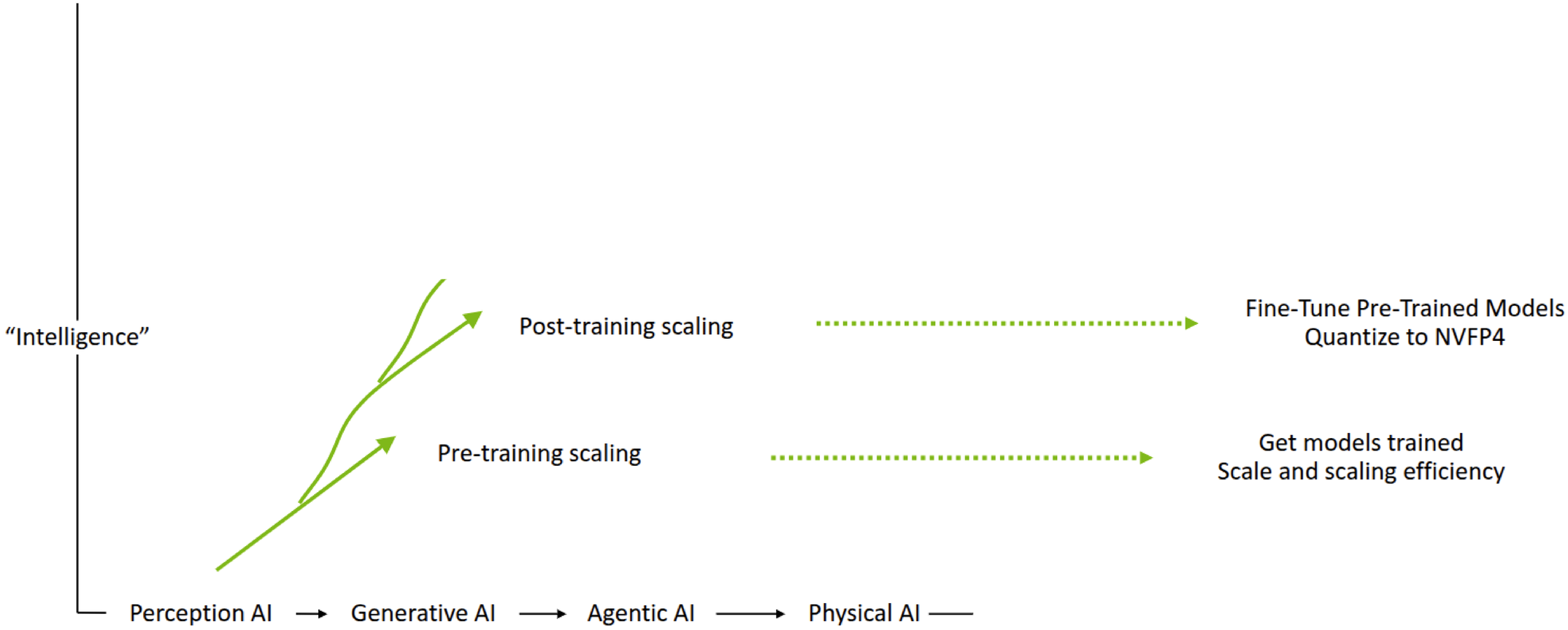
Scaling LLM Inference on HPC/AI Systems

Dr. Séverine Habert | Senior Solutions Architect |
NVIDIA

AI Scaling Laws Drive Exponential Demand for Compute

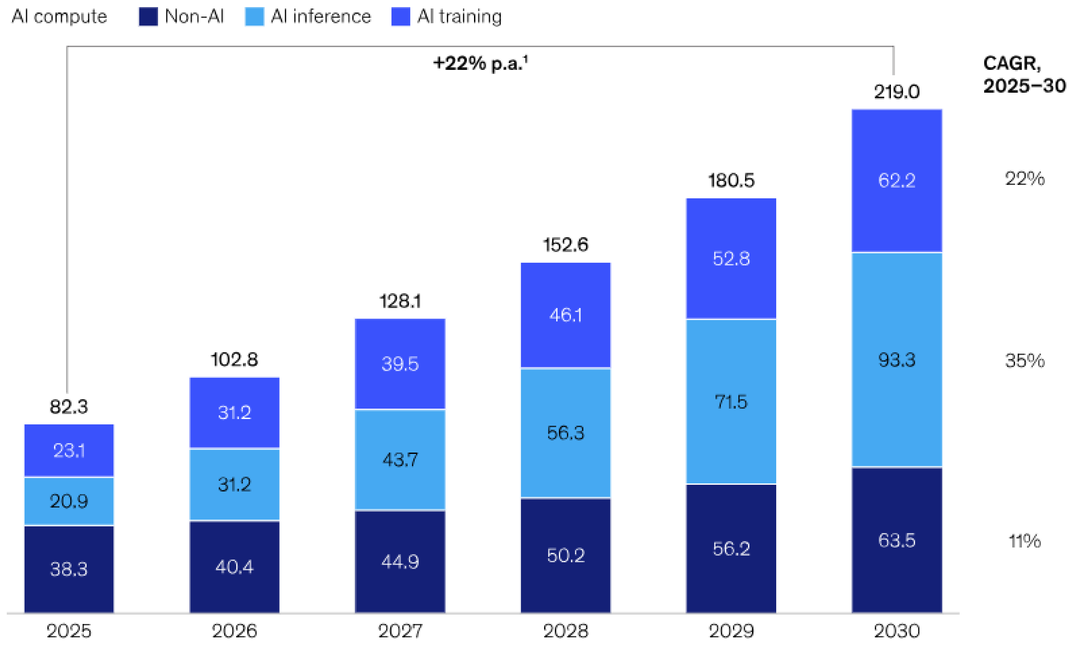


Demand for AI inference

Will overpass AI training by next year at the hyperscalers

Inference workloads could make up more than 40 percent of data center demand in 2030, growing 35 percent CAGR until 2030.

Global data center demand by workload, 2025–30, gigawatts



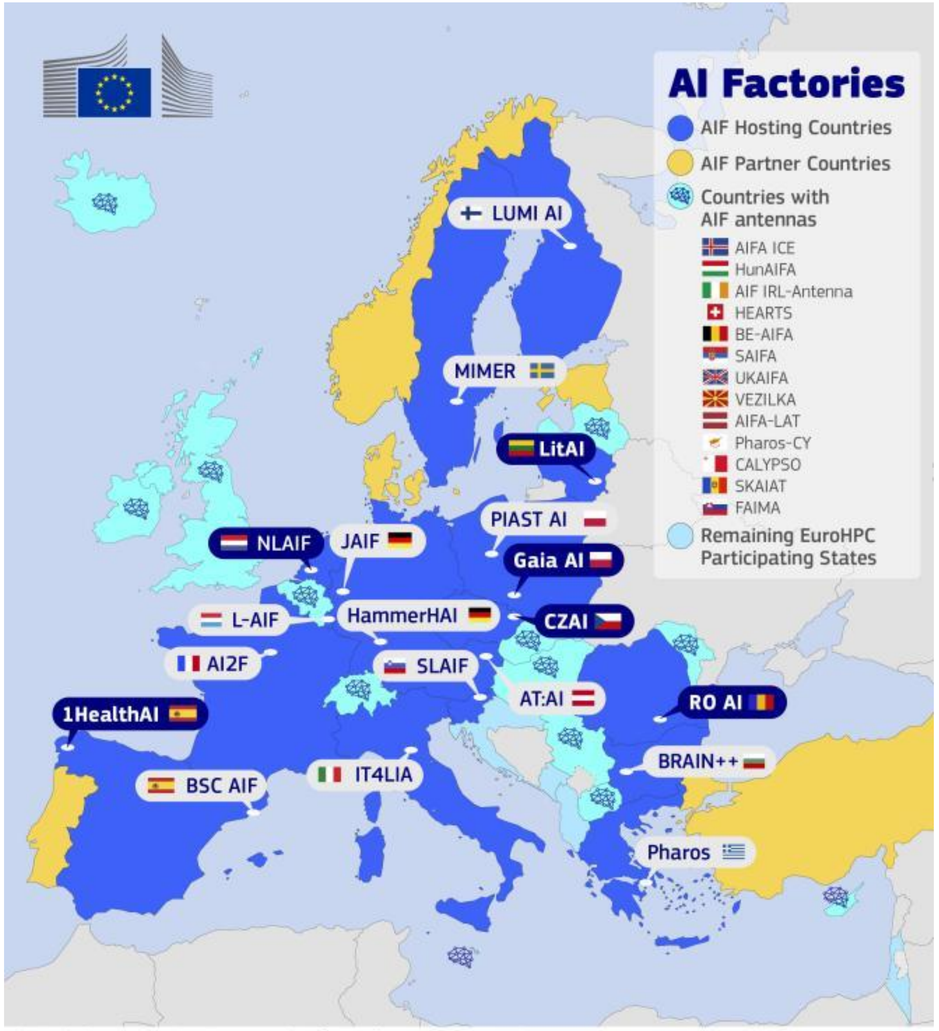
Note: Includes all provider types.
[1]Per annum.
Source: McKinsey Data Center Demand Model

McKinsey & Company

https://www.mckinsey.com/industries/technology-media-and-telecommunications/our-insights/the-next-big-shifts-in-ai-workloads-and-hyperscaler-strategies

What about supercomputing centers in Europe?

AI Factories and Gigafactories



Switzerland's participation is contingent upon the ratification of its accession to Horizon Europe.
The boundaries and names shown and the designations used on this map do not imply official endorsement or acceptance by the European Union.
This designation shall not be construed as recognition of a State of Palestine and is without prejudice to the individual positions of the Member States on this issue.
Administrative boundaries: © EuroGeographics © OpenStreetMap
Cartography: Eurostat – IMAGE, 05/2025

New AI-optimized supercomputer for a growing HammerHAI service portfolio

The AI Factory HammerHAI opened in April 2025, and has already begun offering AI computing and services to European startups and SMEs on existing infrastructure. This includes providing:

- Test access to existing, AI-optimized computing systems currently hosted by HammerHAI consortium partners
- End-to-end support by HammerHAI AI experts, including guidance on relevant tools, services, and expertise
- Access to inference services for large language models
- Professional training courses for AI skills development
- Consulting and AI-adoption support, including AI capability assessments and guidance on AI ethics and risk management

Article 1

Regulation (EU) 2021/1173 is amended as follows:

(1) Article 2 is amended as follows:

(a) the following points are inserted:

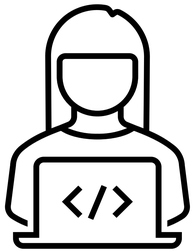
'(3c)"Artificial Intelligence gigafactory" or "AI gigafactory" means a state-of-the-art large-scale facility with sufficient capacity to handle the complete lifecycle, from development to large-scale inference, of very large AI models and applications, providing a supercomputing service infrastructure which is composed of AI-optimised computing capacity, a supporting data centre infrastructure including high-capacity storage and networking, dedicated secure cloud user access environments, and specialised secure AI-oriented support services for its advanced operations, all of which are supported by an environmentally sustainable infrastructure, in particular for energy and water supply systems;

Inference workloads

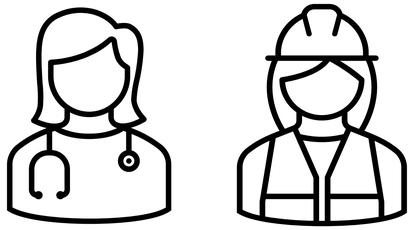
- Sovereign AI services: built, hosted and deployed in Europe

Scientific AI workloads

LLM for chatbot / agents



Industry-specific AI workloads: healthcare, manufacturing, ...



