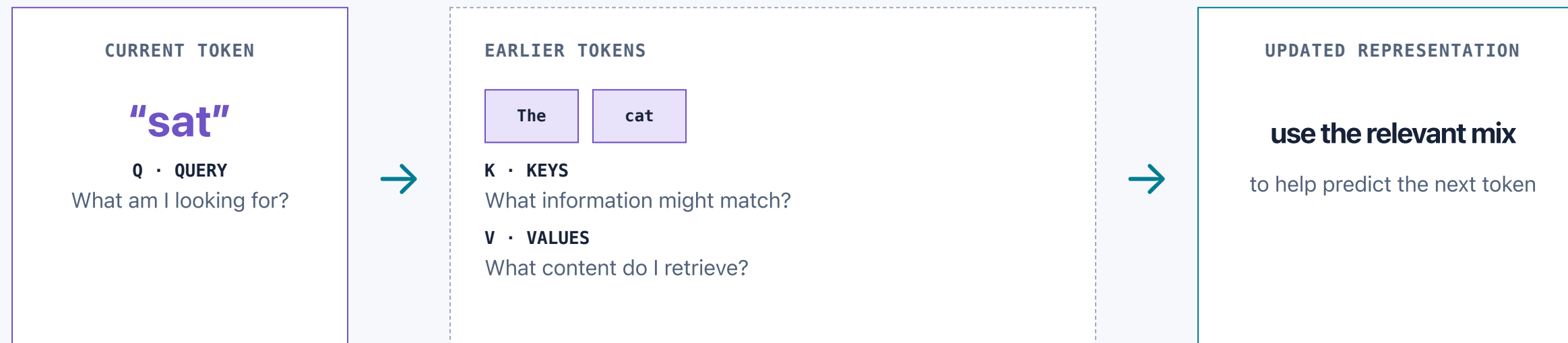
The original encoder–decoder design; decoder-only LLMs use the right-hand generation path.



HOW DECODER-ONLY CONNECTS Both designs generate with an autoregressive decoder. Decoder-only models put the prompt and generated tokens so far into one sequence, then feed it directly to masked self-attention—no encoder output or cross-attention.

Attention lookup.

"Attention Is All You Need"



$$\text{ATTENTION}(Q, K, V) = \text{softmax}(QK^T / \sqrt{d_k}) V \quad \text{scores} \times \text{values}$$

Q / K / V IN PLAIN ENGLISH The current token makes a query (Q). It scores earlier keys (K), then combines their values (V). K and V are the historical state that serving needs again.

Three optimization layers.

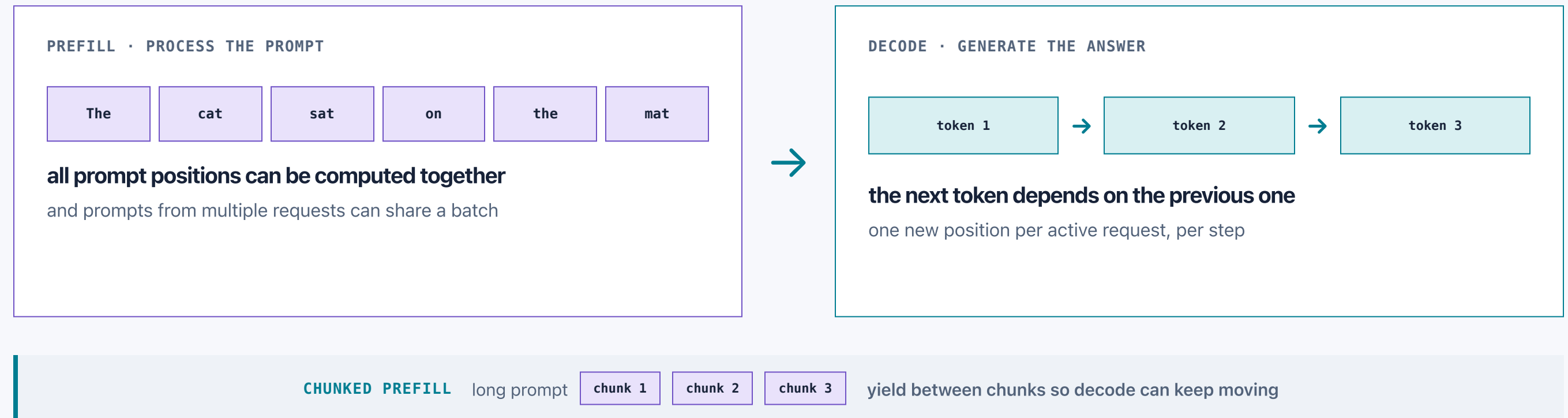
Architecture defines state. Kernels compute attention. Runtimes manage state across requests.

LAYER	EXAMPLES	MAIN QUESTION
MODEL ARCHITECTURE	MHA · GQA/MQA · MLA · Mamba · MoE	What state and compute path does the model define?
ATTENTION KERNEL	FlashAttention · FlashInfer	How is attention computed efficiently on a GPU?
SERVING SYSTEM	vLLM / PagedAttention + prefix caching SGLang / RadixAttention	How is state placed, reused, scheduled, and evicted across requests?

MODEL CONFIGURATION vLLM reads the Hugging Face model config—normally `config.json`—to identify a supported architecture. Use `--hf-overrides` or explicit serving flags to override configuration choices.

Prefill vs. decode.

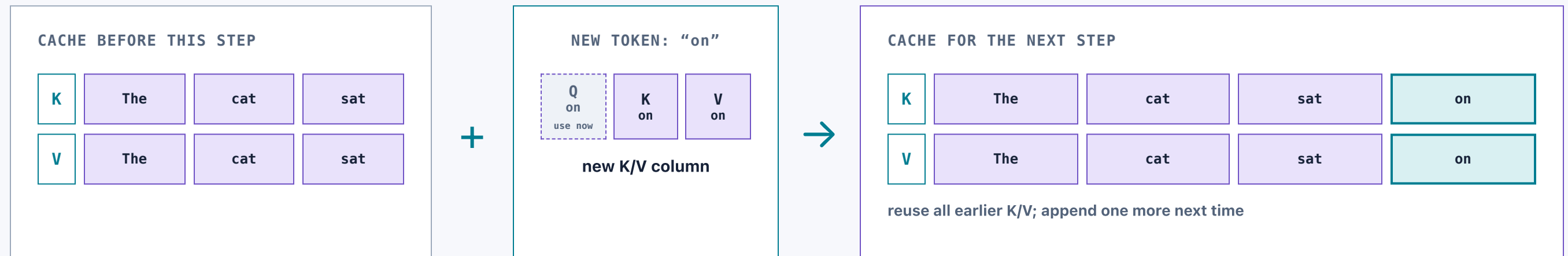
Parallel prompt processing; sequential token generation.



