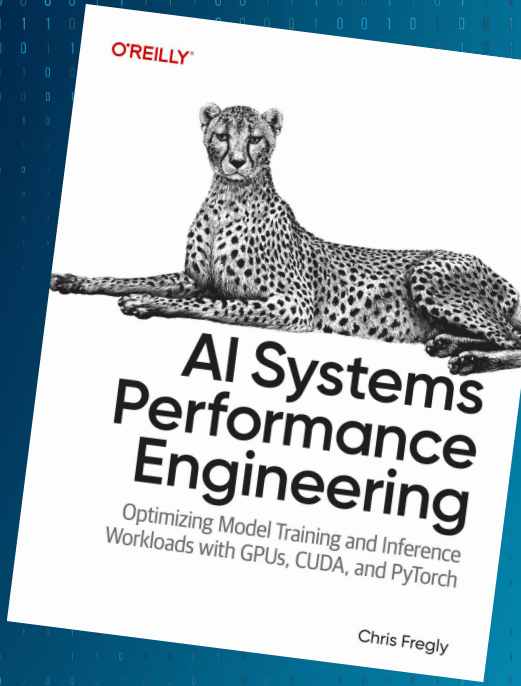
High Performance Agentic AI Inference Systems



Chris Fregly

https://www.amazon.com/Systems-Performance-Engineering-Optimizing-Inference/dp/B0F47689K8/

Why focus on inference performance?