Inference Compute Requirements Scaling Exponentially

Fueled by reasoning models and AI agents

Larger Models

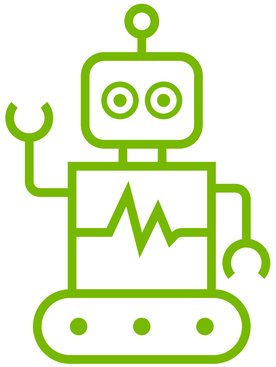
Hundreds of billions of parameters

Long Thinking Time

100x more thinking tokens

Larger Context

Millions of input tokens



Agents

One user prompt involves multiple model executions

“

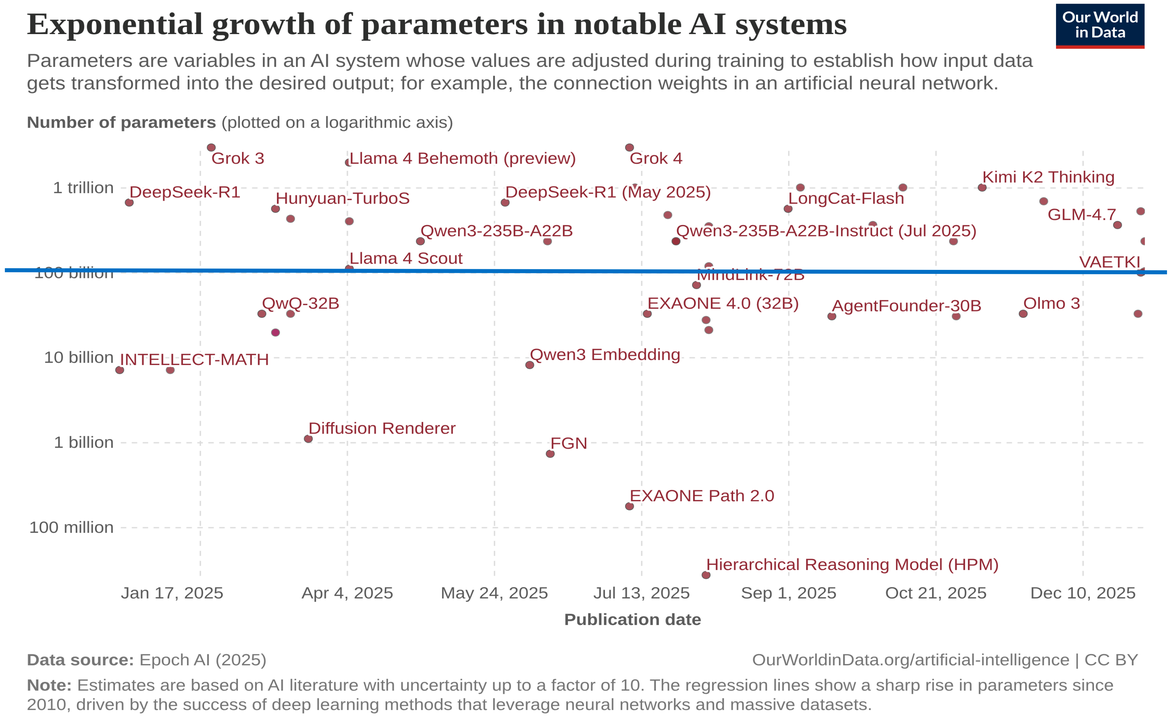
How do we optimize inference?

”

Larger models

Implications for inference

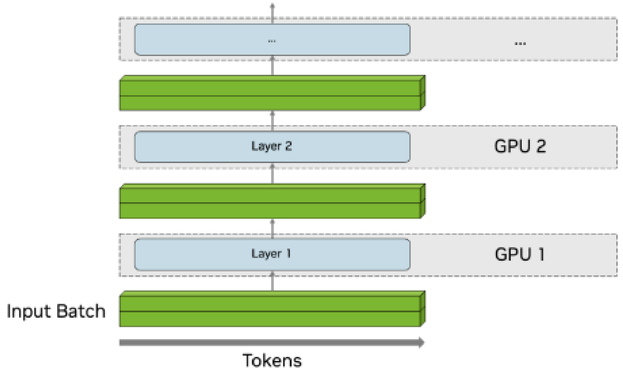
- A single B300 GPU running in BF16 precision can accommodate a 100B-parameter model.
- However, many recent models exceed this size threshold.



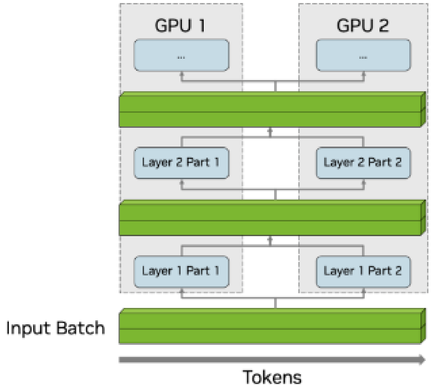
https://ourworldindata.org/grapher/exponential-growth-of-parameters-in-notable-ai-systems?time=2025-01-17..latest&country=

Inference can use Model Parallelism

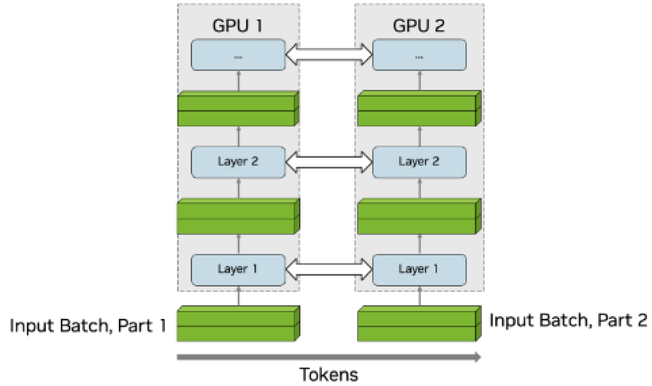
And Data Parallelism



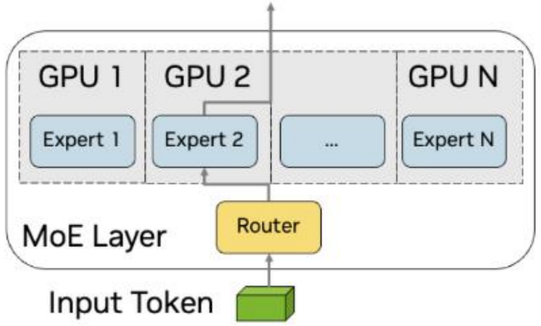
Pipeline Parallelism



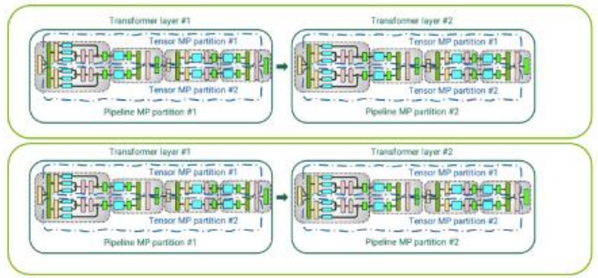
Tensor Parallelism



Context Parallelism



Expert Parallelism



Mixed Parallelism

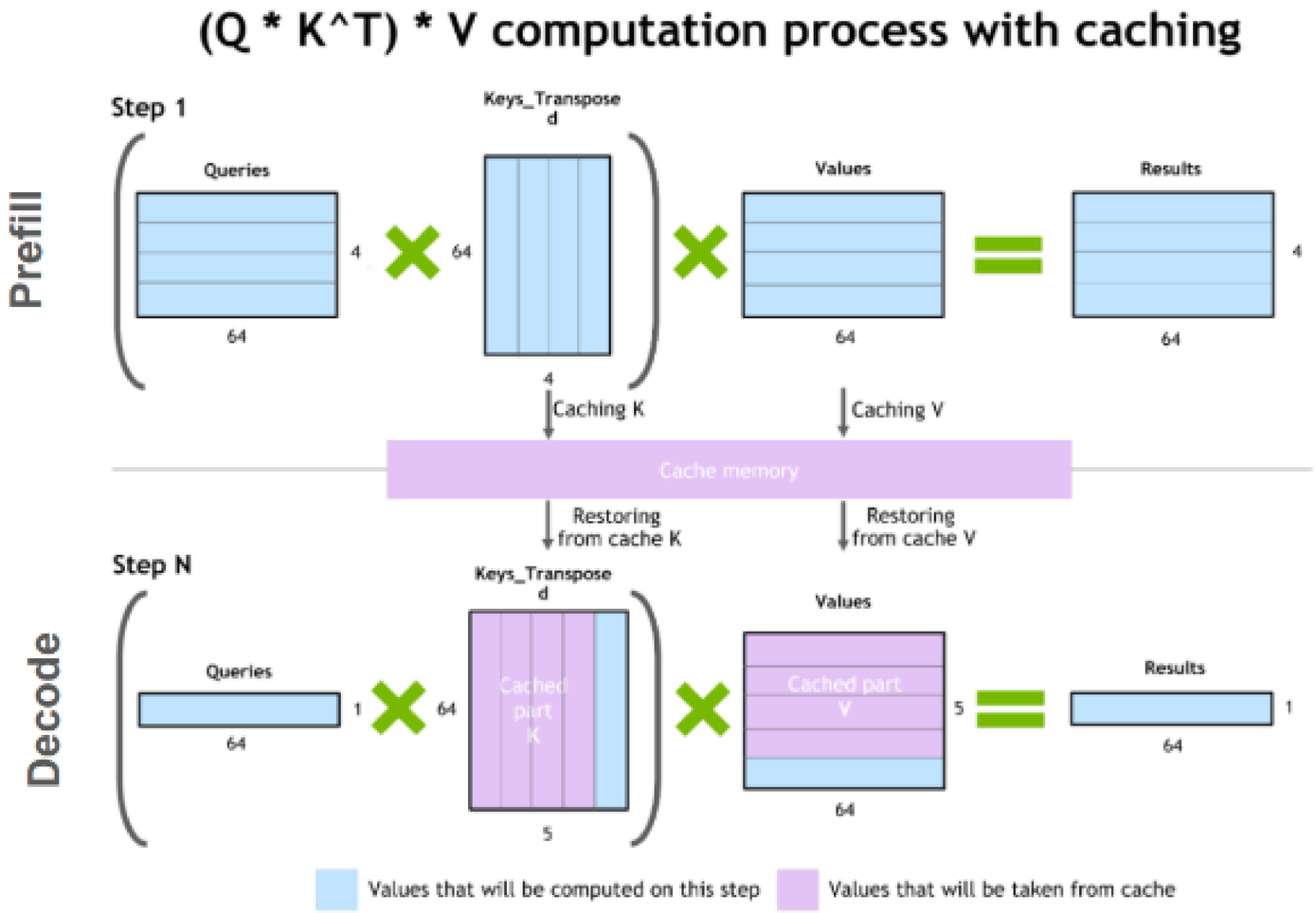
Two Stages of LLM Execution

Prefill vs Decoding and the use of KV-cache

- **Prefill** = Time To First Token (TTFT)
  - Loading the user prompt into the system
  - Populate KV-cache for all the tokens from the prompt.
- **Decode** = Inter-Token Latency (ITL)
  - Generating the response token by token
  - Reuse KV-cache to generate the next token

LLM inference is made of prefill followed by decode

KV-cache


(Q * K^T) * V computation process with caching
Prefill
Step 1
Queries
Keys_Transposed
Values
Results
4
64
64
4
64
64
4
Caching K
Caching V
Cache memory
Restoring from cache K
Restoring from cache V
Decode
Step N
Queries
Keys_Transposed
Values
Results
1
64
64
Cached part K
Cached part V
5
64
5
1
64
Values that will be computed on this step
Values that will be taken from cache

Estimating the size of the KV Cache

Total size of KV cache in bytes = $2 * sizeof(precision) * n_{layers} * d_{model} * seqlen * batch$

