Inference at Scale: Opportunities and challenges

Vikram Sharma Mailthody
Sr. Research Scientist, NVIDIA Research

Disclaimer

“

The views expressed in the content belong to me and my collaborators, not NVIDIA, its affiliates, or employees

Vikram Sharma Mailthody

”

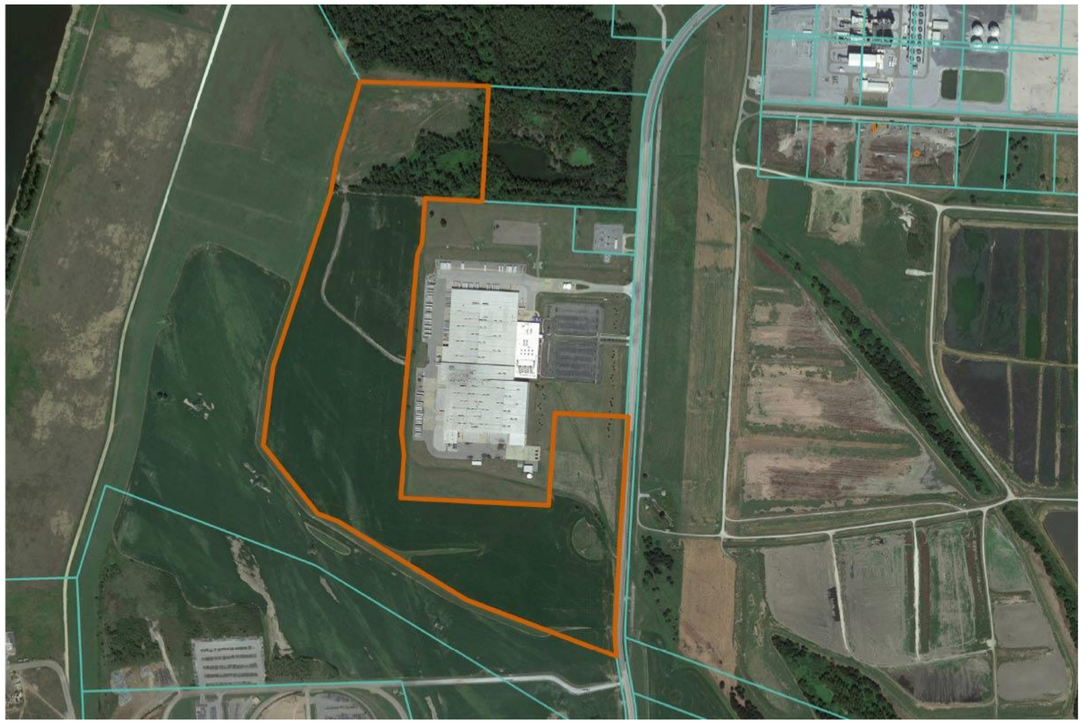
Lecture Overview

- Vacuums to tokens
  - Intro to modern day AI datacenters
  - AI factories
    - Datacenter architecture
    - System architecture
  - Open challenges
- AI inference
  - How it works
  - KV caching
- Inference at scale using Dynamo
  - Aggregated approach
  - Disaggregated Inference
  - KV aware router, Planner, and others
  - Open challenges
    - Fault tolerance
    - Load balancing