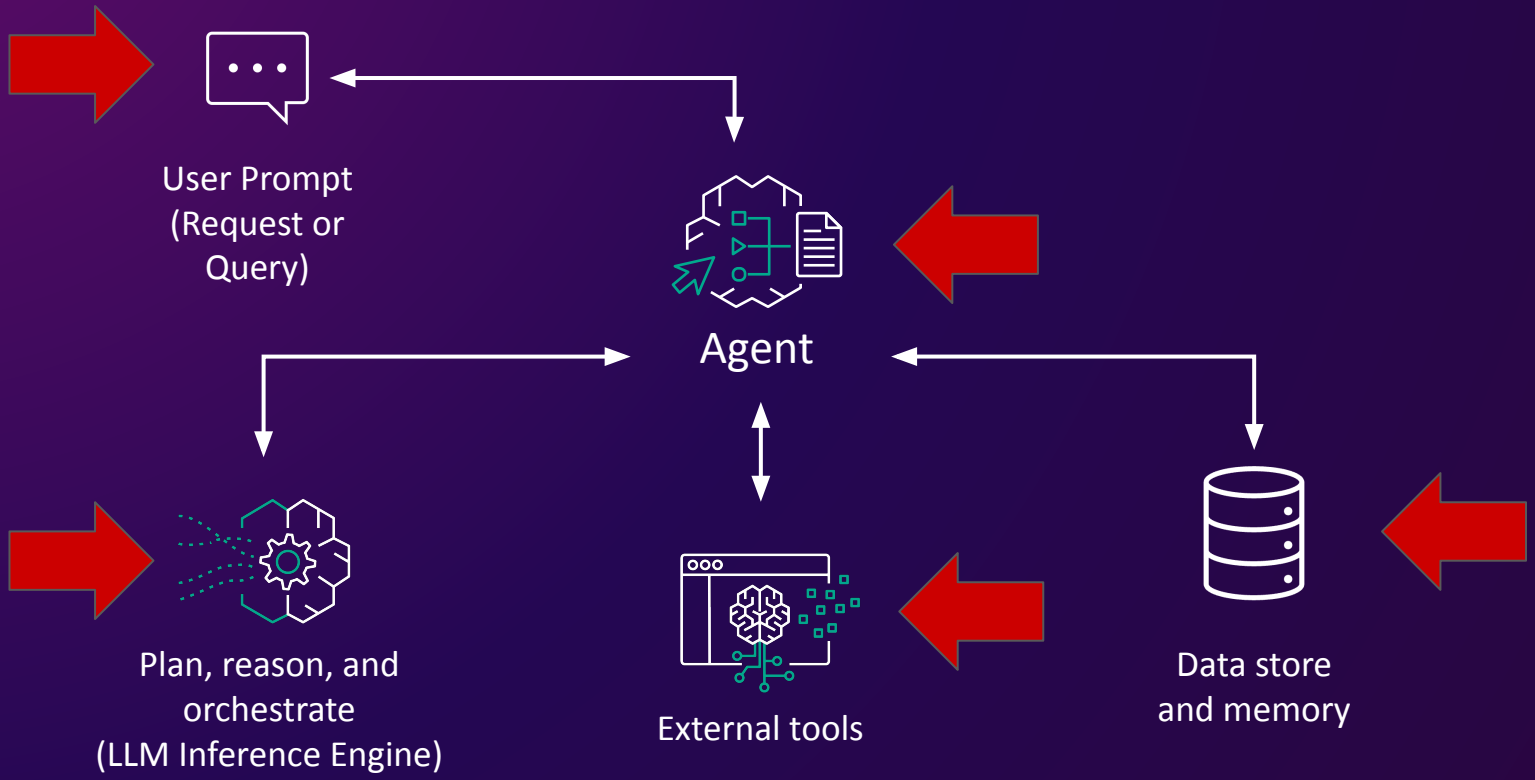
Faster time-to-response

Better end user experience

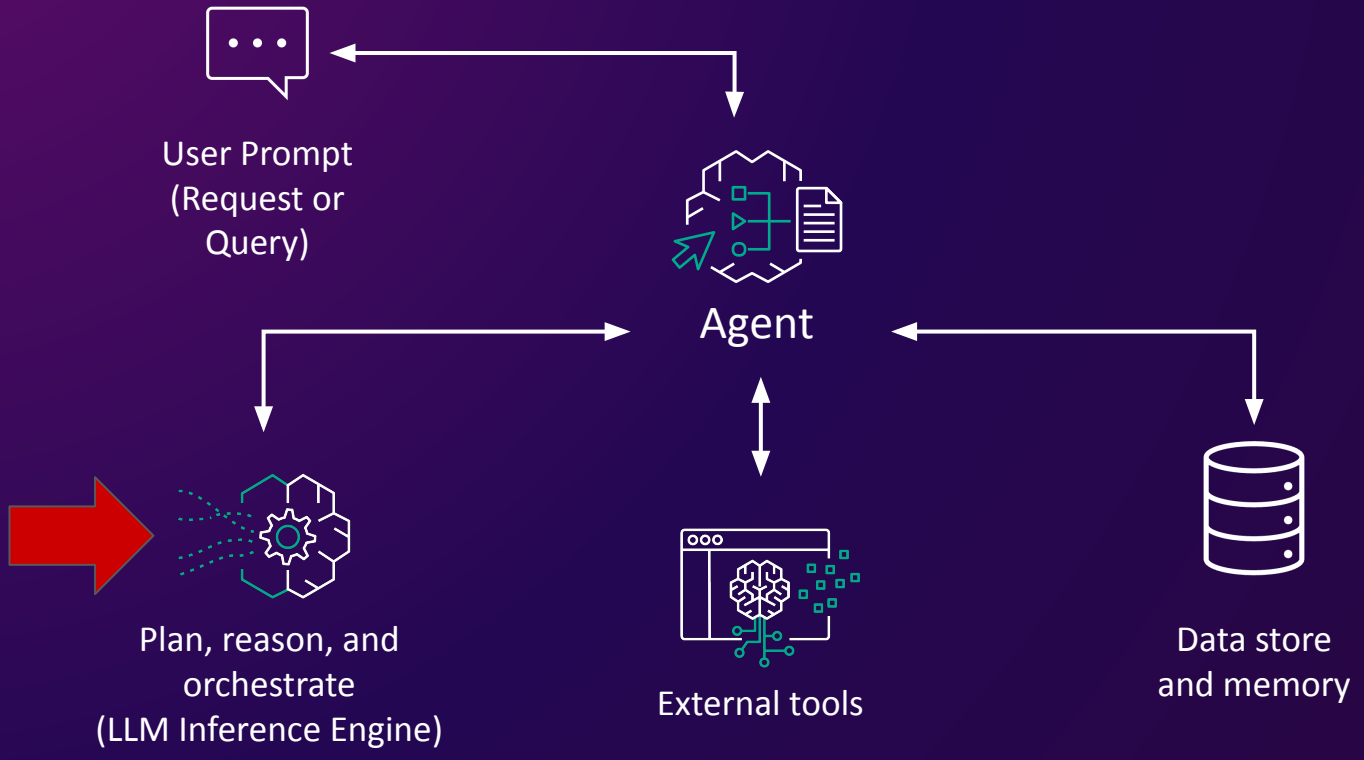
Serve more users with same hardware

GPU cost savings ($$$ => $)

Where are the bottlenecks?



Where are the bottlenecks that *I will cover today?*



Agenda

1. DeepSeek Inference Optimizations

2. vLLM and NVIDIA Dynamo Inference Engines

3. CUDA and PyTorch Optimizations for Modern GPU “Superchips” (NVIDIA Grace Blackwell and NVL72)

“Mechanical sympathy”

Original reference: Race car driver, Jackie Stewart, who was famously aware of the inner workings his race car.
-- Martin Thompson @ https://mechanical-sympathy.blogspot.com

Today’s computing reference: Co-designing software and algorithms hand-in-hand with hardware capabilities to maximize performance.

“Scarcity breeds innovation” - The DeepSeek Story

US Export Controls: The US government's restrictions on the export of advanced NVIDIA GPUs to China created a scarcity of top-tier hardware for companies like DeepSeek.

Adapting to Constraints: Scarcity required DeepSeek to develop new and highly efficient algorithms to squeeze the most out of older-generation GPUs.

Innovative Solutions: DeepSeek's approach led to breakthroughs like DeepSeek R1 which rivals leading frontier models from OpenAI and Anthropic, while requiring significantly less computing power and operational cost.

Scarcity fuels innovation by forcing teams to find creative solutions to limited resources.

DeepSeek Inference Optimizations

Optimized Transformer Attention Algorithm

Custom GPU-to-GPU Communication Library

Optimized Matrix Multiplication Library

Separate Parallelism Strategies for Prefill and Decode Phases

...

Distributed Inference Engine based on vLLM

REPOSITORY NAME	DESCRIPTION
FlashMLA	Efficient MLA decoding kernel for Hopper GPUs
DeepEP	Communication library for Mixture-of-Experts models
DeepGEMM	Optimized General Matrix Multiplication library
Optimized Parallelism Strategies	Framework for optimizing parallelism in distributed deep learning
Fire-Flyer File System (3FS)	Distributed file system optimized for machine learning workflows
DeepSeek-V3/R1 Inference System	Large-scale inference system using cross-node Expert Parallelism

DeepSeek Inference Optimizations *that I cover today*

Optimized Transformer Attention Algorithm

Custom GPU-to-GPU Communication Library

Optimized Matrix Multiplication Library

Separate Parallelism Strategies for Prefill and Decode Phases

...