$2 = two\ matrices\ of\ K\ and\ V$

$precision = bytes/parameter\ (FP16 = 2bytes)$

$n_{layers} = layers\ in\ the\ model$

$d_{model} = Dimension\ of\ the\ embeddings$

$seqlen = length\ of\ context\ in\ tokens\ (input\ prompt + generated\ output)$

$batch = batch\ size$

Example of a KV Cache size for LLaMa2 7B model in FP16 and a batch size of 1

$2 * 2 * 4096 * 32 * 4096 * 1 = 2GB$

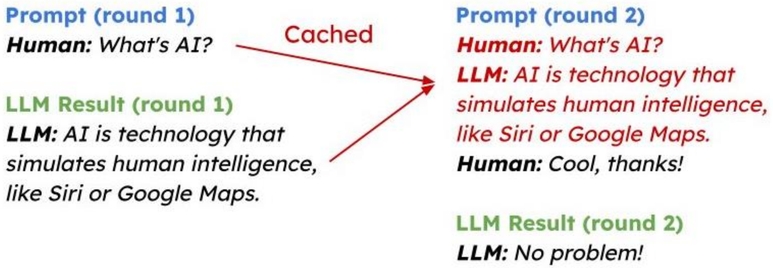
Source: Mastering LLM Techniques: Inference Optimization

Prefix caching

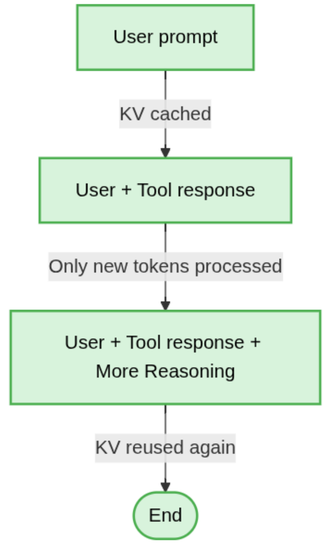
KV cache reuse

KV cache are stored and can be used when generating different responses to the prompts that contain similar prefix

Multi-turn conversation



Agent reuse after tool usage



System prompt

Request A	*A chat between a curious user and an artificial intelligence assistant. The assistant gives helpful, detailed, and polite answers to the user's questions. User: Hello!*
Request B	*A chat between a curious user and an artificial intelligence assistant. The assistant gives helpful, detailed, and polite answers to the user's questions. User: How are you?*
Request C	*A chat between a curious user and an artificial intelligence assistant. The assistant speaks French. User: Bonjour!*

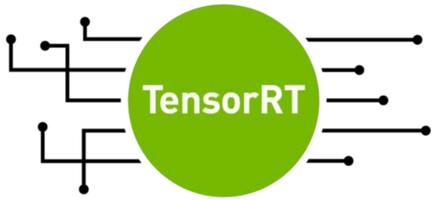
Inference Optimizations before Scaling

Non-distributed inference optimizations

Compression and low-precision

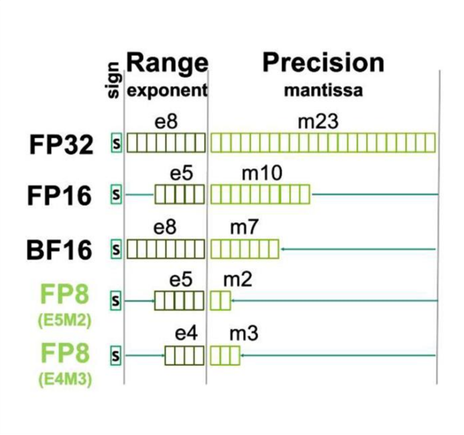
Serving Execution Strategies

Prefix caching

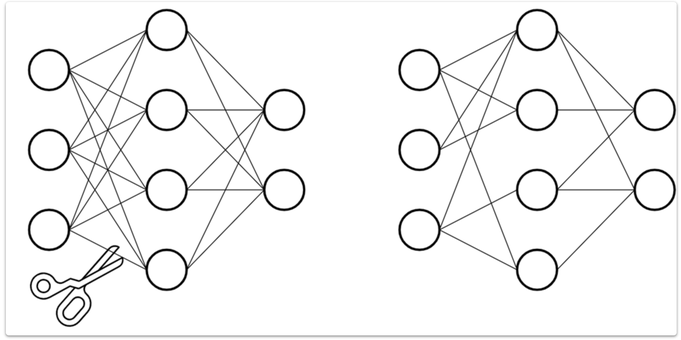


vLLM: https://catalog.ngc.nvidia.com/orgs/nvidia/containers/vllm/tags?version=25.10-py3
SGLang: https://catalog.ngc.nvidia.com/orgs/nvidia/containers/sglang?version=25.10-py3

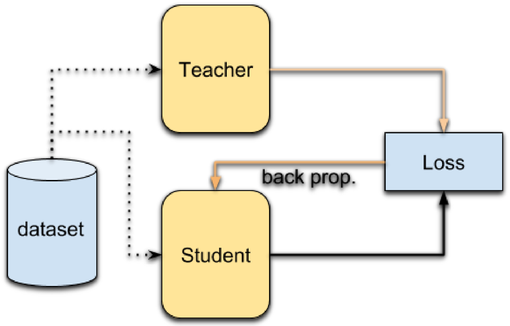
Model Compression Strategies



1. Quantization and low-precision



2. Pruning



3. Distillation

Benefits of low-precision

Memory
Weights and tensors occupy less space in memory

Bandwidth
Faster data movement from main memory HBM to cores (and vice versa)

Compute
More TFLOPS with less bits
Faster matrix multiplications

NVFP4 speed-up

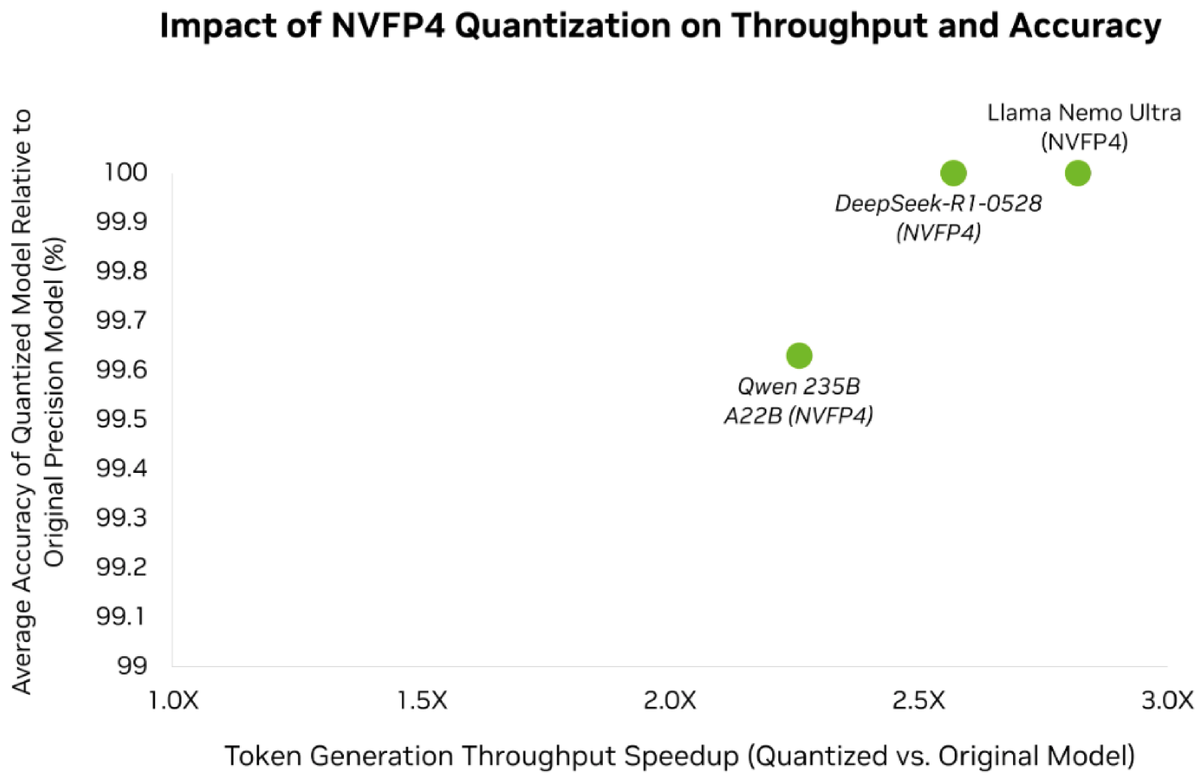


Figure 8. Cross-plot showing NVFP4 token generation throughput speedup and accuracy impact

https://developer.nvidia.com/blog/optimizing-llms-for-performance-and-accuracy-with-post-training-quantization/

Hugging Face | Search models, datasets, users... | Models | Datasets | Spaces | Community | Docs | Enterprise | Pricing

nvidia's Collections + New

- NVIDIA Nemotron v3
- Sortformer
- Inference Optimized Checkpoints (with Model Optimizer)
- Nemotron OCR and Object Detection
- Nemotron ColEmbed V2
- Steering Reasoning VLAs
- Nemotron Inference SW
- Earth-2
- Isaac-for-Healthcare-Dev
- Nemotron-Flash
- NVIDIA Cosmos 2
- Nemotron-Cascade
- Nemotron Speech
- Nemotron-Post-Training-v3
- Nemotron-Pre-Training-Datasets
- Nemotron RAG
- NeMo Gym
- Cosmos Policy
- KVzap
- NVIDIA Nemotron V2
- Speculative Decoding Modules

Inference Optimized Checkpoints (with Model Optimizer)

updated about 7 hours ago

A collection of generative models quantized and optimized for inference with Model Optimizer.

- nvidia/NVIDIA-Nemotron-3-Nano-30B-A3B-FP8 — Text Generation • 32B • Updated 7 days ago • 710k • 276
- nvidia/DeepSeek-R1-0528-NVFP4-v2 — Text Generation • 394B • Updated Sep 2, 2025 • 107k • 11
- nvidia/Kimi-K2-Thinking-NVFP4 — Text Generation • Updated 28 days ago • 32.6k • 20
- nvidia/Kimi-K2.5-NVFP4 — Text Generation • Updated 1 day ago • 2.6k • 8
- nvidia/Llama-3.3-70B-Instruct-NVFP4 — 41B • Updated Aug 22, 2025 • 10.3k • 32
- nvidia/Llama-3.3-70B-Instruct-FP8 — 71B • Updated Aug 22, 2025 • 18.6k • 15
- nvidia/Llama-3.1-8B-Instruct-NVFP4 — 5B • Updated Sep 16, 2025 • 90.7k • 6
- nvidia/Llama-3.1-8B-Instruct-FP8 — Text Generation • 8B • Updated Aug 22, 2025 • 41.6k • 32
- nvidia/DeepSeek-R1-0528-NVFP4 — Text Generation • 397B • Updated Aug 22, 2025 • 14.6k • 40

Upvote 83 +79

- Add to collection
- Theme
- Share collection
- View history
- Collection guide
- Browse collections
- Public
- Delete collection

Gating Group

https://huggingface.co/collections/nvidia/inference-optimized-checkpoints-with-model-optimizer