VLLM DEFAULT In vLLM V1, chunked prefill is enabled by default. `--max-num-batched-tokens` sets the token budget that controls how prompt work is chunked and interleaved.

The KV cache.

Save each token's K/V once, then reuse it.



VLLM DEFAULT Prefix caching is enabled by default in vLLM V1: complete matching prefix blocks can be reused across requests. `--kv-cache-dtype` trades KV precision for memory footprint.

From one request to many concurrent requests.

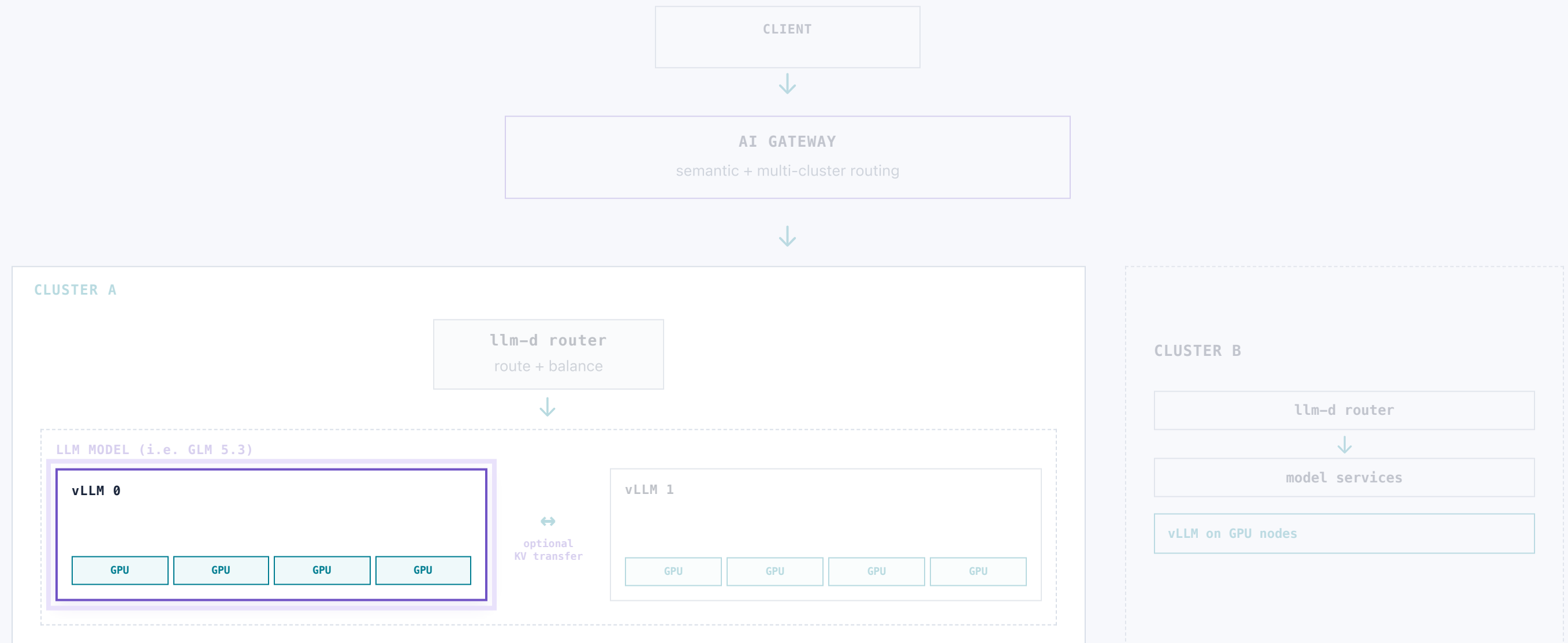
Many growing decode loops share one runtime.



VLLM SCHEDULING KNOBS `--max-num-batched-tokens` and `--max-num-seqs` bound each step; `--max-model-len` caps context. `--enforce-eager` disables CUDA Graphs—it is not a scheduler policy.

Zoom into vLLM.

The runtime turns live requests into GPU work.



Start a server.

One command exposes the online serving API.

```
$ vllm serve <model>
```



OPENAI-COMPATIBLE SERVER

including supported Anthropic-compatible Messages API paths

ACCEPT HTTP REQUESTS

STREAM OUTPUT TOKENS

RUN THE vLLM RUNTIME

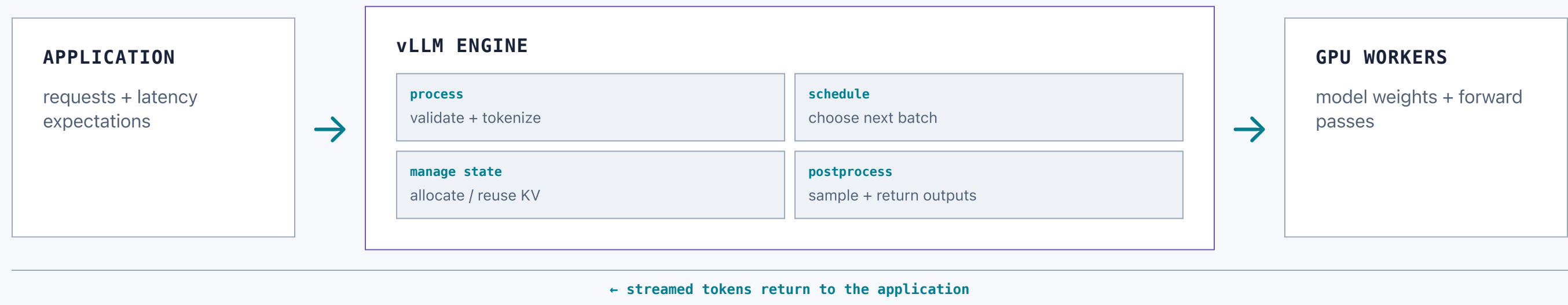
OR USE PYTHON

For offline or embedded
inference:

```
from vllm import LLM  
llm = LLM(model="Qwen/Qwen2.5-7B-Instruct")
```

What vLLM does.

It turns many live requests into efficient GPU work.