Distributed Inference Engine based on vLLM

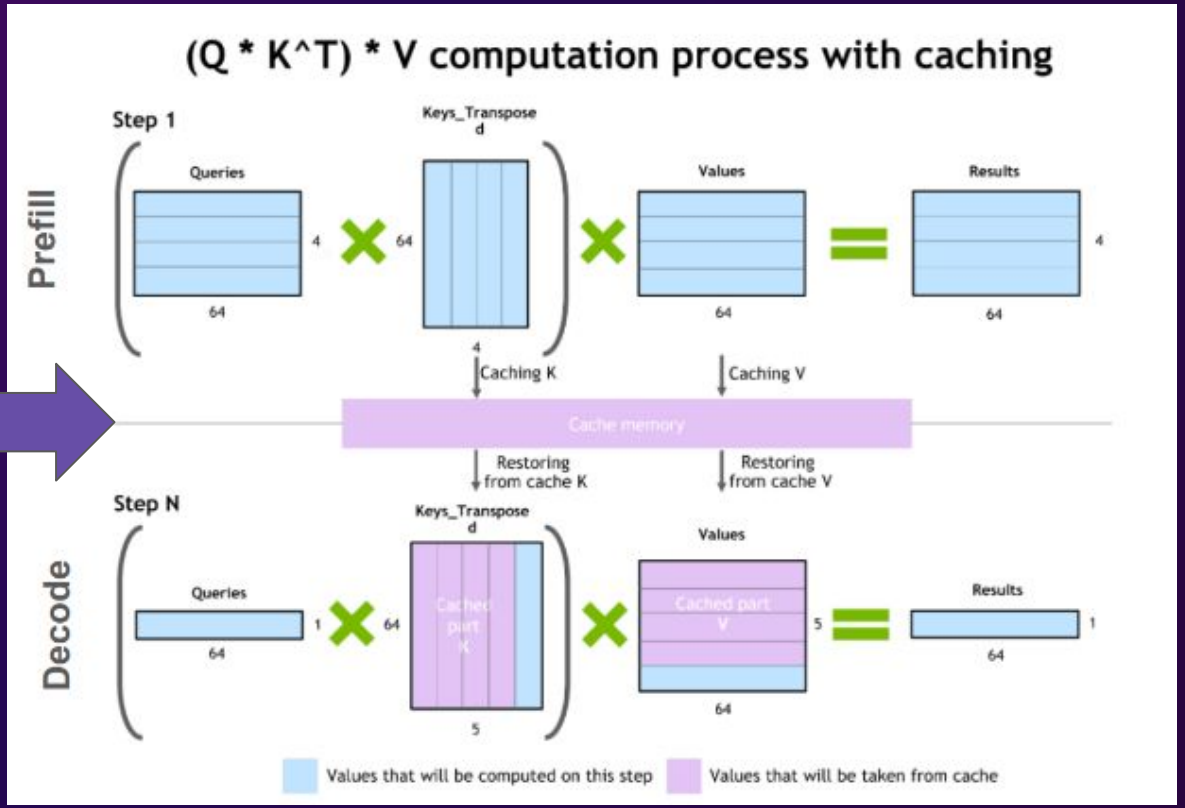
REPOSITORY NAME	DESCRIPTION
FlashMLA	Efficient MLA decoding kernel for Hopper GPUs
DeepEP	Communication library for Mixture-of-Experts models
DeepGEMM	Optimized General Matrix Multiplication library
Optimized Parallelism Strategies	Framework for optimizing parallelism in distributed deep learning
Fire-Flyer File System (3FS)	Distributed file system optimized for machine learning workflows
DeepSeek-V3/R1 Inference System	Large-scale inference system using cross-node Expert Parallelism

DeepSeek's Hybrid Parallelism Strategy for Prefill and Decode

Prefilling Phase
- Transformer's Attention Mechanism
- Populates Key-Value (KV) Cache
- Compute FLOPs Heavy

Key-Value (KV) Cache
(Shared by both phases)

Decoding Phase
- Generates token-by-token
- "Autoregressive"
- Memory-bandwidth heavy

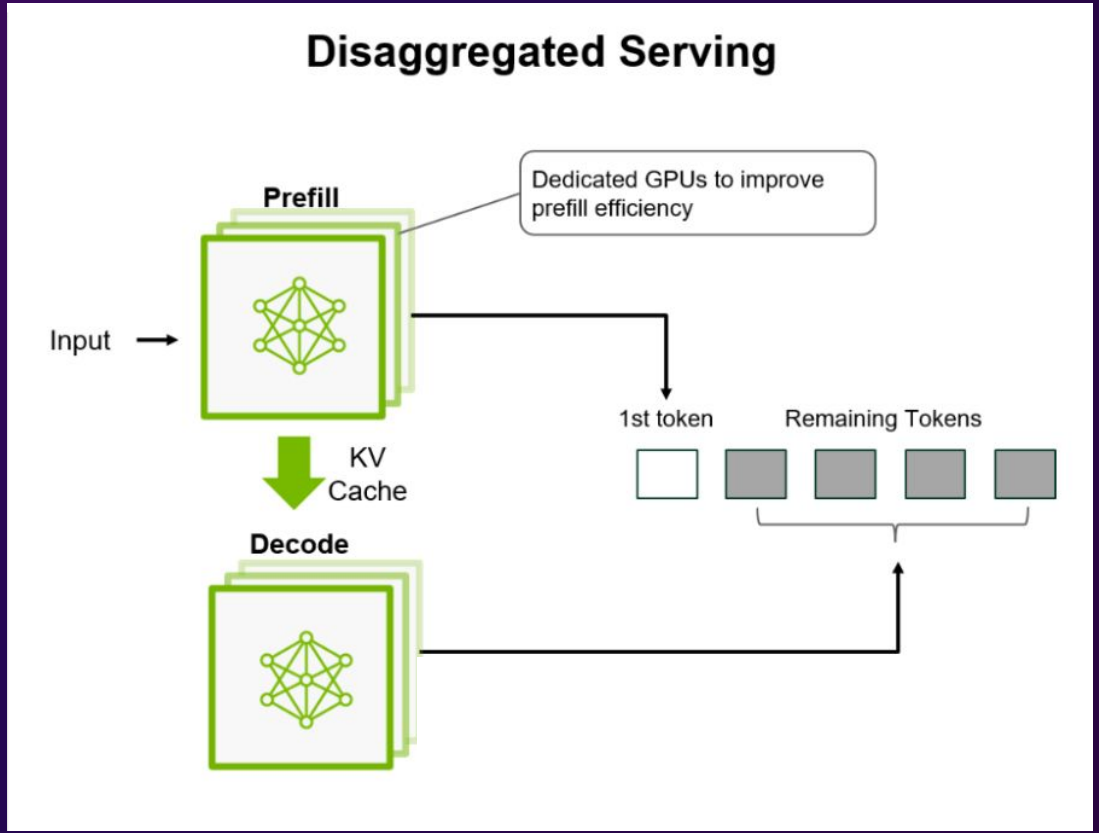


DeepSeek's Hybrid Parallelism Strategy for Prefill and Decode

Prefilling Phase
- Transformer's Attention Mechanism
- Populates Key-Value (KV) Cache
- Compute FLOPs Heavy

Key-Value (KV) Cache
(Shared by both phases)

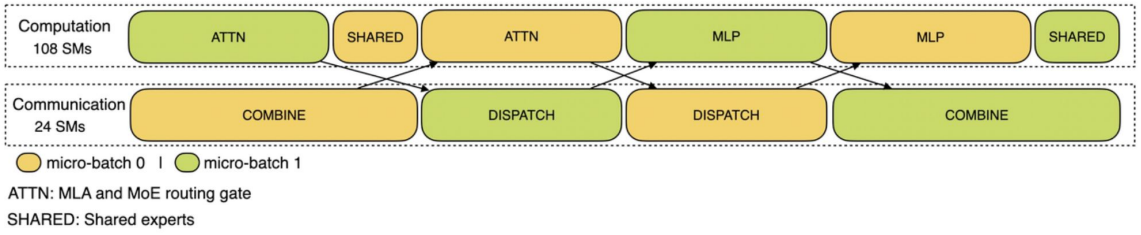
Decoding Phase
- Generates token-by-token
- "Autoregressive"
- Memory-bandwidth heavy



DeepSeek's Hybrid Parallelism Strategy for Prefill and Decode

Prefilling Phase
[Routed Expert EP32, MLA/Shared Expert DP32]
Each deployment unit spans 4 nodes with 32 redundant routed experts, where each GPU handles 9 routed experts and 1 shared expert.