Credits: Wen-mei Hwu, Junda Chen

AI Today

Vacuums to tokens in <122 days ☺

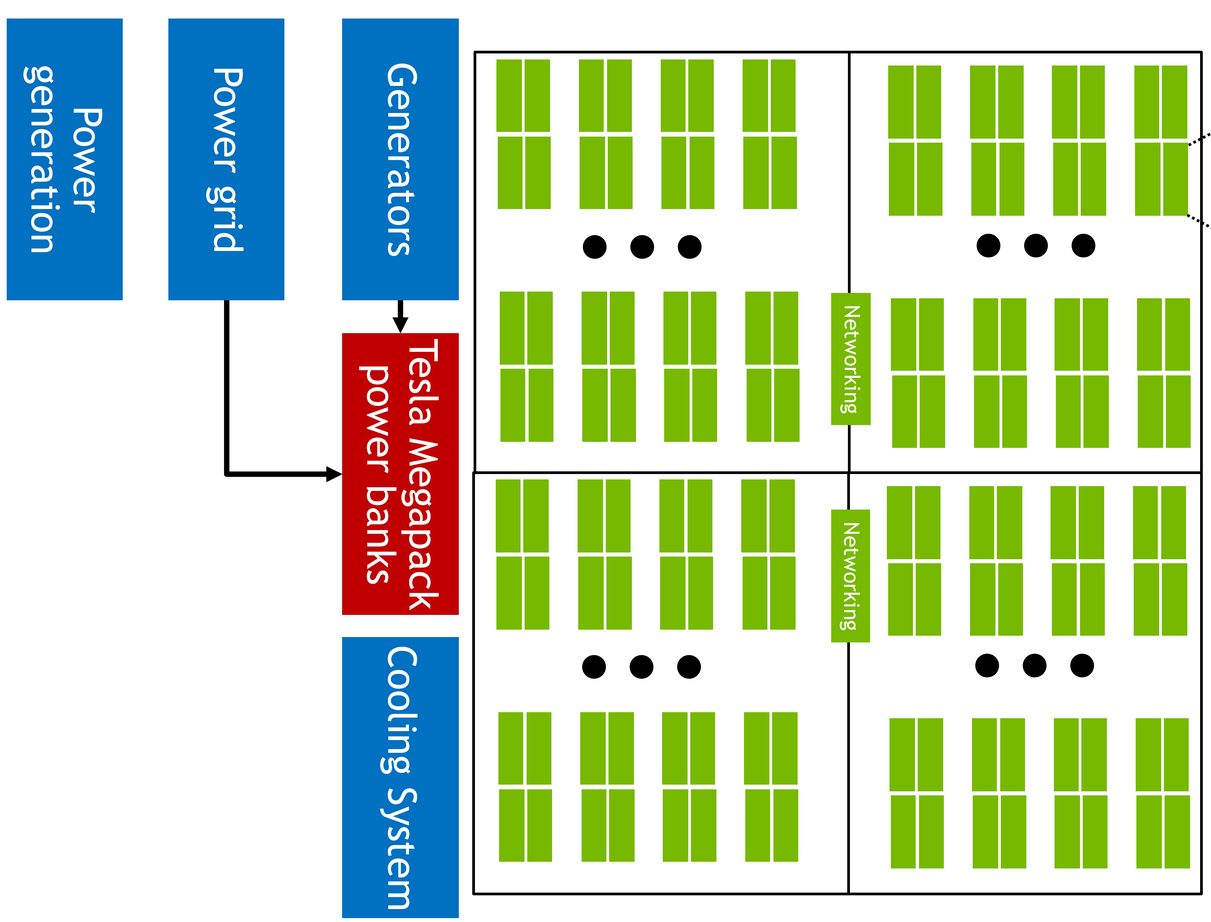


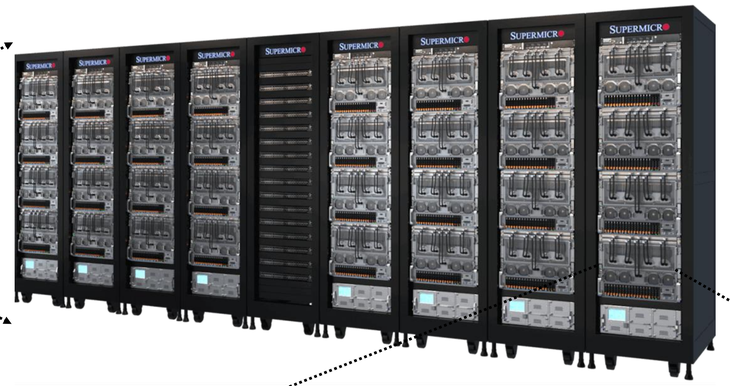
xAI 100K Cluster → 200K soon

>150MW Datacenter

AI factory - xAI 100K Cluster → 200K → ???