PLAIN-LANGUAGE DEFINITION vLLM is an open-source inference and serving runtime. It processes requests, schedules feasible batches, manages live state, executes the model on GPUs, then postprocesses and returns outputs.

PagedAttention blocks.

Avoid one large contiguous cache reservation per request.

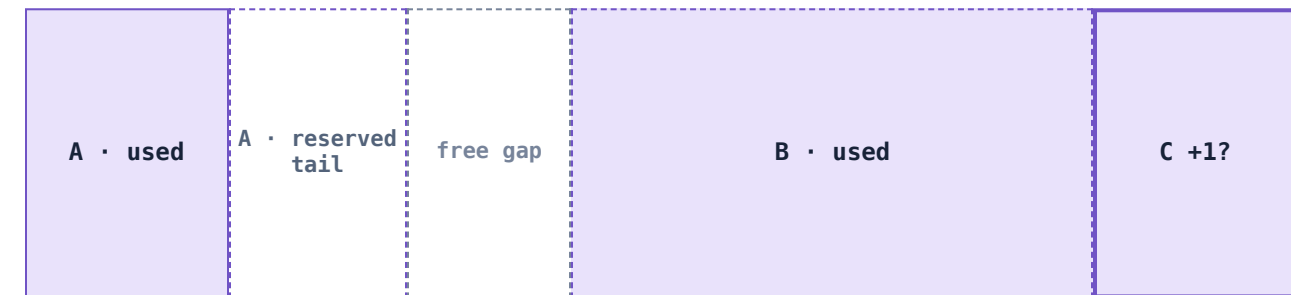
LIVE K/V CACHES – GROWING AT DIFFERENT RATES



must fit



ONE CONTIGUOUS REGION PER REQUEST

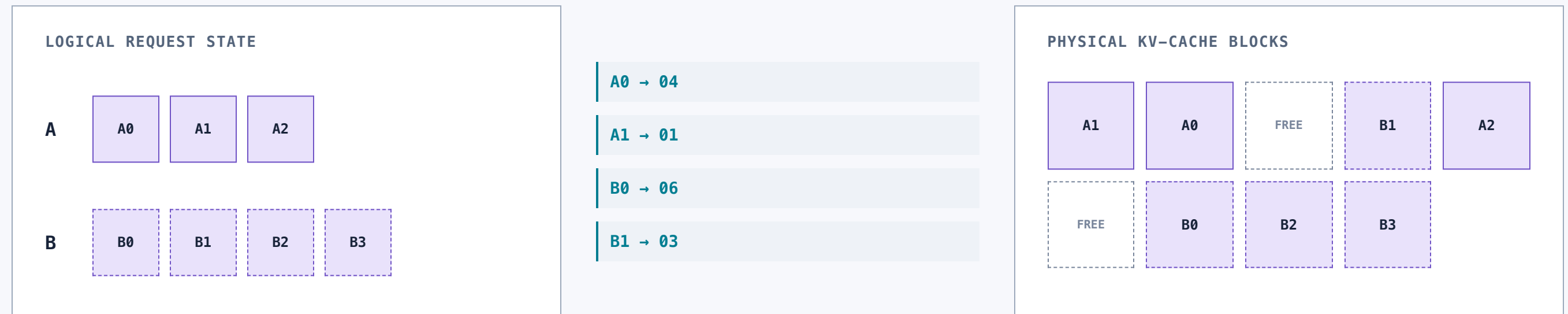


unused reservation is internal waste; small free gaps can be unsuitable for the next contiguous growth

PAGEDATTENTION MOVE Split a request's K/V state into fixed-size blocks. Allocate another block only when needed, rather than reserving one large contiguous region up front.

Block mapping.

Logical request state can use any free physical blocks.



WHAT THIS ENABLES When a request grows, vLLM can attach another available physical block. When it finishes, its blocks return to the pool for other requests.

Inside GPU memory.

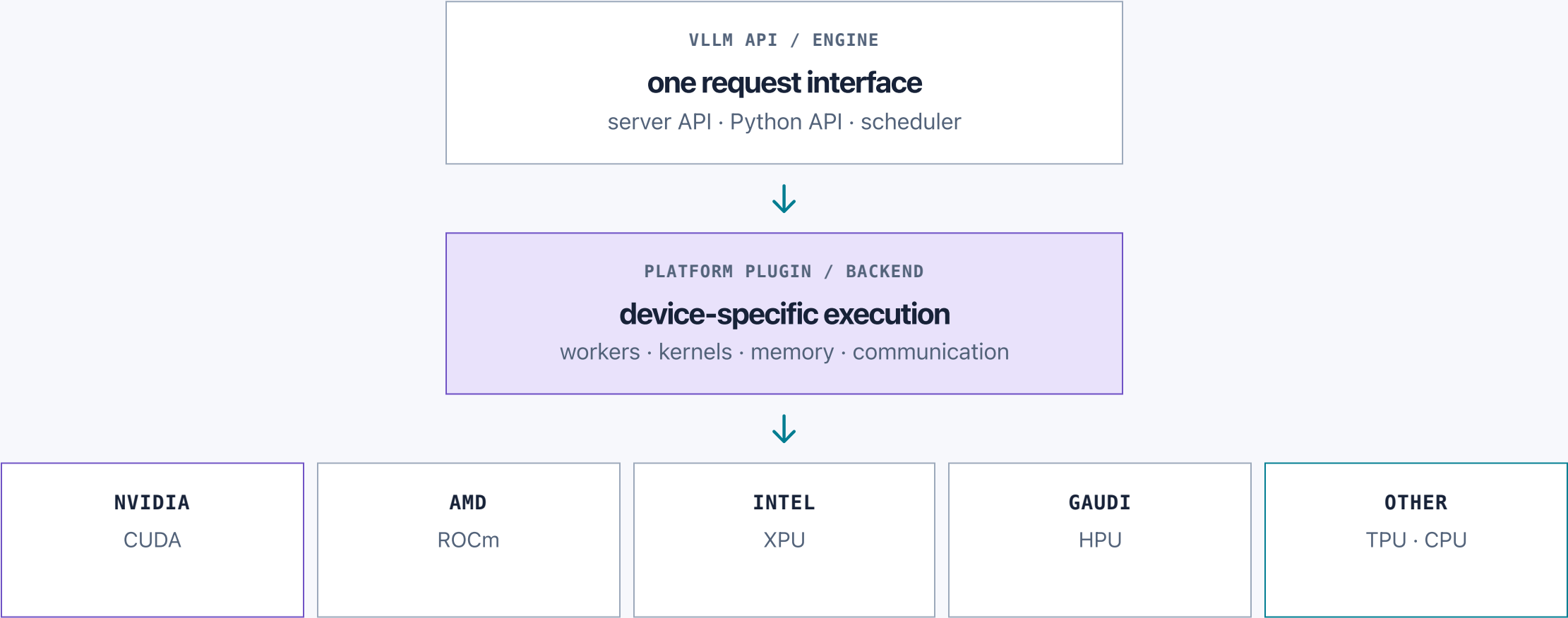
Three things share one finite pool.