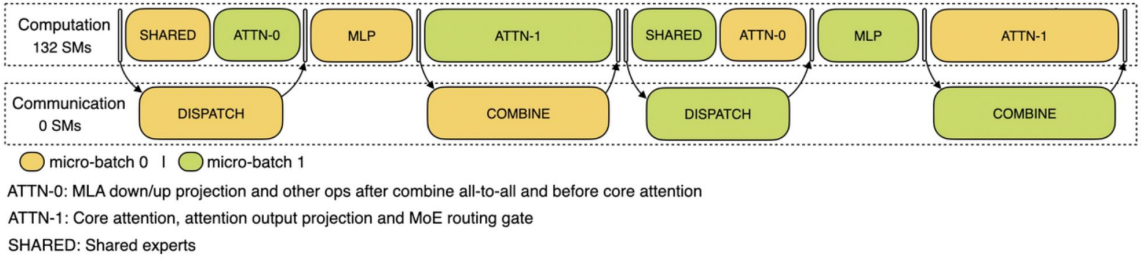
Large-scale cross-node EP introduces significant communication overhead. To mitigate this, we employ a dual-batch overlap strategy to hide communication costs and improve overall throughput by splitting a batch of requests into two microbatches. During the prefilling phase, these two microbatches executed alternately and the communication cost of one microbatch is hide behind the computation of the other.



Communication-Computation Overlapping during Prefilling Phase

Decoding Phase
[Routed Expert EP144, MLA/Shared Expert DP144]
Each deployment unit spans 18 nodes with 32 redundant routed experts, where each GPU manages 2 routed experts and 1 shared expert.

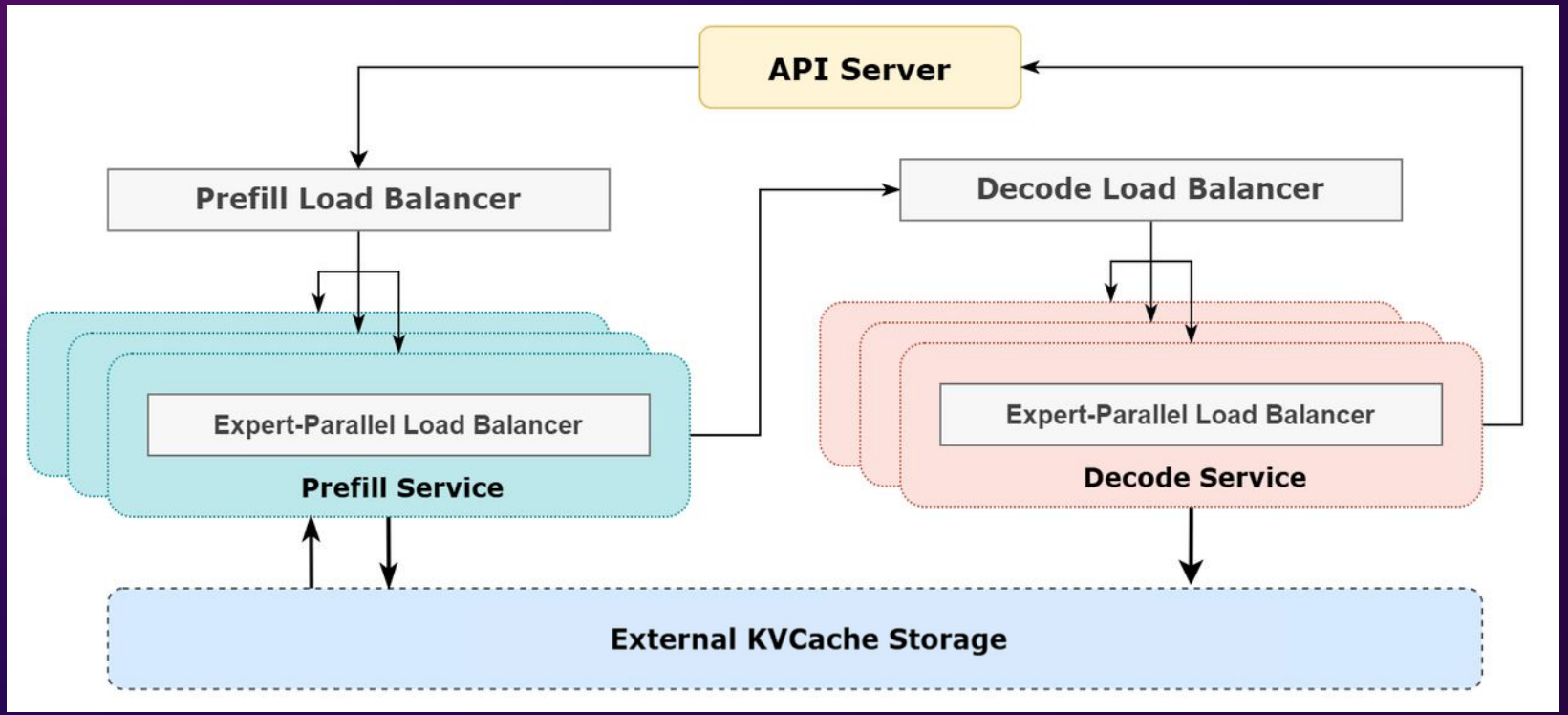
During the decoding phase, the execution durations of different stages are unbalanced. Hence, we subdivide the attention layer into two steps and use a 5-stage pipeline to achieve a seamless communication-computation overlapping.