NVIDIA

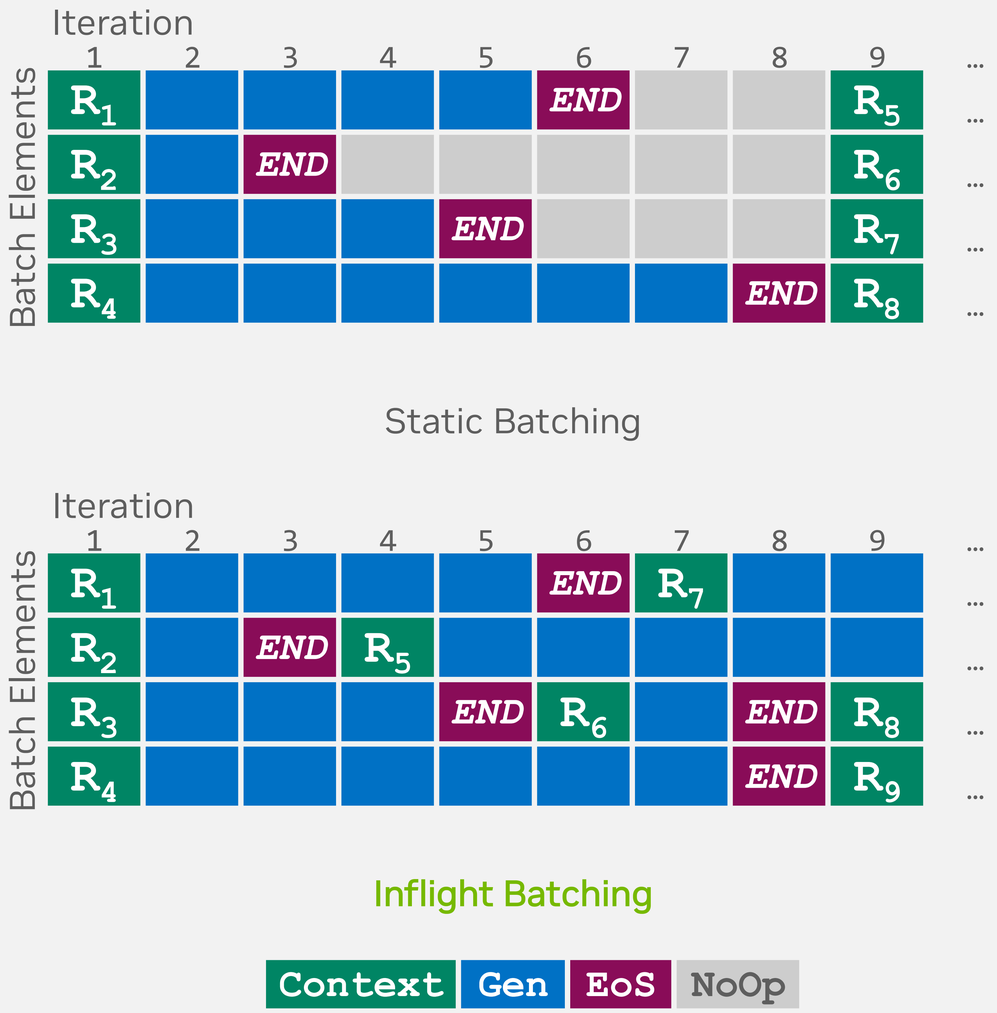
Inference Engine optimizations

Inflight/continuous Batching

Maximizing GPU Utilization during LLM Serving

Inflight Batching to optimize GPU utilization during LLM Serving

- Replaces completed requests in the batch
  - Evicts requests after EoS & inserts a new request
- Improves throughput, time to first token, & GPU utilization



KV Cache Optimizations

Paged & Quantized KV Cache

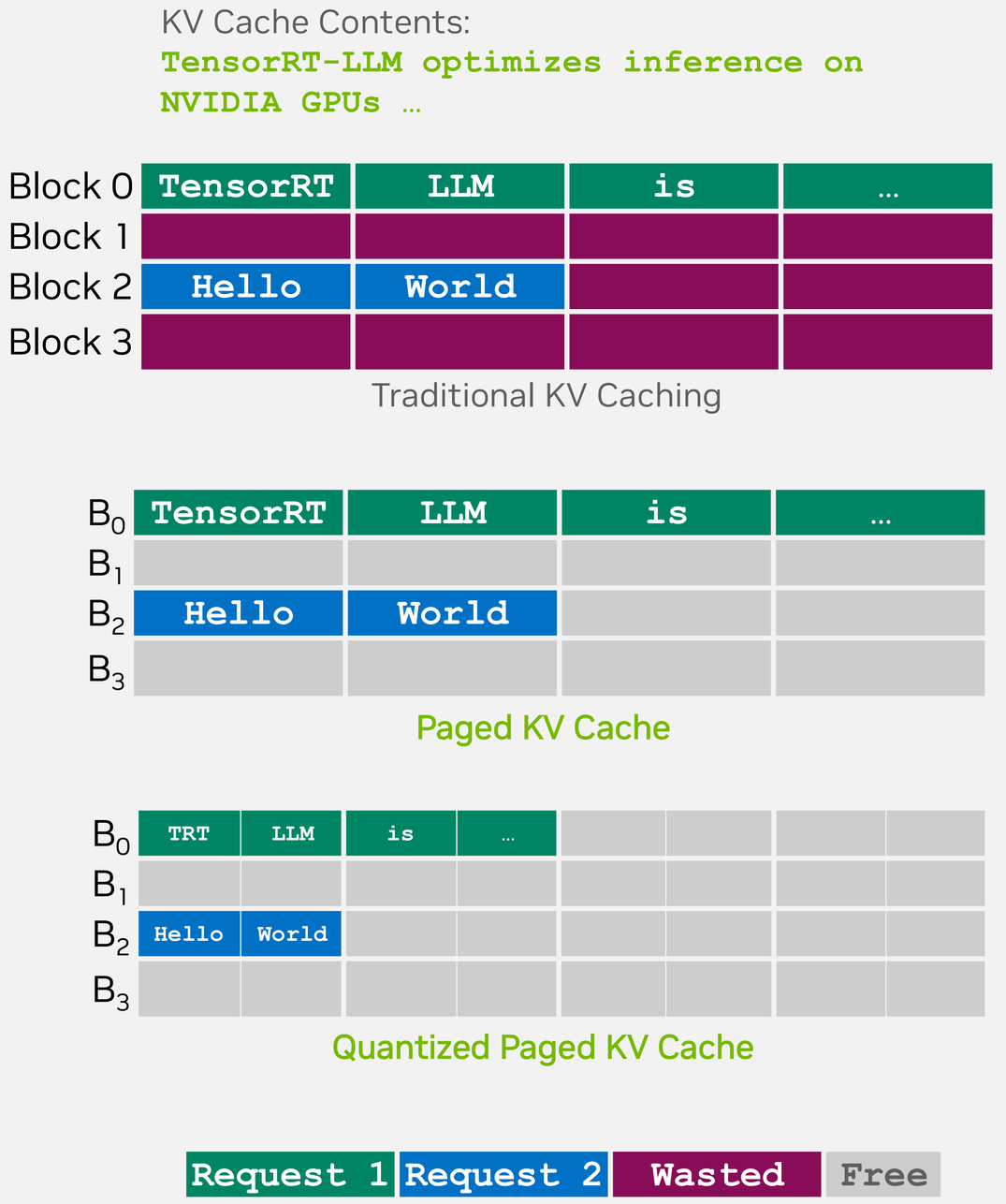
Paged KV Cache improves memory consumption & utilization

- Stores keys & values in non-contiguous memory space
- Allows for reduced memory consumption of KV cache
- Allocates memory on demand

Quantized KV Cache improves memory consumption & perf

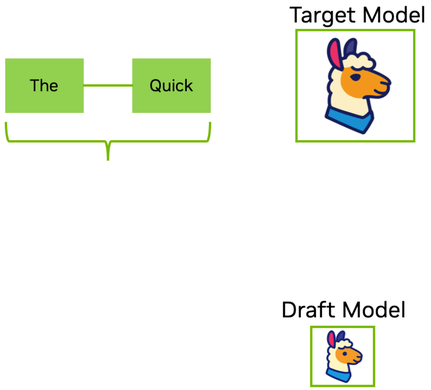
- Reduces KV Cache elements from 16b to 8b (or less!)
- Reduces memory transfer improving performance
- Supports INT8 / FP8 KV Caches

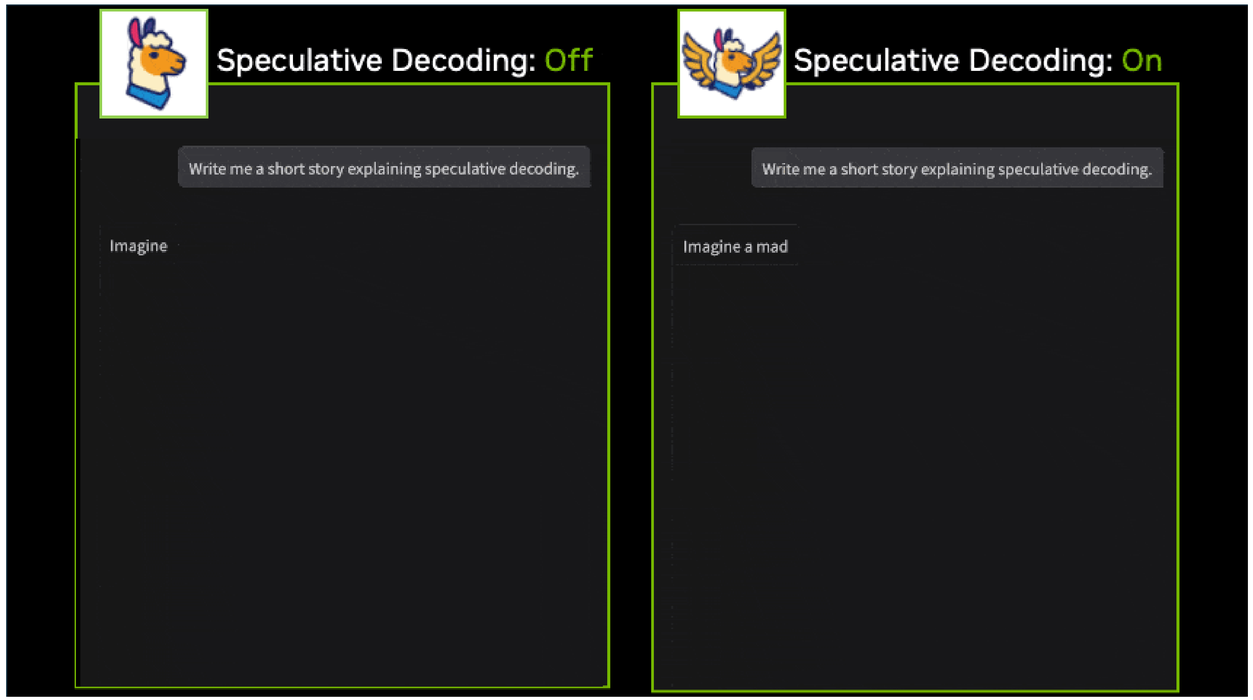
Both allow for increased peak performance



Speculative decoding

- Decode step is sequential
- Speculative decoding is to use smaller, faster draft model to propose multiple tokens ahead, then target model verifies in parallel
- **Advantage**: Generates multiple tokens per step, cuts drastically latency and throughput

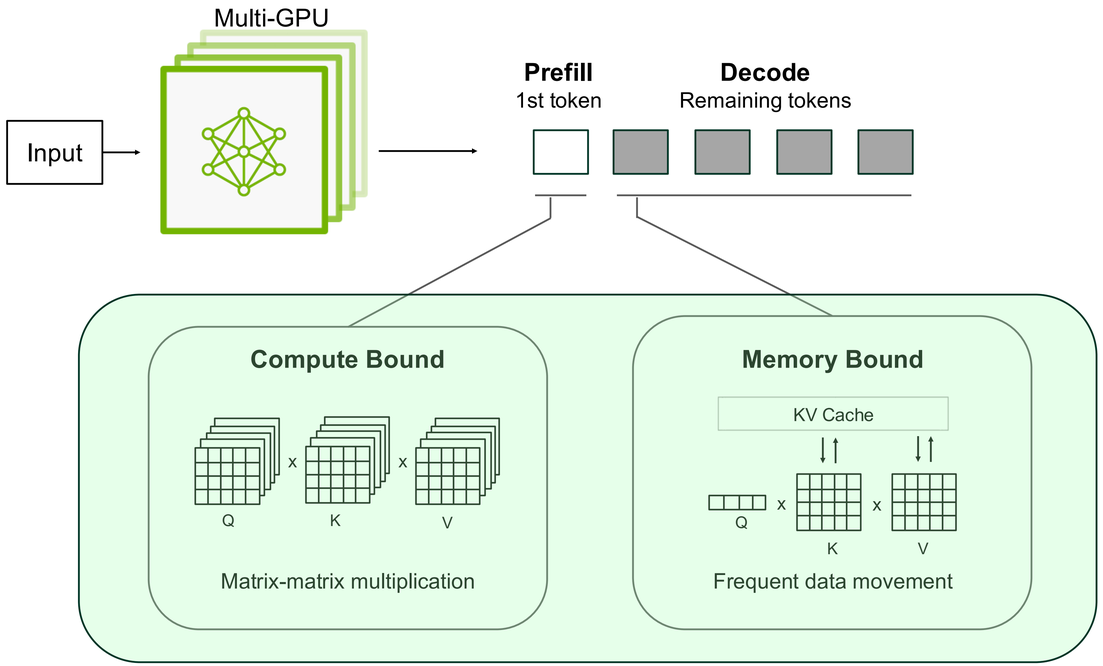




https://developer.nvidia.com/blog/an-introduction-to-speculative-decoding-for-reducing-latency-in-ai-inference/