VLLM GPU MEMORY `--gpu-memory-utilization` sets vLLM's target GPU-memory fraction; after weights and runtime overhead, profiling determines the KV-block pool. `--kv-cache-memory-bytes` can set it explicitly.

One API, many devices.

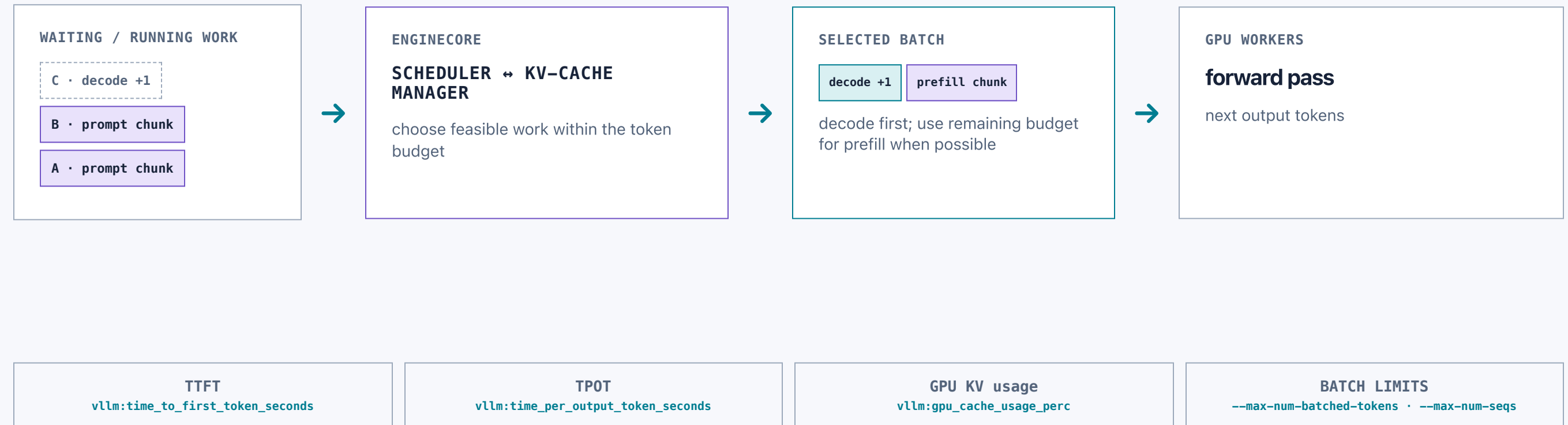
vLLM selects a platform backend for the accelerator it finds.



SAME API, DIFFERENT CAPABILITIES Platform support is build- and device-specific. Check the available kernels, quantization, parallelism, metrics, and tuning options on the target accelerator.

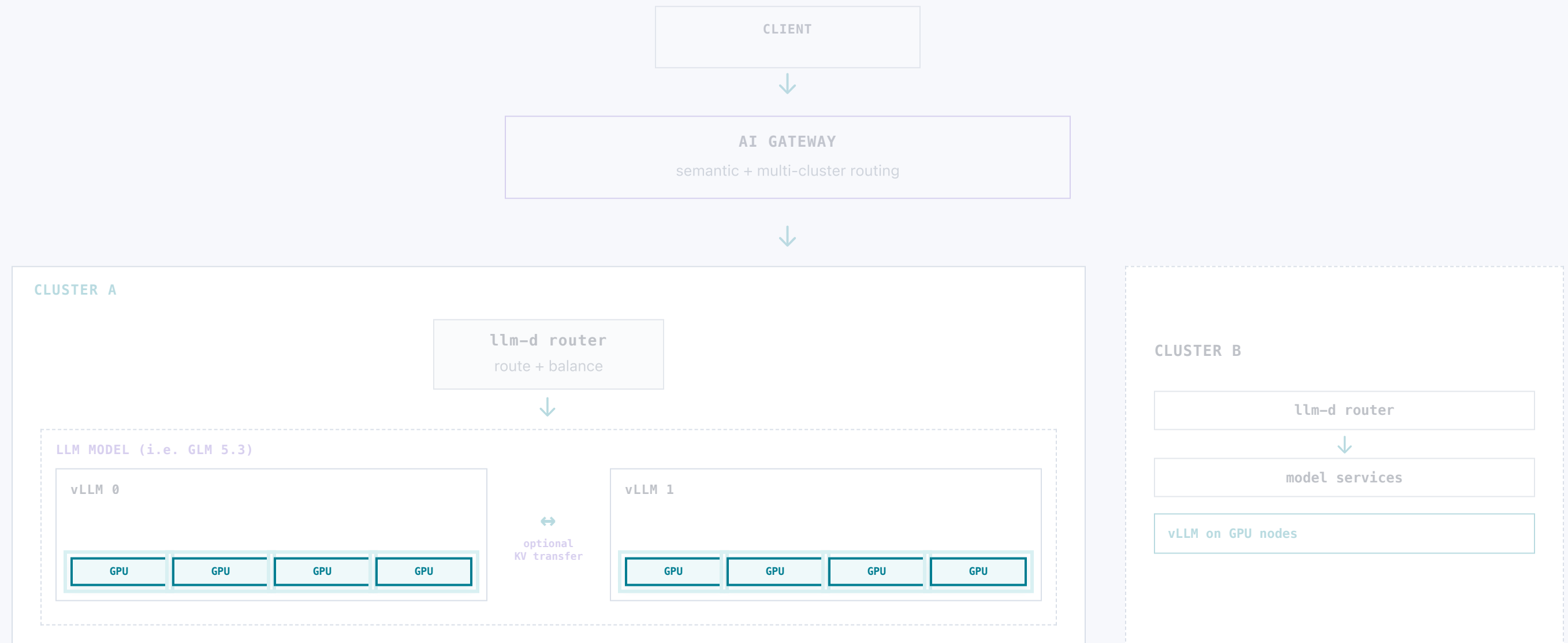
Request scheduling.