4 data halls, each data hall is 25K GPUs, liquid cooled racks, each rack with 64 GPUs





4 to 8 DGXH100 Per rack

DGXH100 with 8 GPU

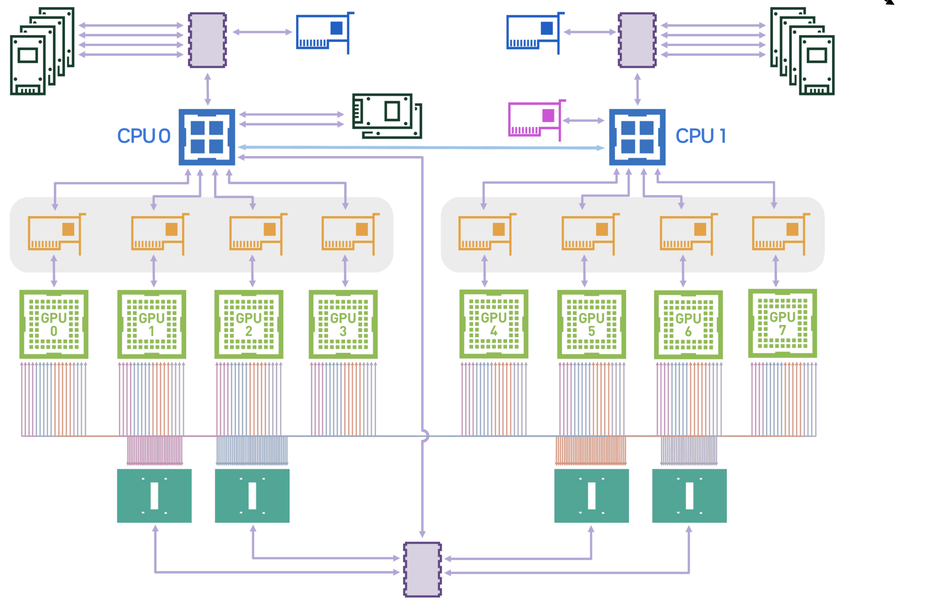


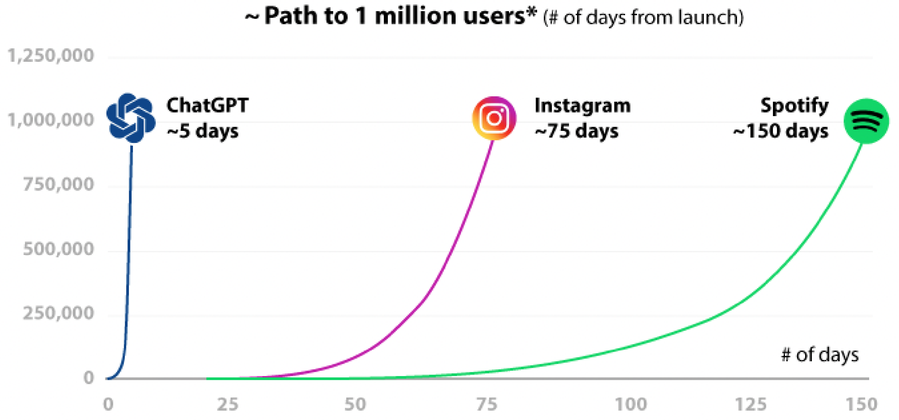
*Figure for illustration purposes only and is not accurate

https://www.servethehome.com/inside-100000-nvidia-gpu-xai-colossus-cluster-supermicro-helped-build-for-elon-musk/