LLM inference

The effect of monolithic inference

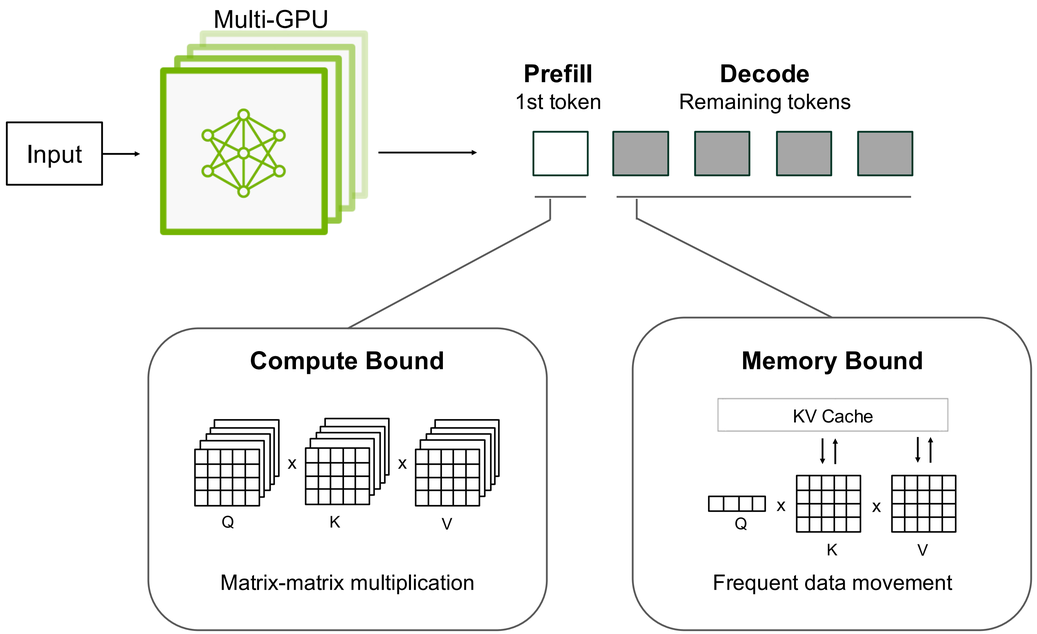


Share same parallelism strategy

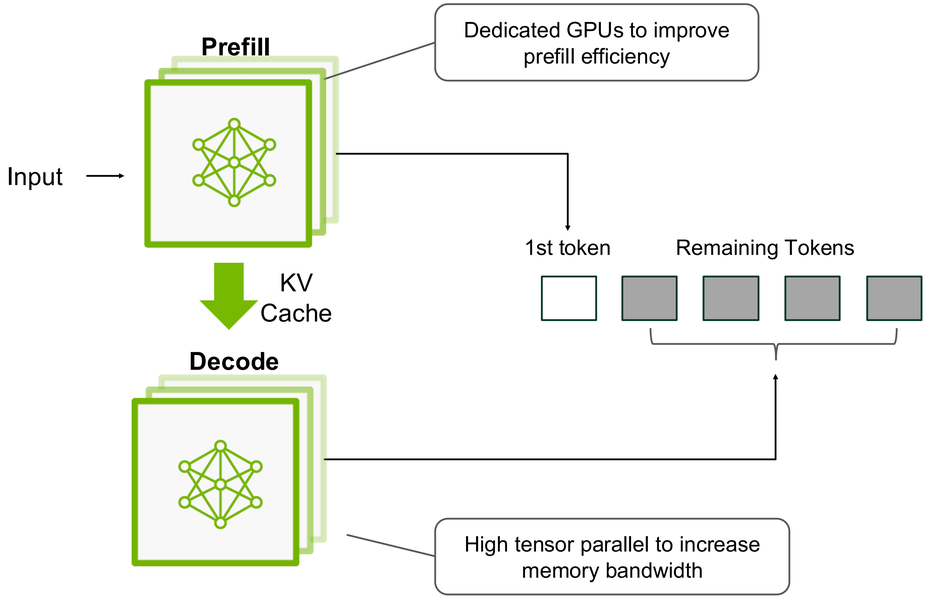
New Inference Optimization Techniques to Boost Inference

Disaggregated serving separates prefill and decode allowing each to be optimized independently

Traditional Serving



Disaggregated Serving



More flexibility to optimize cost and user experience

Why Different Parallelism Strategies Matter

- Disaggregation allows each worker type (Prefill vs Decode) to use the parallelism method best suited to its workload.

- Prefill prefers **low tensor parallelism**, **higher data parallelism** (large batches, less communication)

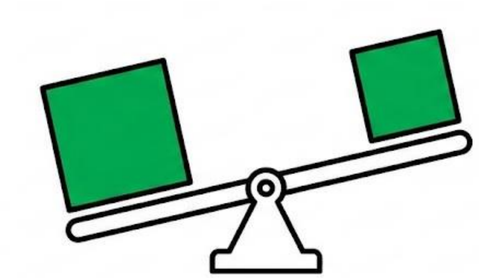
- Decode prefers **high tensor parallelism**, **high expert parallelism** (for MoE)

- **Advantages**:
- Independent tuning of Tensor / Expert / Data parallelism per phase
- Better GPU utilization under mixed workloads
- Freedom to use different GPU types for prefill vs decode
- Higher throughput & better latency for real-world workloads

Inference as multiple workloads

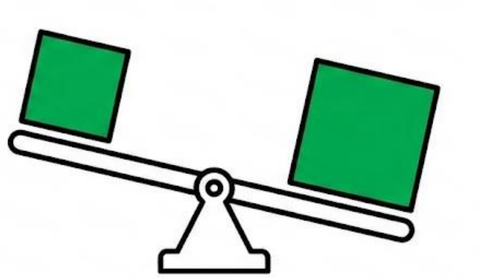
Prefill-heavy workload

- Short outputs, long prompts
- Example workloads: retrieval-augmented QA with large contexts, classification
- Most FLOPs are in prefill; decode is relatively small.
- Workers split: 2:1 or 3:1 prefill:decode workers to keep TTFT and prefill ingestion fast.



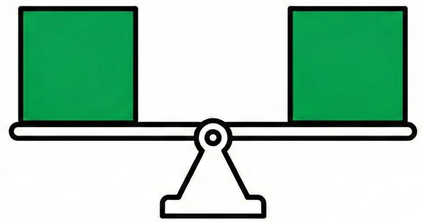
Decode-heavy workload

- Long outputs, moderate prompts
- Example workloads: reasoning, code generation, story writing, assistant chat with long replies
- Decode dominates latency and throughput; inter-token latency is critical for “streaming” UX.
- Workers split: 1:3 prefill:decode workers



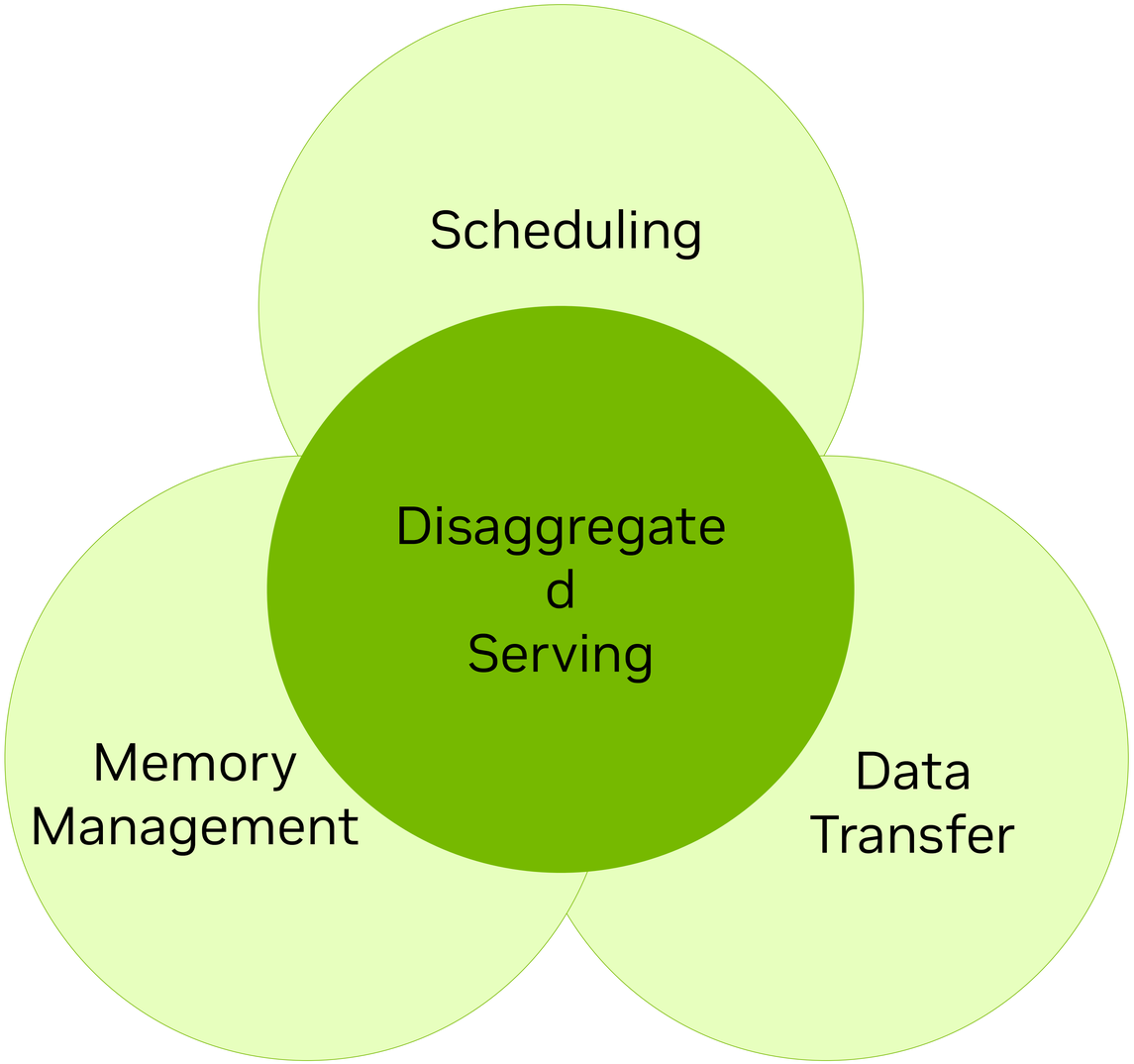
Balanced or mixed workload

- Balanced or mixed workload — Prompt and response lengths are same order, and traffic mix is diverse
- ISL and OSL are comparable on average
- Workers split: around 1:1
- More prefill = better TTFT, worse streaming smoothness under load.
- More decode = smoother streaming and higher steady-state throughput, but TTFT may increase