KV-cache headroom constrains the next feasible batch.



Scale across GPUs.

Now follow model execution from GPU workers to multi-GPU patterns.



How should I scale?

Choose the bottleneck—not just GPU count or KV-cache space for context.

WHEN

One replica needs more GPU memory

Tensor parallelism

Shard every layer across GPUs.

BEST: NVLink / NVSwitch
WATCH: collectives every layer

WHEN

Model must span nodes

Pipeline parallelism

Place layer groups into stages—often one node each.

BEST: TP within node + PP across nodes
WATCH: stage balance + activation hops

WHEN

Model fits, but traffic does not

Data parallelism

Replicate the whole serving engine.

BEST: independent requests
WATCH: separate KV caches

WHEN

The model has MoE experts

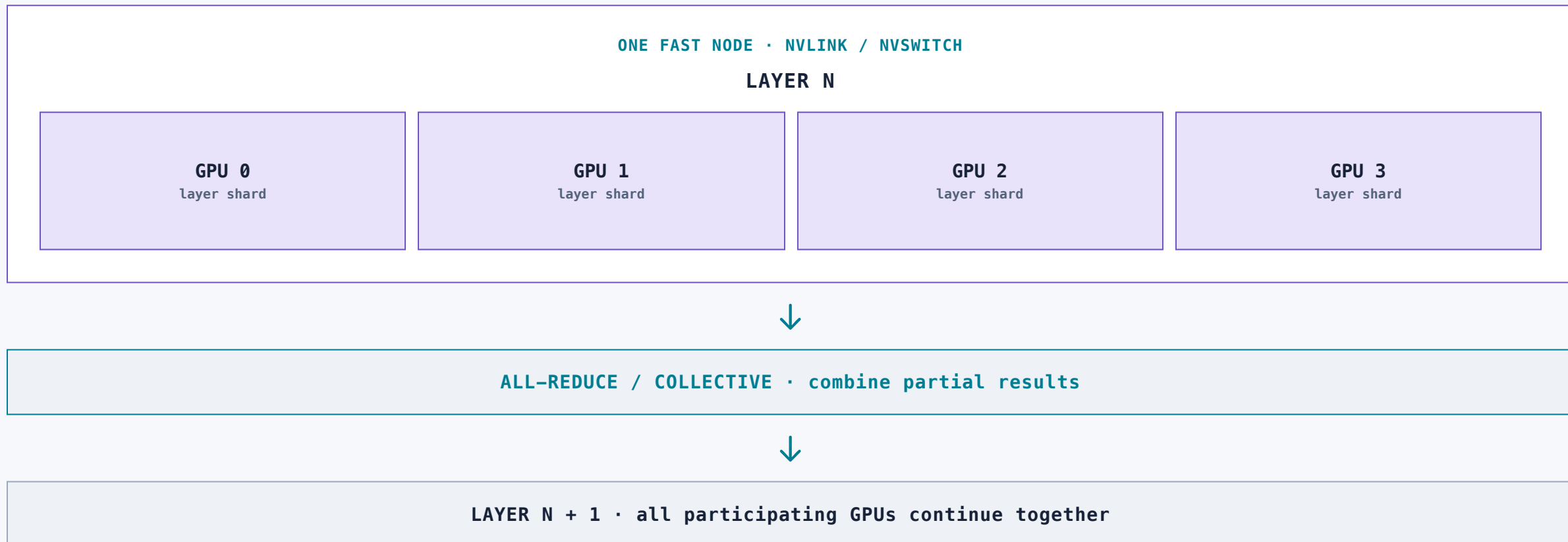
Expert parallelism

Place experts across GPUs; route tokens to them.

ONLY FOR MoE
WATCH: token all-to-all

Tensor parallelism.

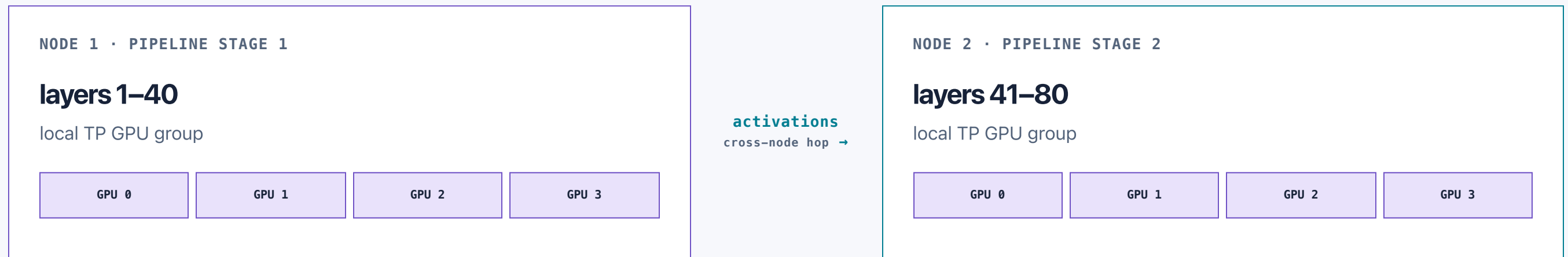
Shard every layer across a fast GPU group in one node.



USE TP INSIDE A FAST NODE It makes a dense model fit across local GPUs, but every layer requires a collective. Repeated cross-node collectives are usually expensive.

Pipeline parallelism.

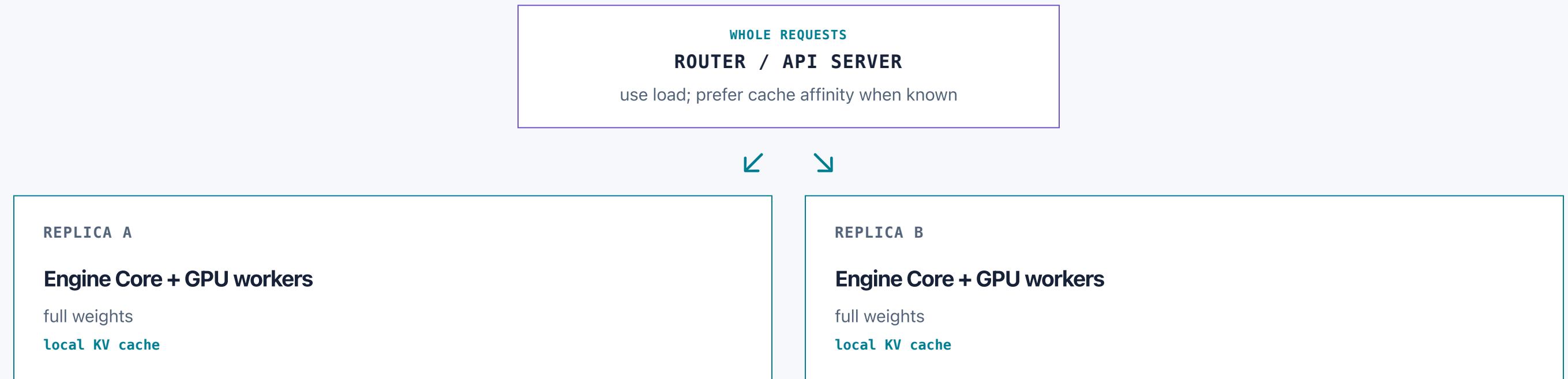
Split model depth across nodes; keep tensor parallelism local to each stage.