Communication-Computation Overlapping during Decoding Phase

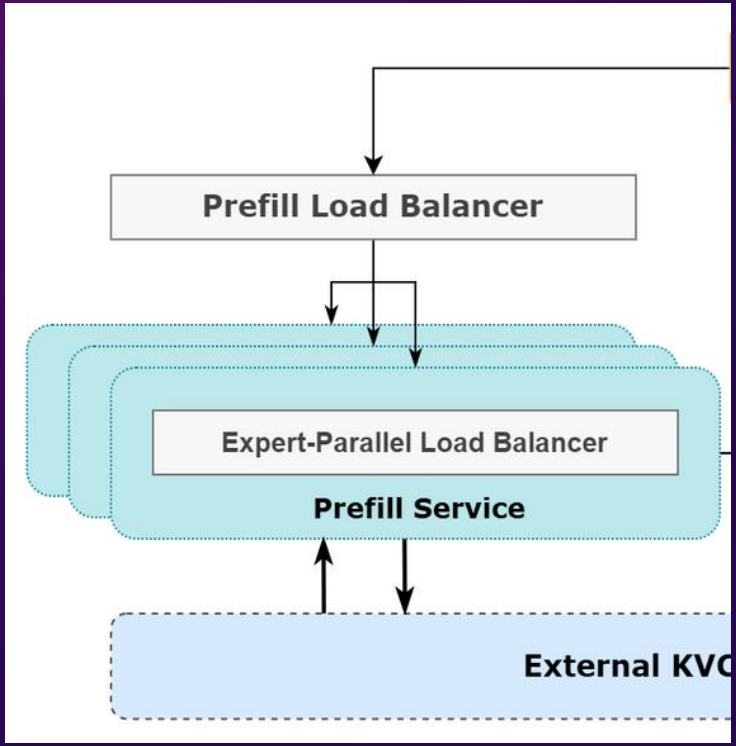
DeepSeek's Distributed Inference Engine

Goal: Maximize resource utilization by balancing computational and communication loads across prefill and decode phases - and across all GPUs.



DeepSeek's Distributed Inference Engine

Goal: Maximize resource utilization by balancing computational and communication loads across prefill and decode phases - and across all GPUs.



Optimization Goals of Prefill Phase

Balance Attention computation across GPUs

Equalize input token counts per GPU

Prevent prolonged processing on specific GPUs

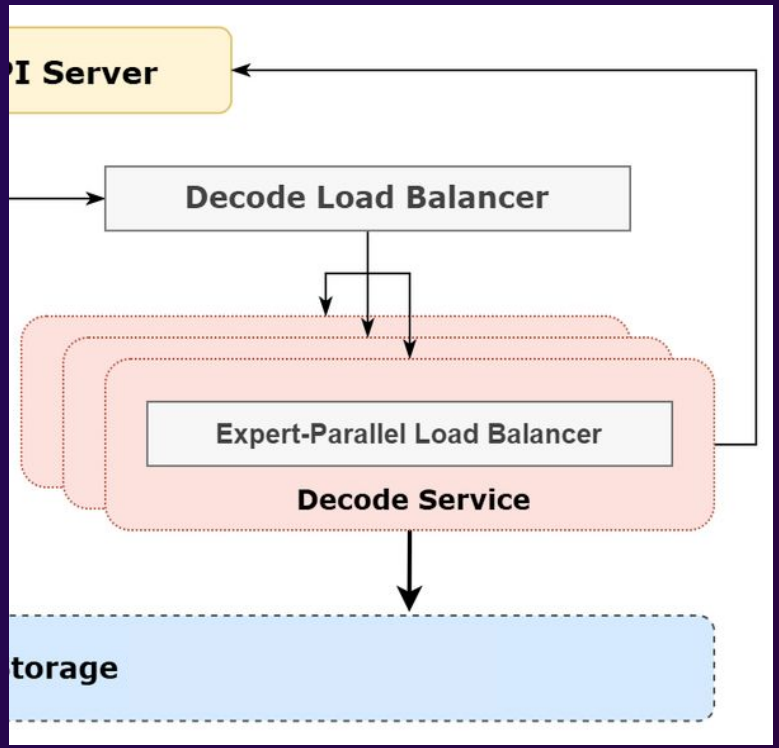
DeepSeek's Distributed Inference Engine

Goal: Maximize resource utilization by balancing computational and communication loads across prefill and decode phases - and across all GPUs.

Optimization Goals of Decode Phase

Balance KV Cache usage across GPUs

Equalize request counts per GPU

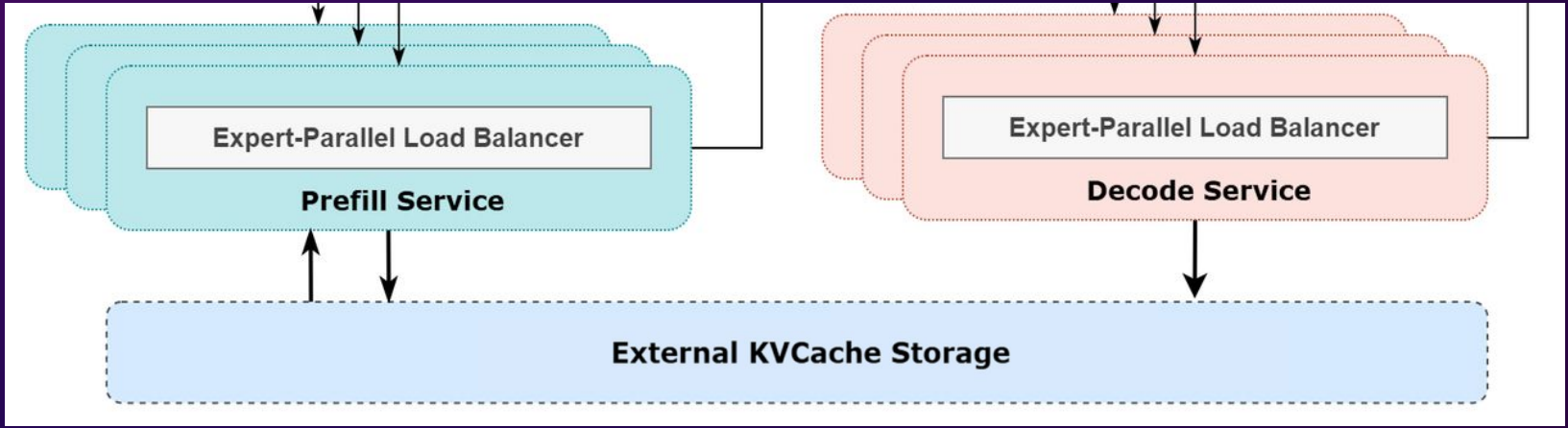


DeepSeek's Distributed Inference Engine

Goal: Maximize resource utilization by balancing computational and communication loads across prefill and decode phases - and across all GPUs.