Dynamo

Systematic approach to AI inference at scale



System level optimization

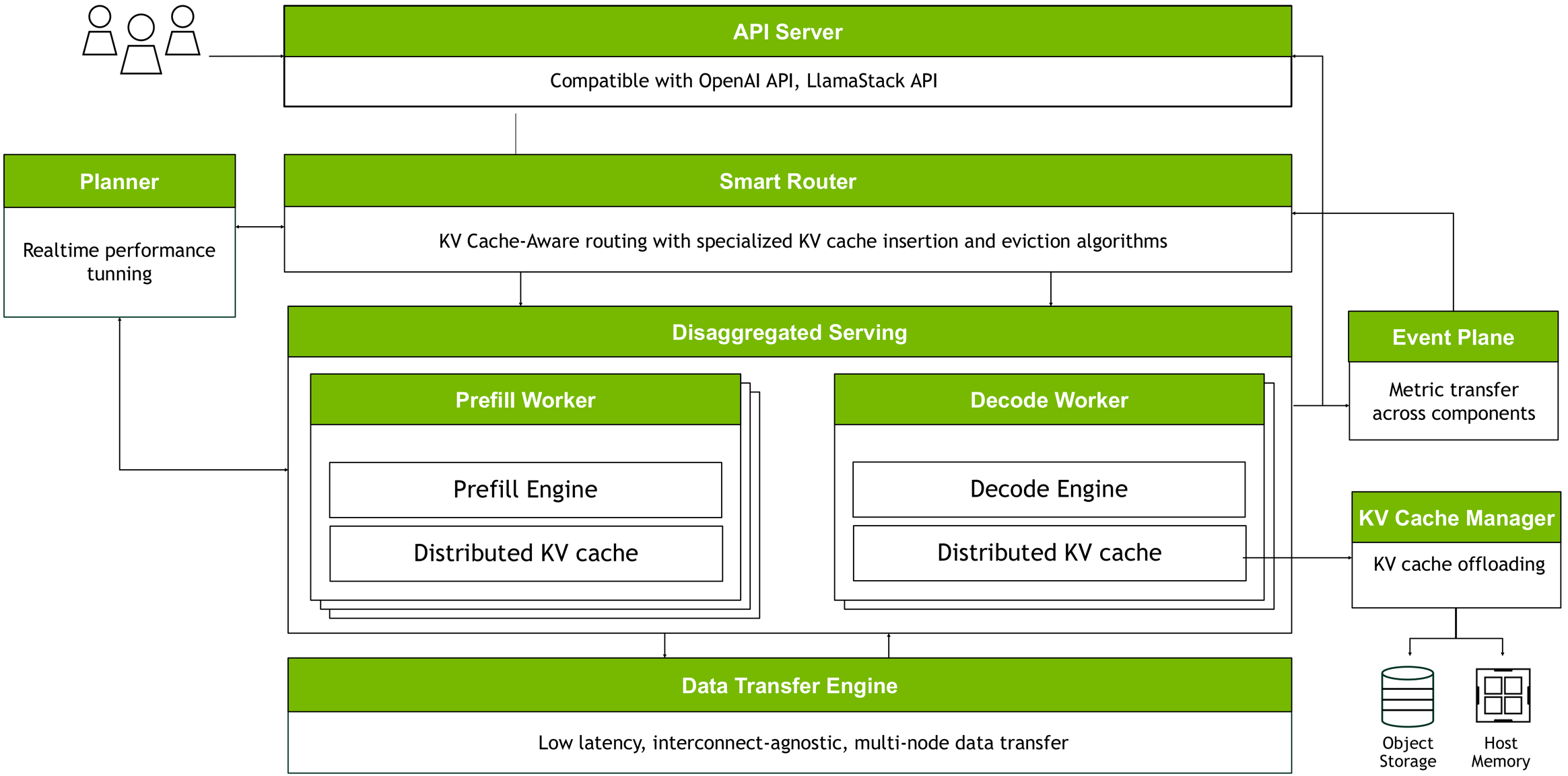
Supports

SGLang, TensorRT-LLM, & vLLM

Modular components

Config to production

Architecture and Components



Benefits of disaggregation on multiple nodes

Benchmark on multiple nodes

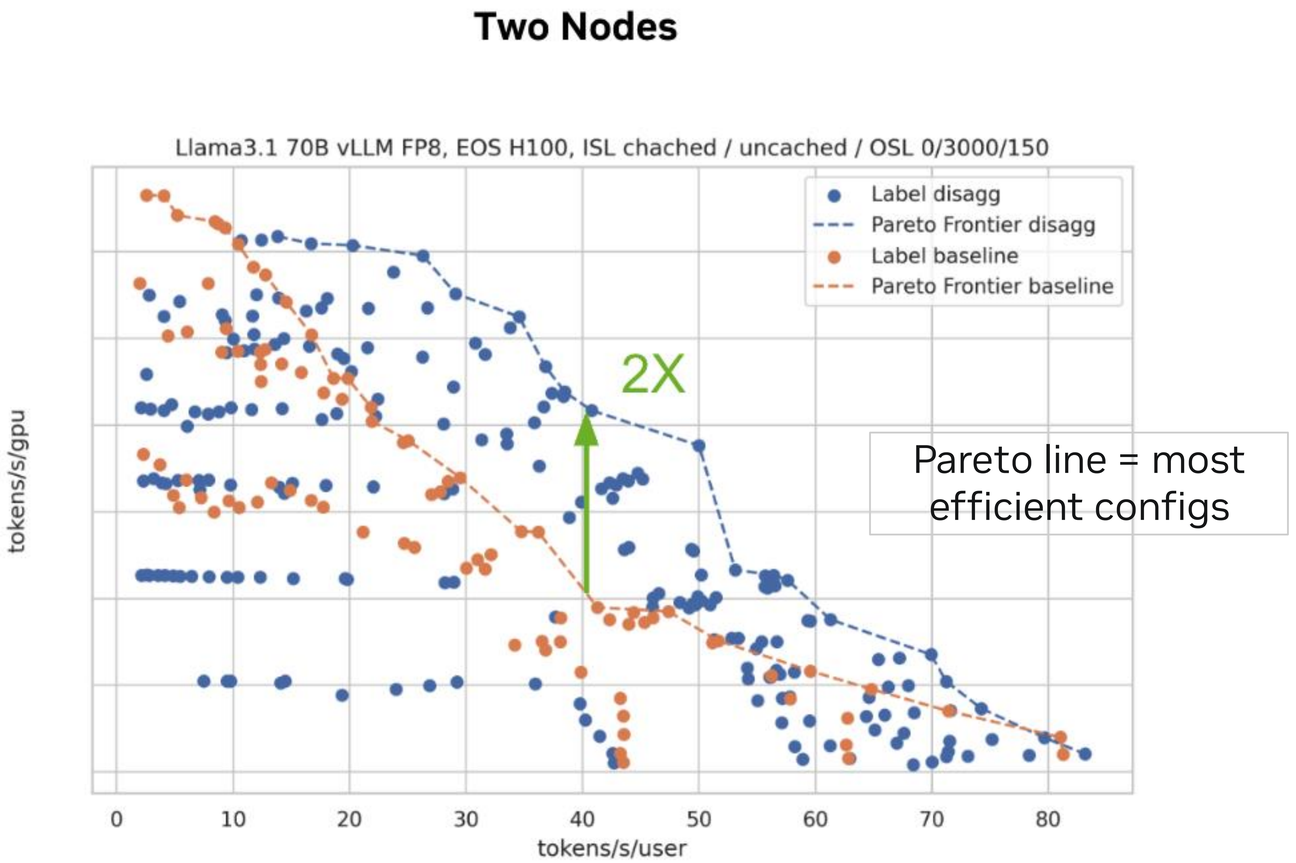
Key result:

- disaggregated consistently outperforms aggregated
- Up to 2× throughput per GPU improvement

Config insights:

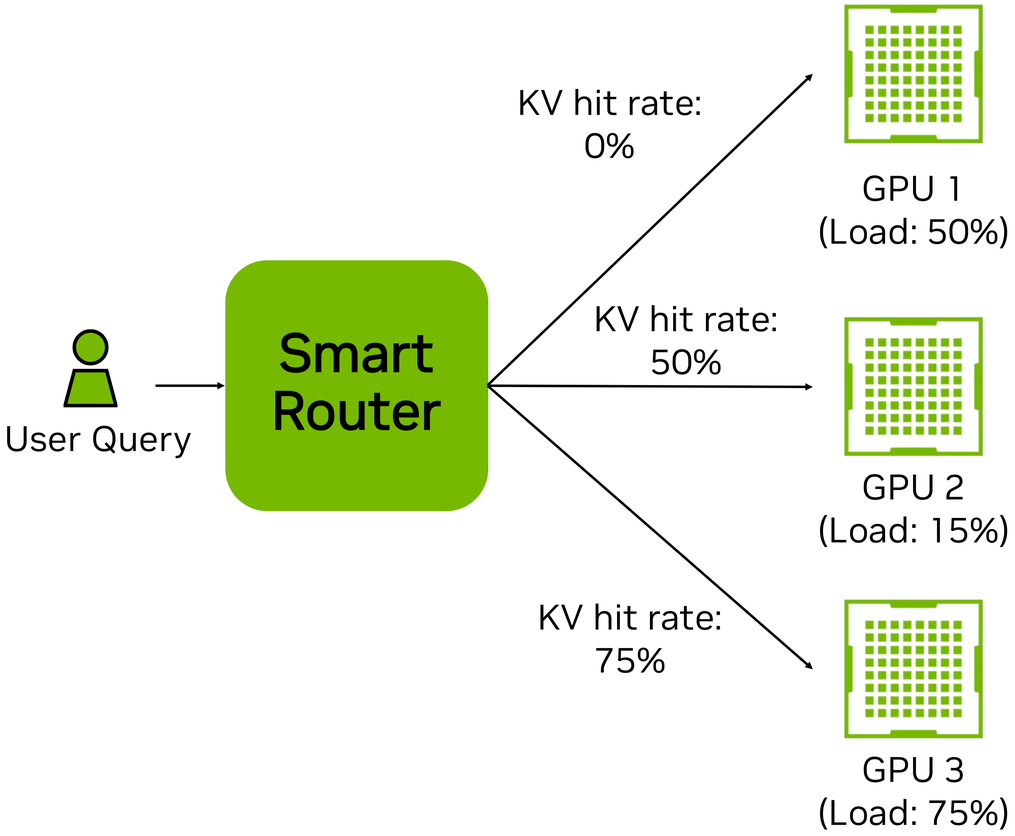
- Aggregated best: TP8 DP2
- Disaggregated best: prefill TP2 DP4, decode TP8

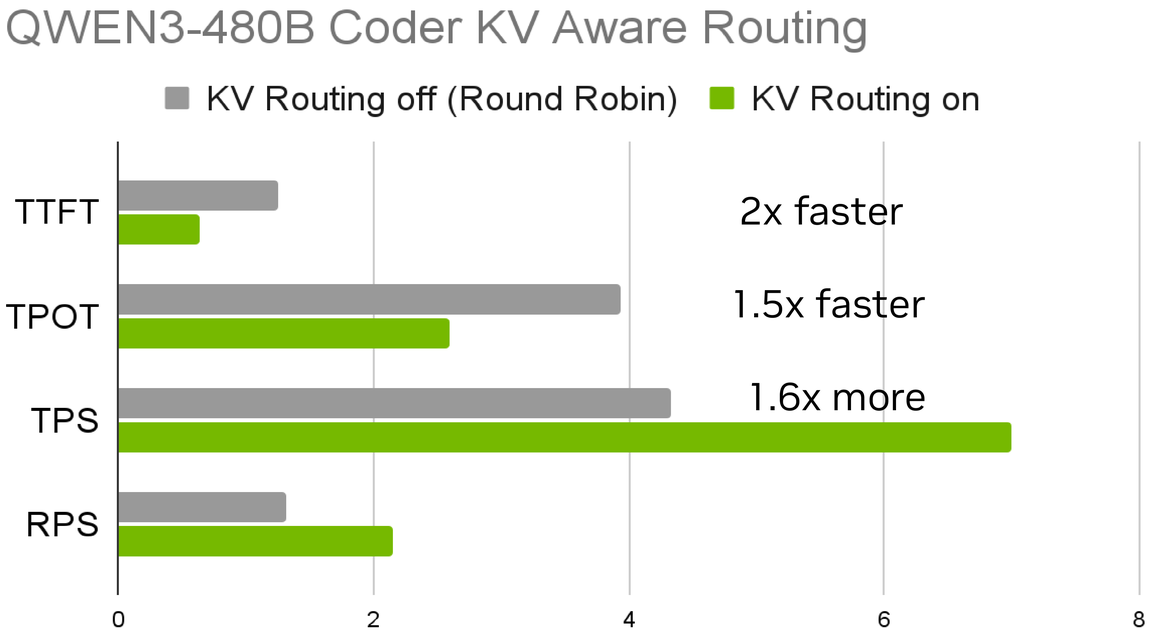
Prefill favors more data parallelism for batch efficiency
Decode favors higher tensor parallelism for GPU utilization