USE PP TO SPAN NODES A stage owns a range of layers; only activations cross between stages. Balance the stages, and use TP inside each node when needed.

Data parallelism.

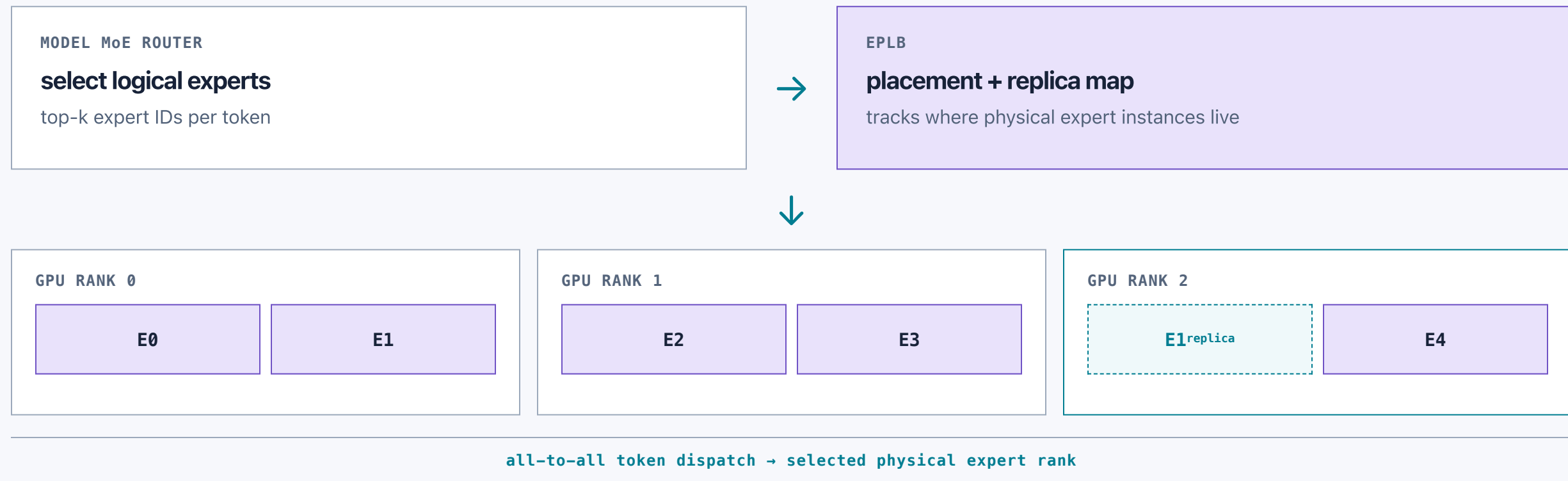
The model fits; add replicas when independent request demand grows.



USE DP FOR CONCURRENCY Each replica owns a local cache. A cache-aware router can prefer a matching prefix or session; otherwise, load-aware routing chooses a feasible replica.

Expert parallelism.

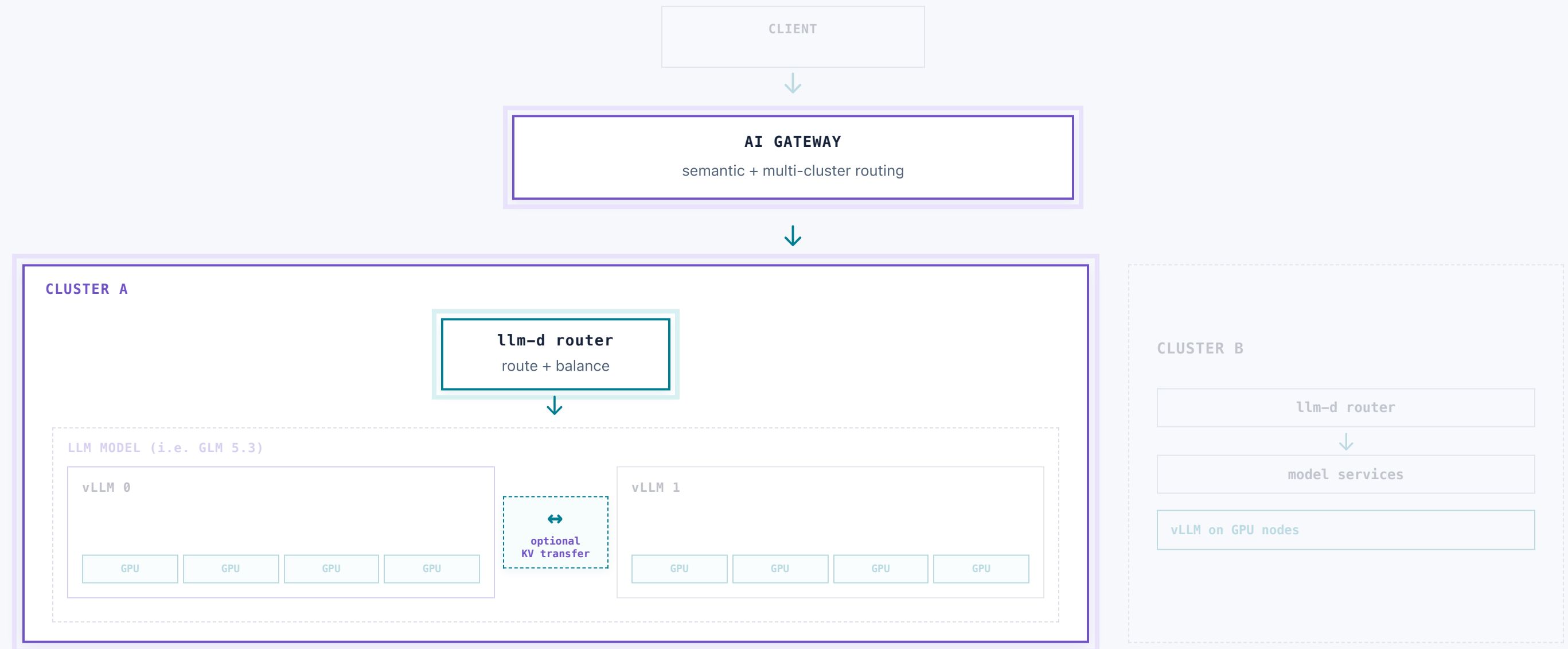
For MoE models: distribute experts, then balance where their work runs.