Optimization Goals of Expert-Parallel Load Balancers

Balance expert computation across GPUs



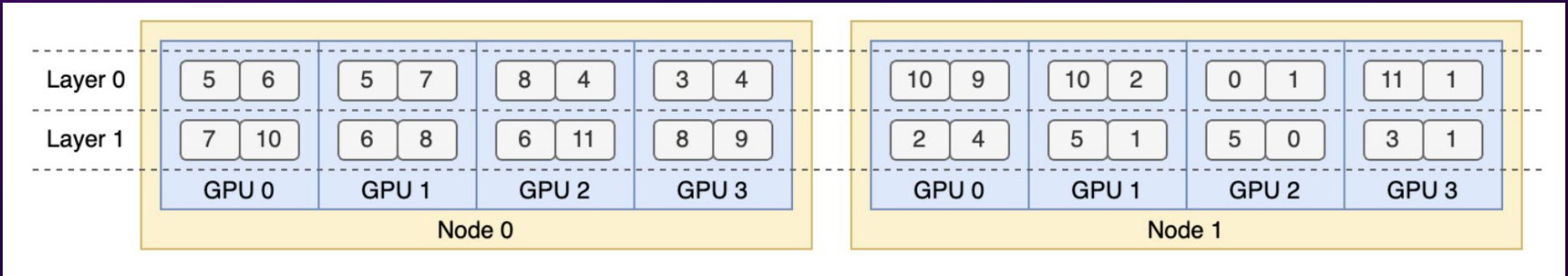
DeepSeek's Distributed Inference Engine

Goal: Maximize resource utilization by balancing computational and communication loads across prefill and decode phases - and across all GPUs.