NVIDIA Dynamo: Smart Router

Reducing costly re-computation of KV cache

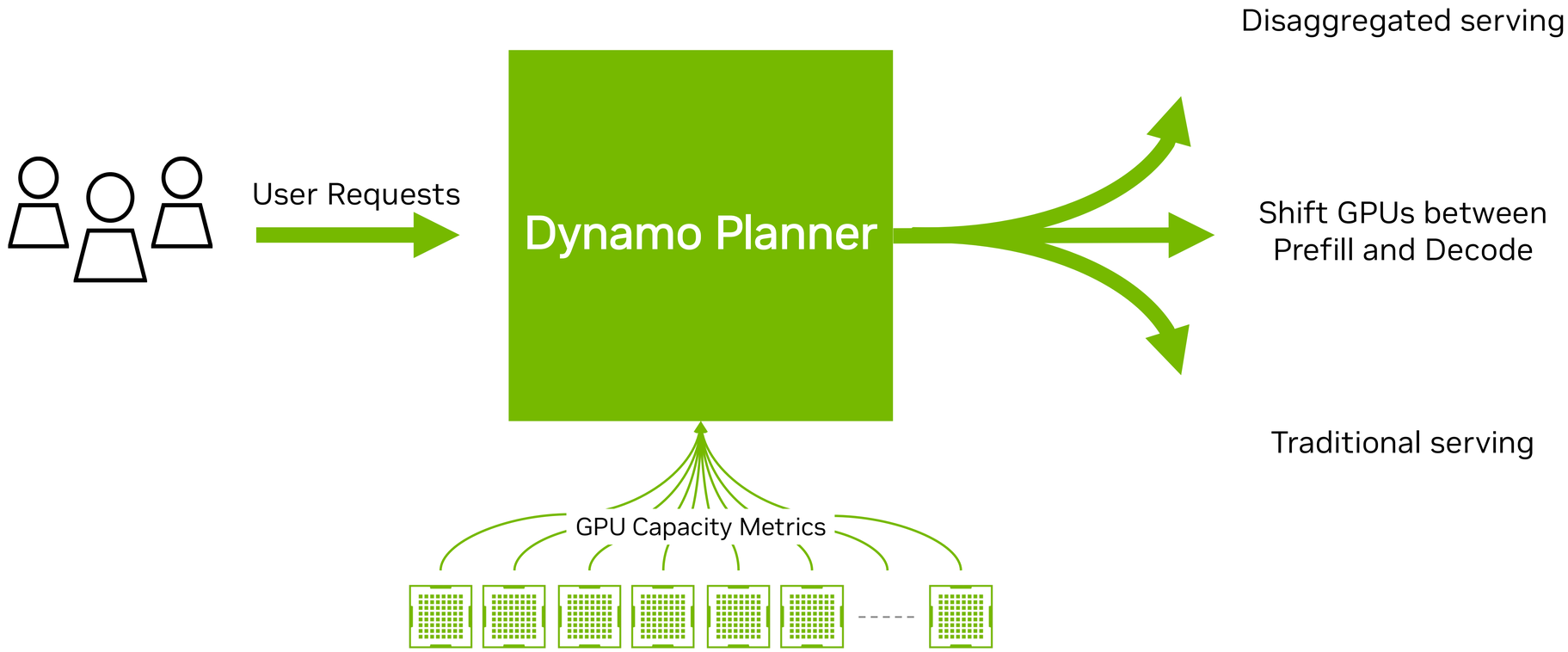




https://www.baseten.co/blog/how-baseten-achieved-2x-faster-inference-with-nvidia-dynamo/#qwen3-coder-benchmarks-with-kv-routing

NVIDIA Dynamo: GPU Planner

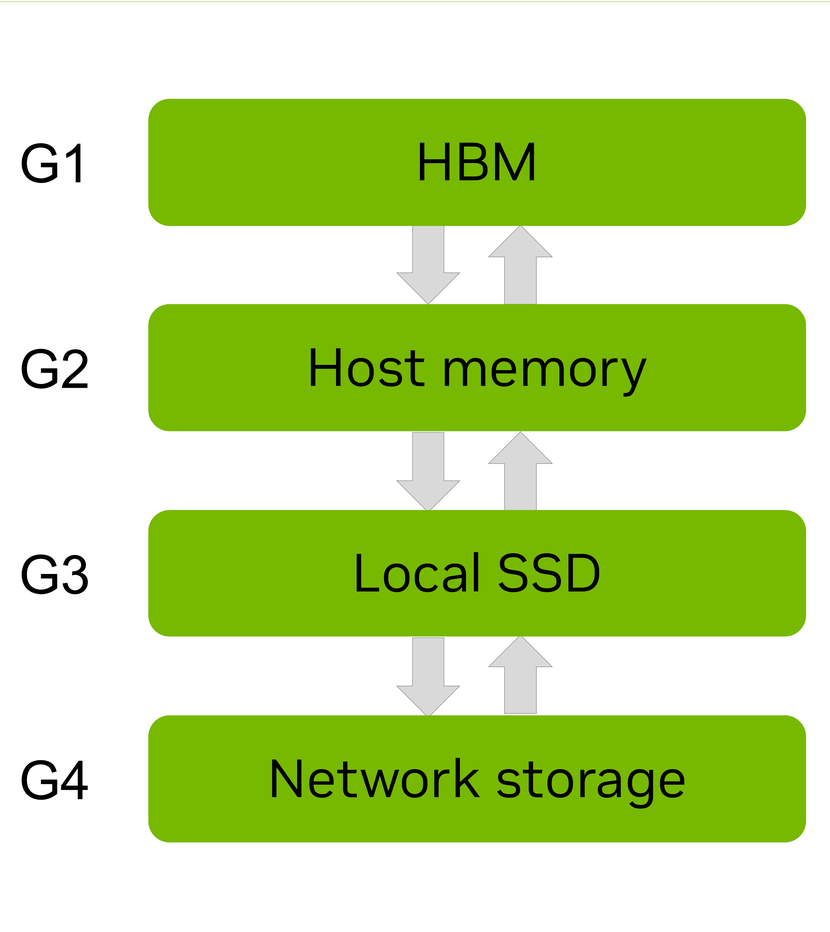
Optimizing GPU resources for distributed inference

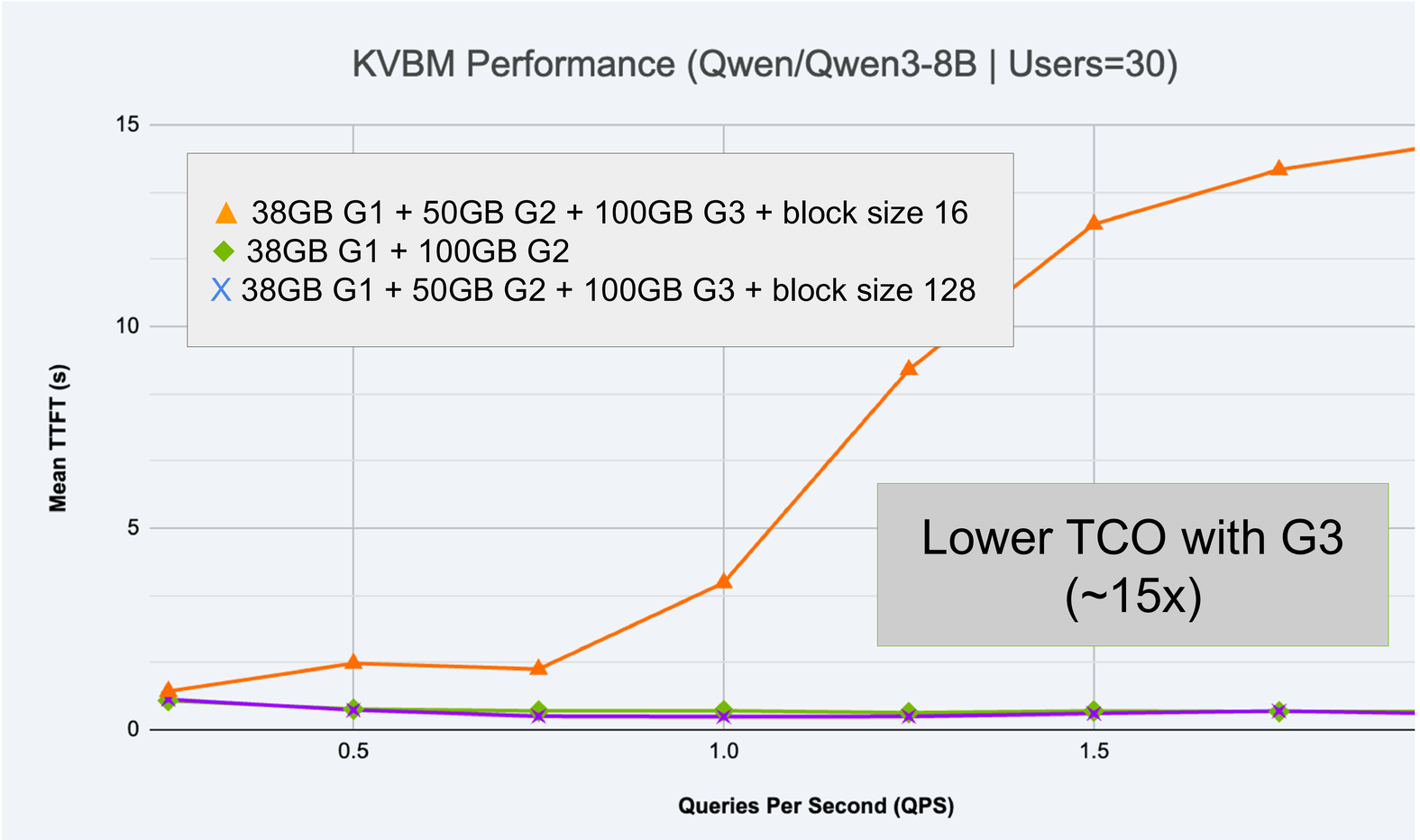


Efficient Resource Allocation | Adjust to Fluctuating Demand | Lower Inference Costs