MODEL ROUTING ≠ **EPLB** The MoE router chooses logical experts. EPLB balances physical expert placement and optional replicas so the runtime can send work to where those experts live.

Global and cluster routing.

The AI Gateway selects a cluster; llm-d routes within the selected cluster.



Scale further.

Route, move work, or tier state only for a measured bottleneck.

INTELLIGENT ROUTING · `llm-d`

prefer a warm cache

route to a replica or cluster with a matching prefix or session

```
llm-d · cache-aware affinity
```

DISAGGREGATE PREFILL / DECODE

separate the two workloads

transfer K/V state between dedicated instances

```
prefill → KV connector → decode
```

KV CACHE CONNECTOR

extend the cache boundary

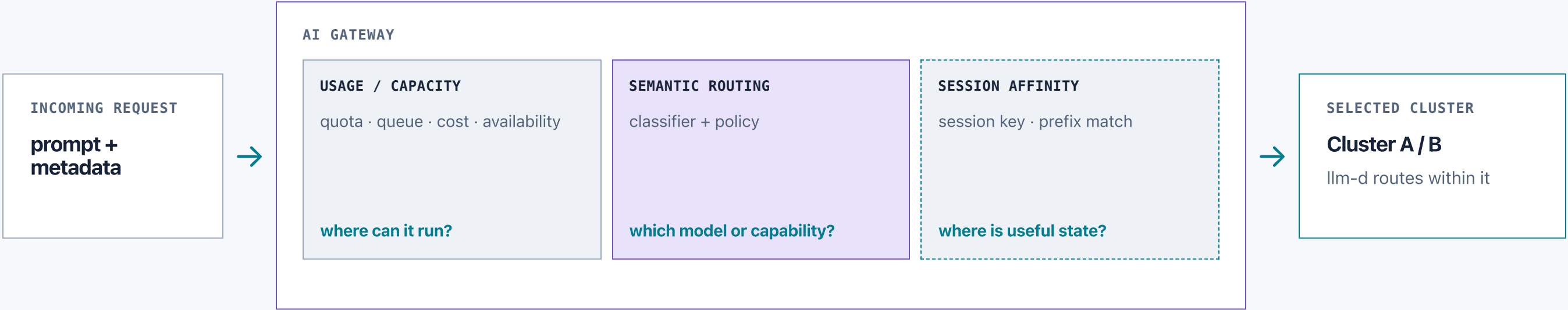
offload, transfer, or share state through a connector and remote tier

```
--kv-transfer-config
```

NOT FREE SPEEDUPS Prefer cache reuse through routing before moving state. Disaggregation and cache tiers add control, but also transfers, lookups, and operational failure modes.

Global routing policy.

Choose a cluster and model using load, intent, and session affinity.

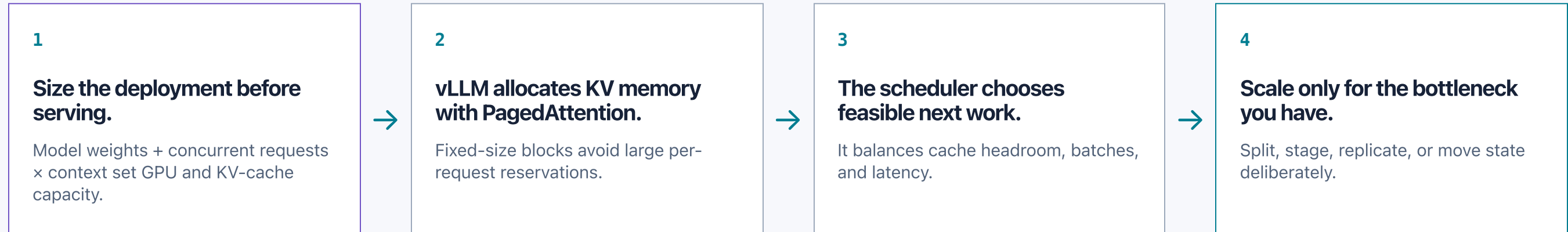


Affinity can improve reuse; availability, failover, and SLO policy can override it.

SEPARATE THE DECISIONS The gateway makes global placement and model-selection decisions. llm-d makes the next routing decision inside the chosen cluster.

Sizing vLLM.

Weights, KV state, scheduling, and topology determine serving capacity.



Why context matters: longer prompts make prefill compute $O(L^2)$; active context makes KV memory and per-token decode work grow $O(L)$. Size weights, concurrency, and context before choosing a topology.

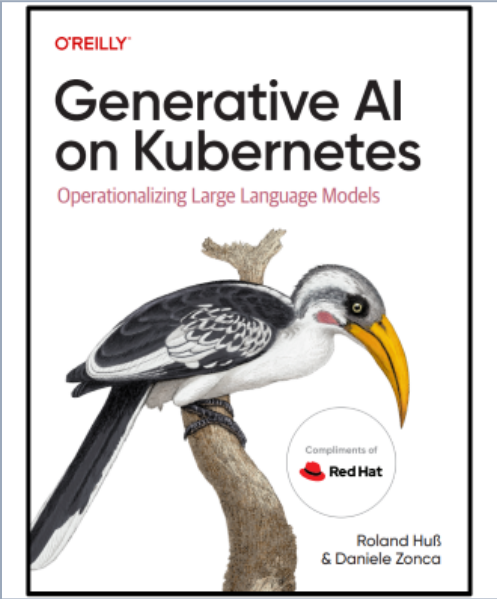
Thanks.