Optimization Goals of Expert-Parallel Load Balancers

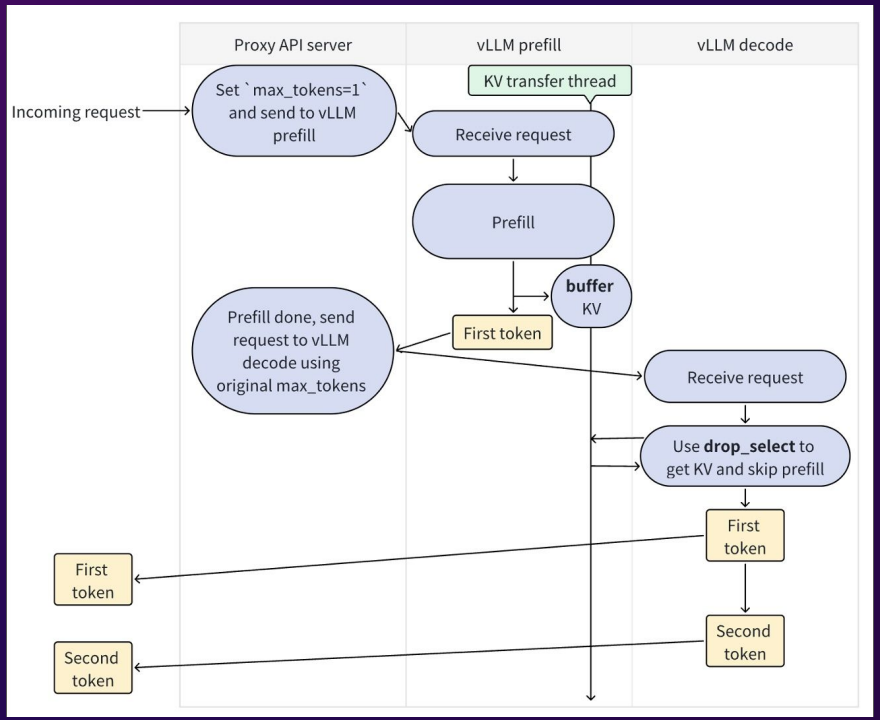
Balance expert computation across GPUs

Redundant experts across GPUs on same node to avoid inter-node communication

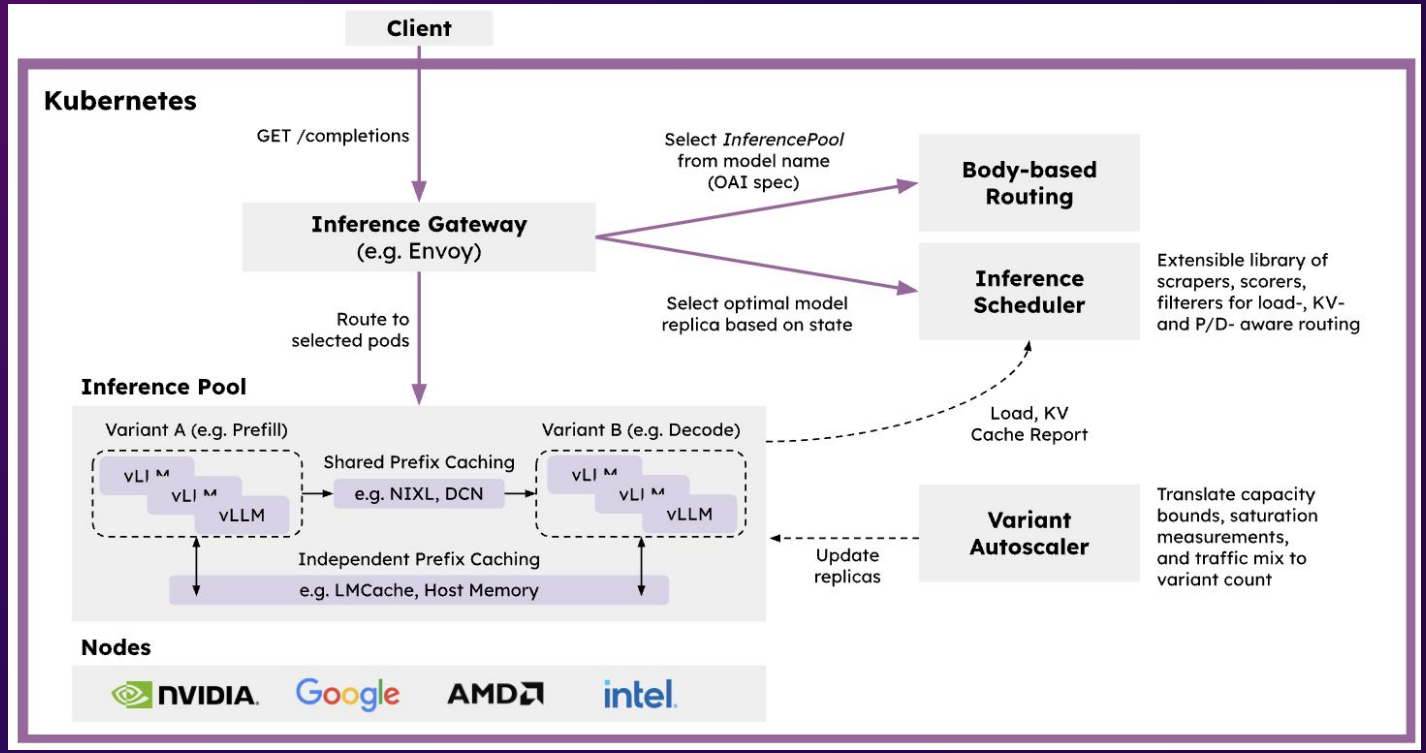


DeepSeek's Distributed Inference Engine is based on vLLM

Goal: Maximize resource utilization by balancing computational and communication loads across prefill and decode phases - and across all GPUs.

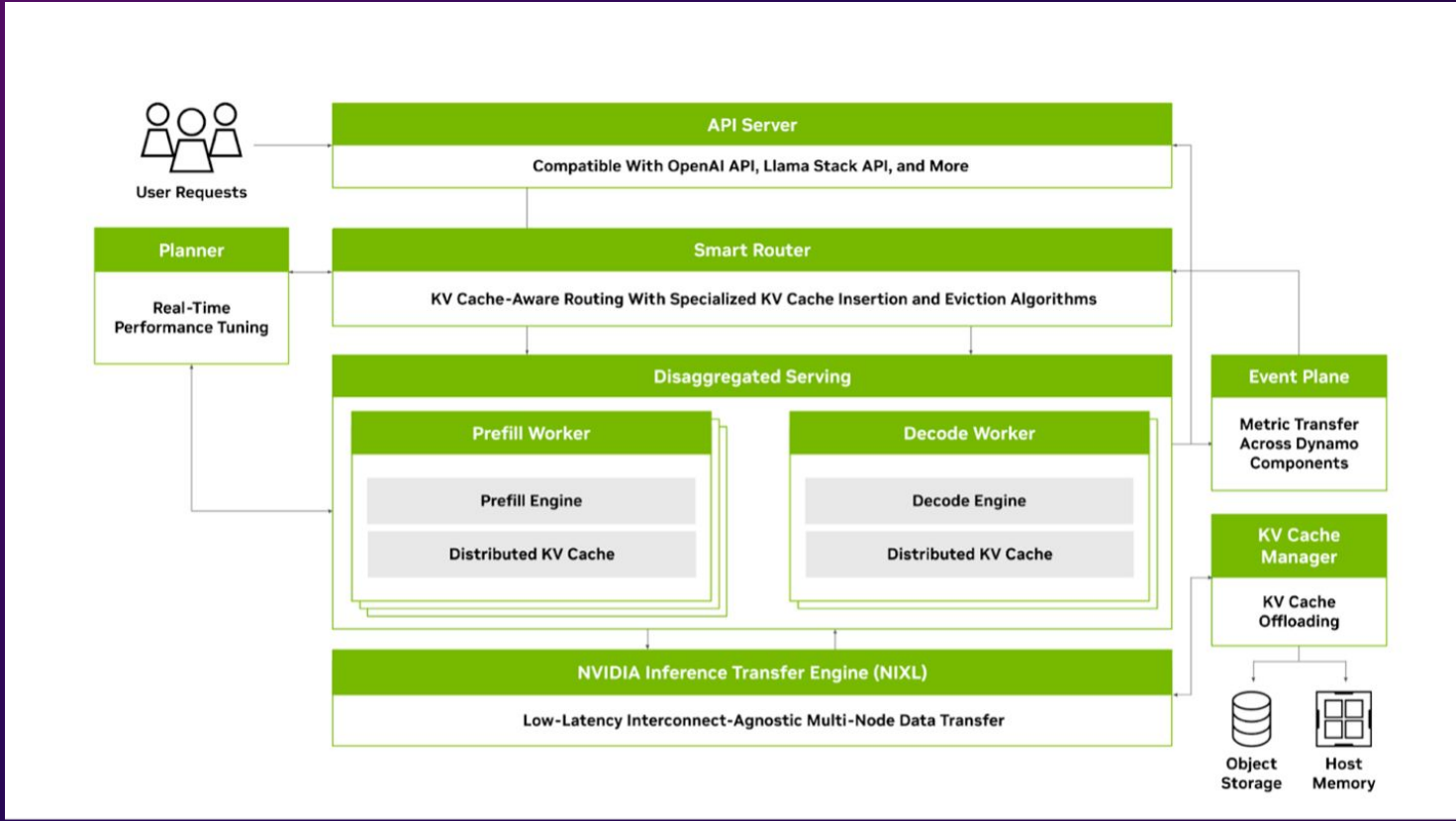


Check out the distributed vLLM on Kubernetes project (llm-d)



https://github.com/llm-d/llm-d

Another option: NVIDIA Dynamo with vLLM backend



Mechanical sympathy terms for modern AI systems

Disaggregated prefill-decode and independent scaling

› Scaling the prefill and decode phases independently since they have different compute and memory-bandwidth characteristics

Arithmetic intensity of an algorithm (or CUDA kernel)

› Ratio of FLOPs performed per bytes of memory moved.

› The goal is to increase arithmetic intensity if possible since GPUs have high FLOPs and low memory bandwidth.

Speculative decoding and multi-token prediction (MTP)

› LLM inference engine generates (a.k.a decodes) more than 1 token at a time to better-utilize the high FLOPs capacity of modern GPUs and **increase arithmetic intensity**

Dynamic and continuous batching

› LLM inference engine batches up multiple requests across many end users to increase the GPU compute FLOPs per byte of data movement and **increase arithmetic intensity**

Co-designed NVIDIA GPU and CUDA optimizations

Tensor memory accelerator (TMA)

› Physical hardware on the GPU that offloads memory-movement from the GPUs “cores” called Streaming Multiprocessors (SMs).