Memory Management

Leverage all memory and storage available in the data center

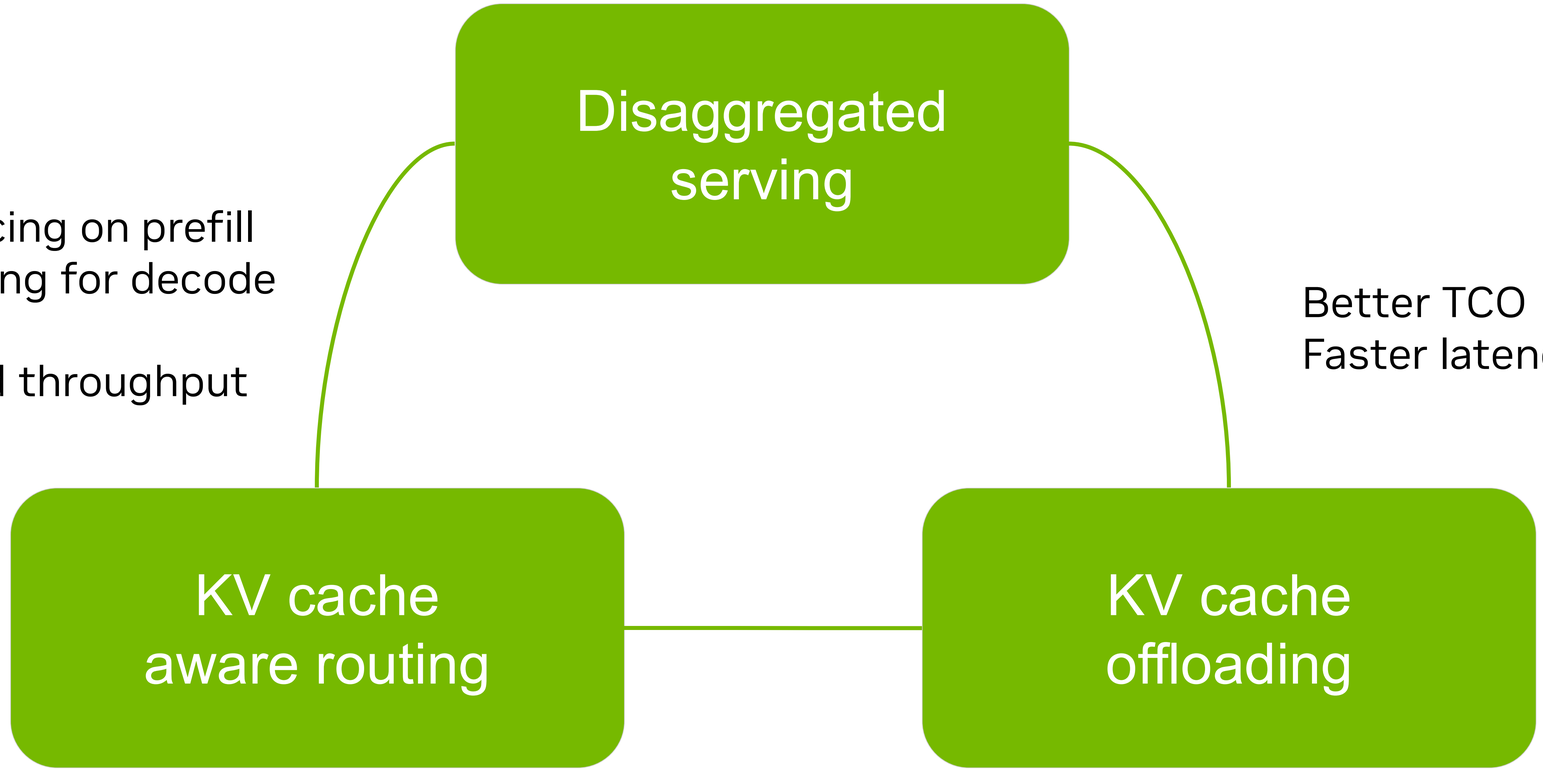




Dynamo strengths

Compute load balancing on prefill
Memory load balancing for decode

-> Better latency and throughput



Better TCO (Less prefill workers)
Faster latency (TTFT)

Increased KV cache hit

Where will Dynamo shine?

But not a requirement

- you use Tensor Parallelism (model does not fit on a single GPU)
- you have fluctuating traffic that requires dynamic GPU scheduling.
- your application has repeating requests, very suitable for KV/prefix caching (reasoning model, agents, multi-turn chat)
- you want to keep as much "history" (KV/prefix cache) as possible, to reduce TTFT
- you have mixed workloads in terms of prefill / decode requirements

Benchmarking via **AIperf**

Benchmarking tool for LLM endpoints

- AIPerf is NVIDIA client-side benchmarking tool that measures real LLM serving metrics like TTFT, ITL, throughput, and latency against any OpenAI-compatible inference endpoint.

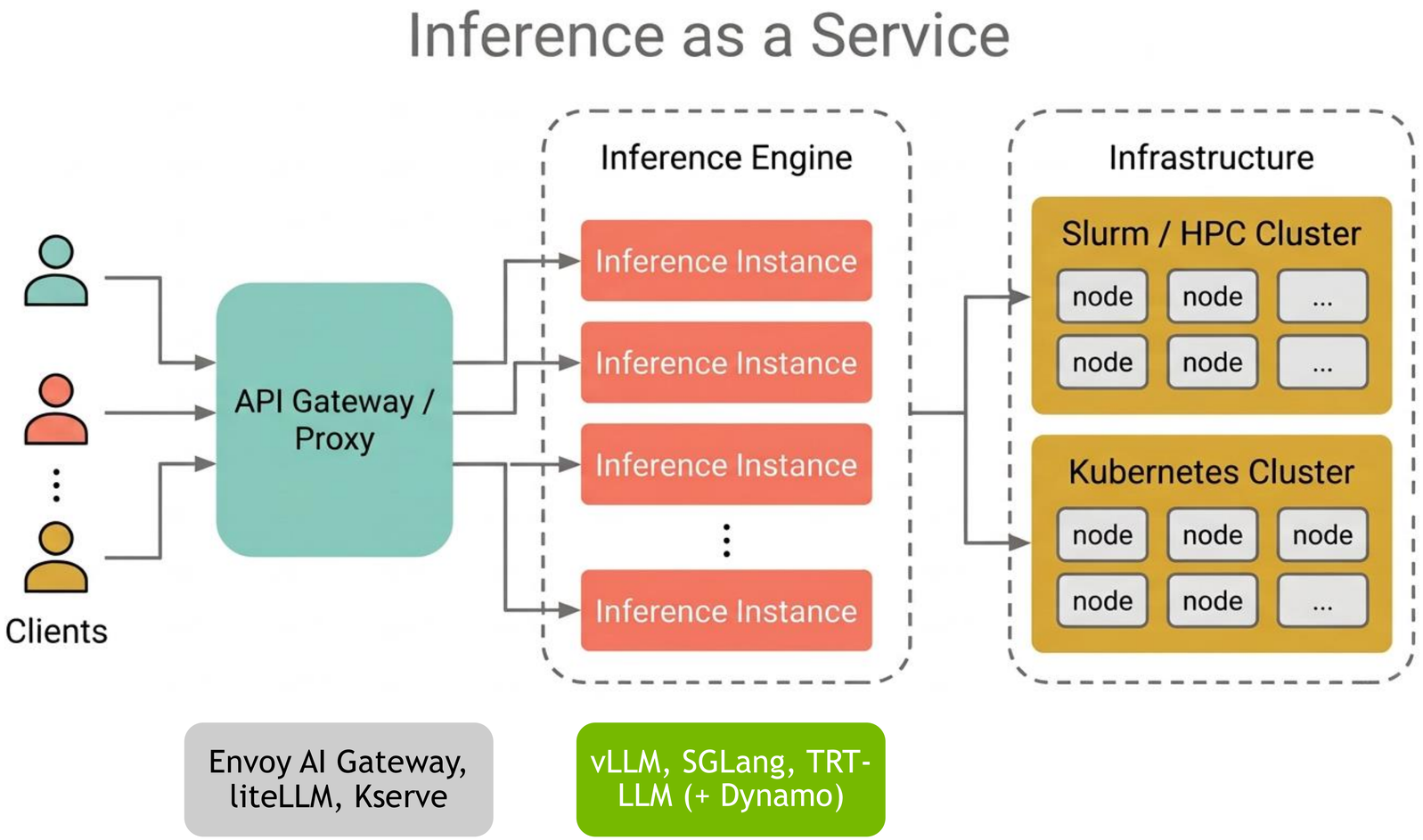
- Can be used with any LLM backend (for example NIM, vLLM, Triton, Dynamo)
- Gives a standardized way to compare inference performance across different models and serving stacks.

NVIDIA AIPerf | LLM Metrics

Metric	avg	min	max	p99	p90	p75	std
Time to First Token (ms)	89.40	19.33	131.72	131.69	131.45	123.02	31.11
Time to Second Token (ms)	20.17	5.09	43.43	43.43	43.36	41.84	17.80
Request Latency (ms)	1,193.64	1,156.45	1,223.08	1,223.03	1,222.84	1,211.86	24.80
Inter Token Latency (ms)	5.55	5.42	5.76	5.75	5.64	5.63	0.10
Output Token Throughput Per User (tokens/sec/user)	180.27	173.62	184.54	184.54	184.53	184.21	3.30
Output Sequence Length (tokens)	200.00	200.00	200.00	200.00	200.00	200.00	0.00
Input Sequence Length (tokens)	200.77	181.00	218.00	216.84	214.00	207.75	9.38
Output Token Throughput (tokens/sec)	1,668.68	N/A	N/A	N/A	N/A	N/A	N/A
Request Throughput (requests/sec)	8.34	N/A	N/A	N/A	N/A	N/A	N/A
Request Count (requests)	30.00	N/A	N/A	N/A	N/A	N/A	N/A

Inference as a Service

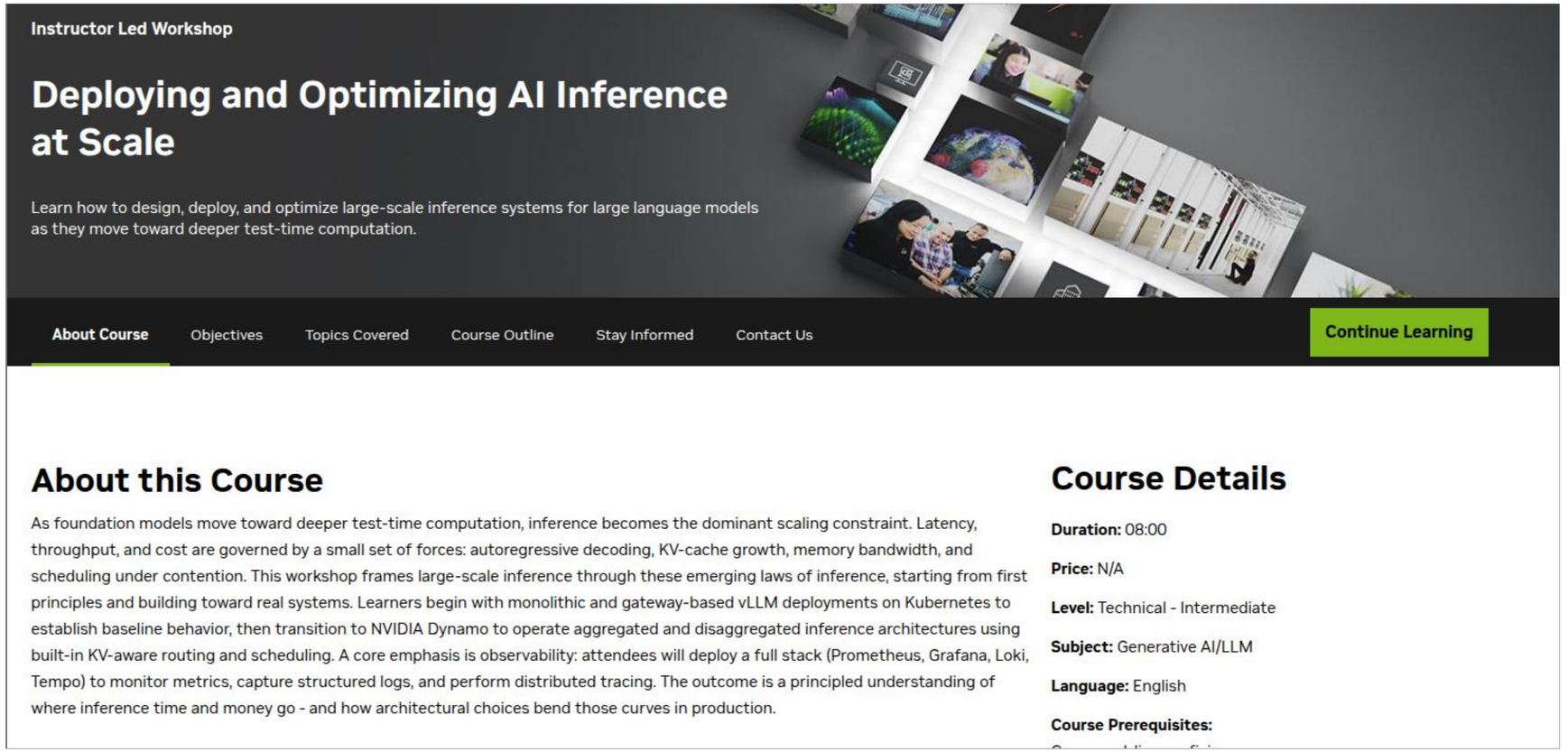
Bring inference to your cluster



Learn more

Instructor-led workshop to deep dive

Learn how to architect your first production-ready inference as a service



Inference is the next HPC challenge

- The workload is here: inference demand is growing
  - reasoning models and agents multiply compute exponentially
  - **Your cluster will serve inference in any case**

- Optimizations are compounding: compression, request serving, caching, disaggregation, KV-cache routing and KV-cache offloading
- NVIDIA Dynamo brings all of this together on your cluster

- The software stack exists already; it is time to build your service

Happy to collaborate with you if you are building such infrastructures and services!

Questions?

Contact: shabert@nvidia.com