If you want to learn more,
[here are two resources.](#)

FREE DOWNLOAD AFTER REGISTRATION

Generative AI on Kubernetes

O'Reilly · Roland Huß & Daniele Zonca



red.ht/generative-ai-on-kubernetes

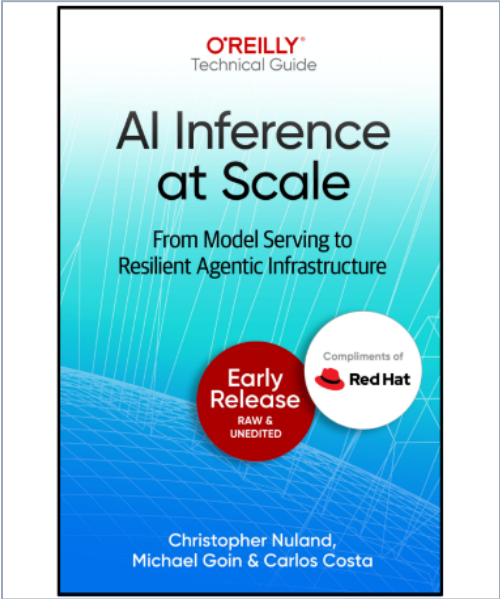


Or get it on O'Reilly Learning, Amazon, and more.

EARLY RELEASE · TECHNICAL GUIDE

AI Inference at Scale

O'Reilly · Christopher Nuland · Michael Goin · Carlos Costa



red.ht/ai-inference-at-scale

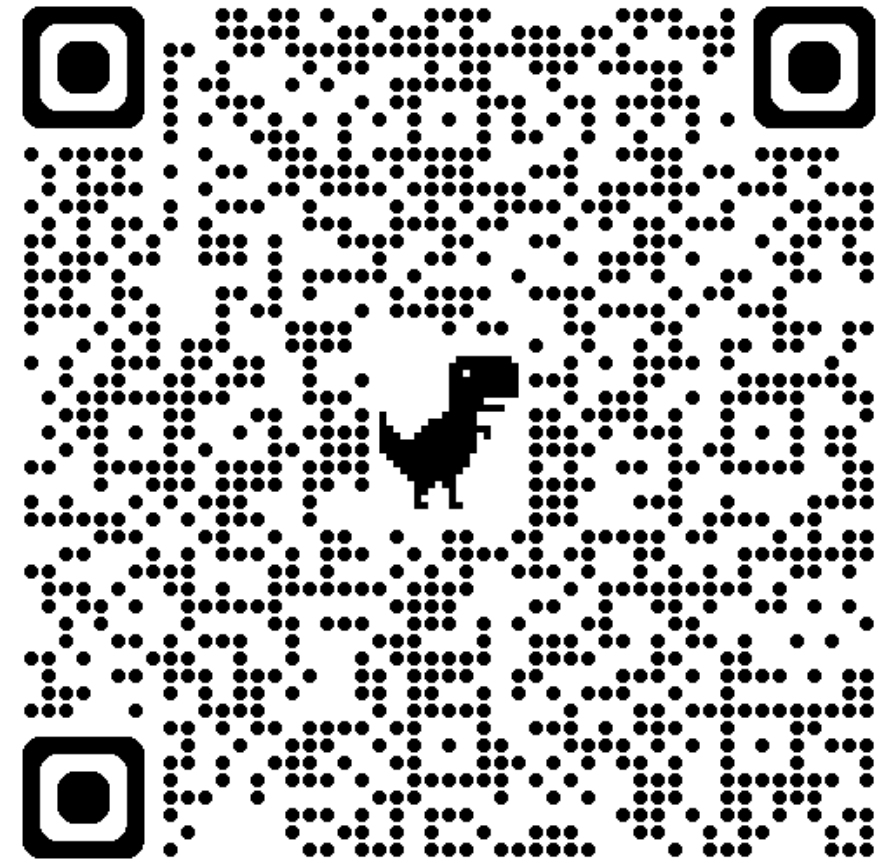
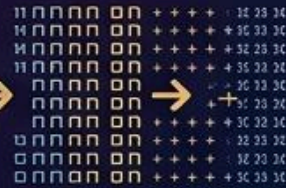
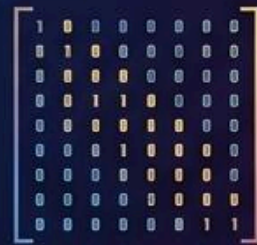
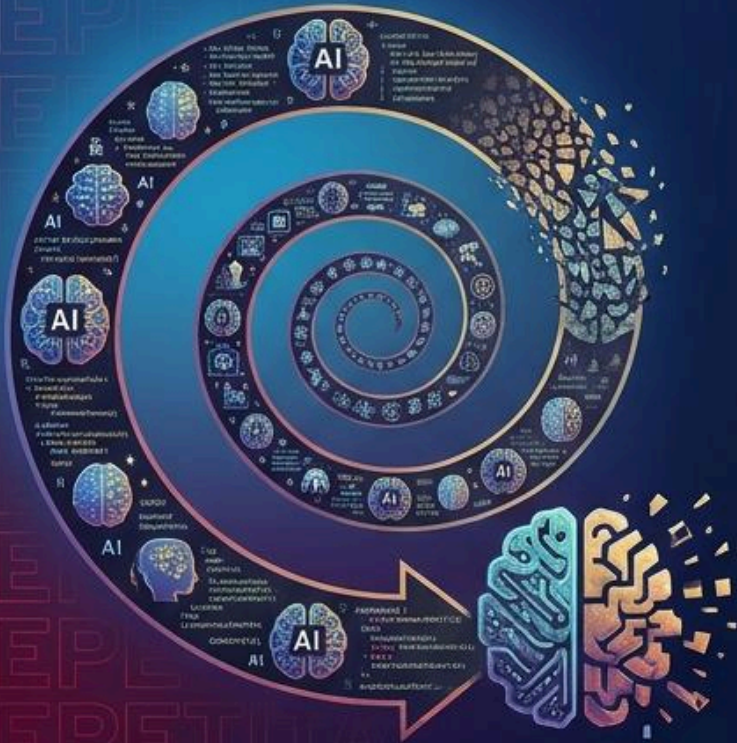


Early access release.

Papers We Love **Milano**

Repetita Non Iuvant: Why Generative AI Models Cannot Feed Themselves

VALERIA ZUCCOLI EXPLAINS
AI MODEL COLLAPSE



OCTOBER 14TH | 18:30

RED HAT ITALY | VIA FARA 26 (7th Floor)

<https://bit.ly/pwlmi-repetita>