› Frees up the SMs to performance useful work instead of moving memory.

Faster interconnects between CPU⇔GPU and GPU⇔GPU for unified memory

› Fast NVLink interconnects are preferred over slow, traditional PCIe interconnects.

› Allows seamless unified memory between large CPU DRAM and small GPU VRAM

› Grace-Blackwell (CPU-GPU) “Superchips” use fast, coherent, unified CPU-GPU memory

Thread-block clusters

› Groups of threads - running across multiple SMs - work together on large matrices

› Modern GPUs provide distributed shared-memory and native synchronization primitives

DeepSeek's clever GPU on-chip L1/L2 cache trick

GPUs have a well-defined memory hierarchy

› SM registers => L1 cache => L2 cache => VRAM global memory

DeepSeek's DeepEP library uses a low-level trick to better utilize L1, L2, VRAM

› PTX (GPU virtual instruction set): **LD.Global.NC.L1::no_allocate.L2::256b**

› Meaning: when accessing VRMA global memory, bypass L1 cache and pull 256-byte chunks directly into L2

Cleverness

› Reduces cache-pollution overhead, boosts L2 bandwidth, and shrinks global-load latency

› Produces significant performance wins for memory-bandwidth–bound kernels common in LLMs

› Ideal for read-only access patterns such as large Transformer- model weight loads, Attention KV cache fetches since L1 cache reuse is minimal

Co-designed GPU, CUDA, and PyTorch optimizations

PyTorch Compiler (torch.compile)

› Generates fused CUDA kernels to increase arithmetic intensity (FLOPs per byte moved)

› Optimizes GPU register usage through loop unrolling and other compiler tricks

PyTorch support for CUDA Graphs (multi-kernel graphs of execution)

› Reduces CUDA kernel launch overhead

› Allows predefined graphs of execution for different inference batch sizes (1, 4, 8, 16, etc)

PyTorch support for reduced precision to better-utilize GPU Tensor Cores

› TF32 (10x performance of FP32), FP16, FP8, FP4 (FP4 support coming soon to PyTorch!)

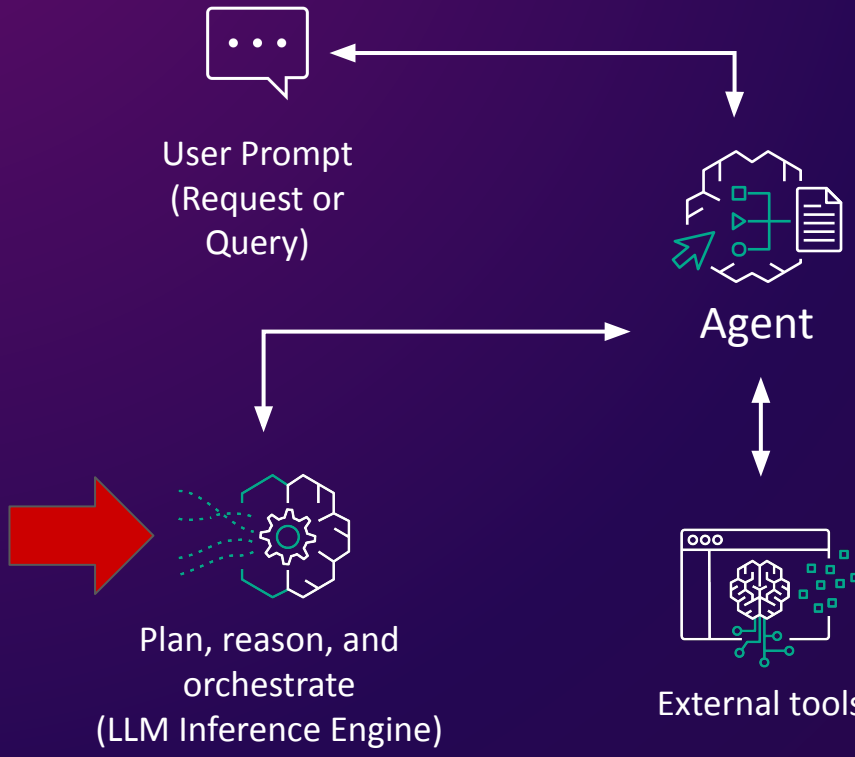
› Modern GPUs provide distributed shared-memory and native synchronization primitives

GPU Tensor Cores can multiple instructions concurrently in a single cycle

› TF32, FP16, FP8, and FP4 instructions can run in parallel in a single cycle

› Even faster when combined with Tensor Memory Accelerator (TMA) for data movement

Conclusion: faster LLM inference means faster agents



Faster time-to-response

Better end user experience

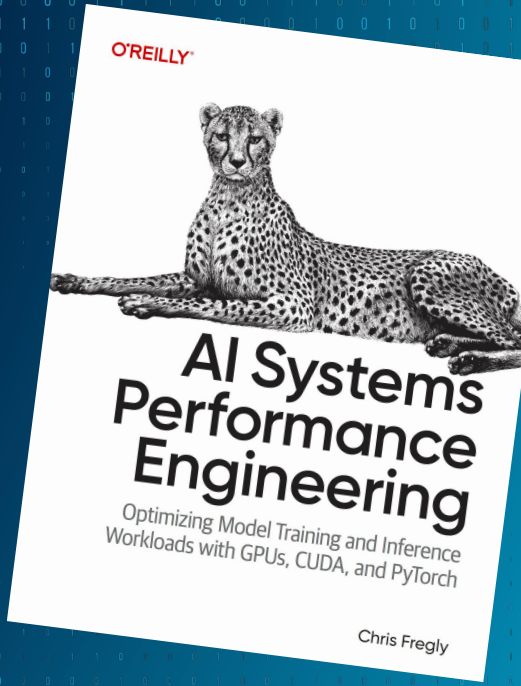
Serve more users with same hardware

GPU cost savings ($$$ => $)

Thank You!

High Performance Agentic AI Inference Systems

Chris Fregly



https://www.amazon.com/Systems-Performance-Engineering-Optimizing-Inference/dp/B0F47689K8/