AI Success

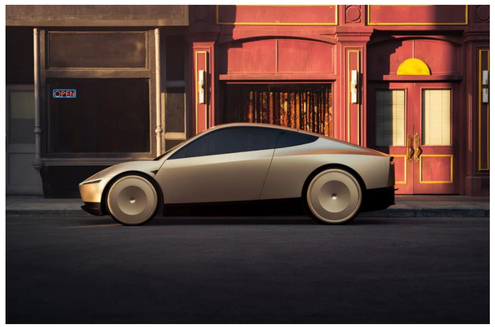
Industry transformation is happening

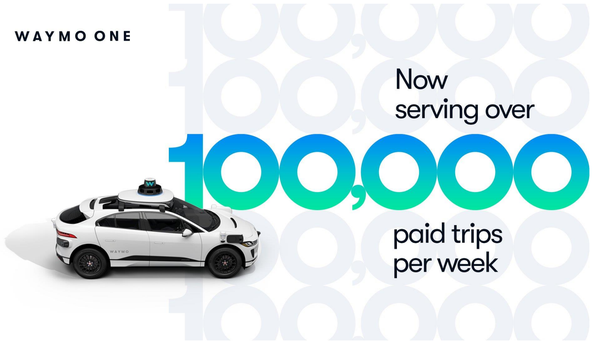
Chatbots

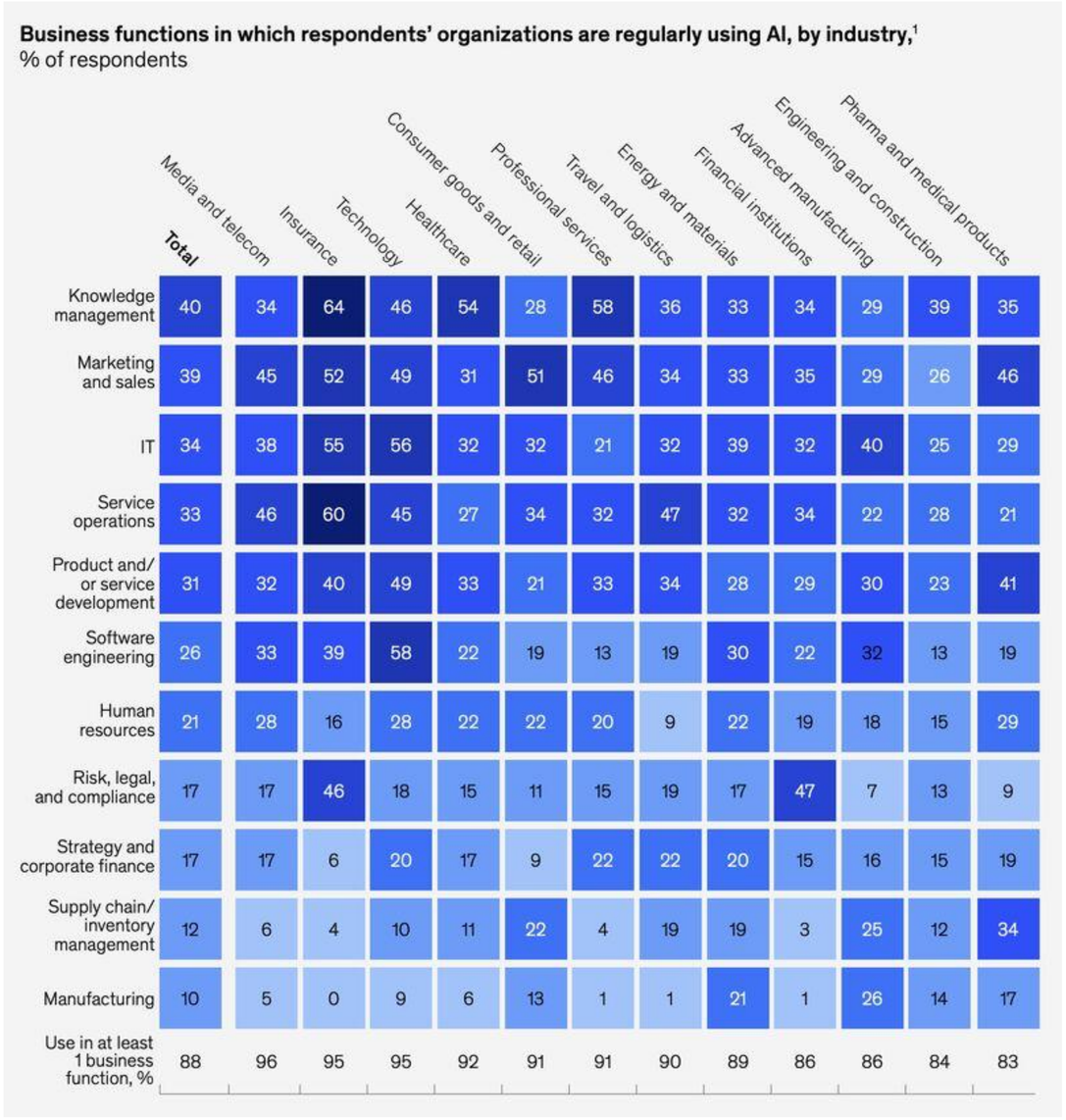


Sources: Google, Subredditstats, Media Reports

Self driving cars





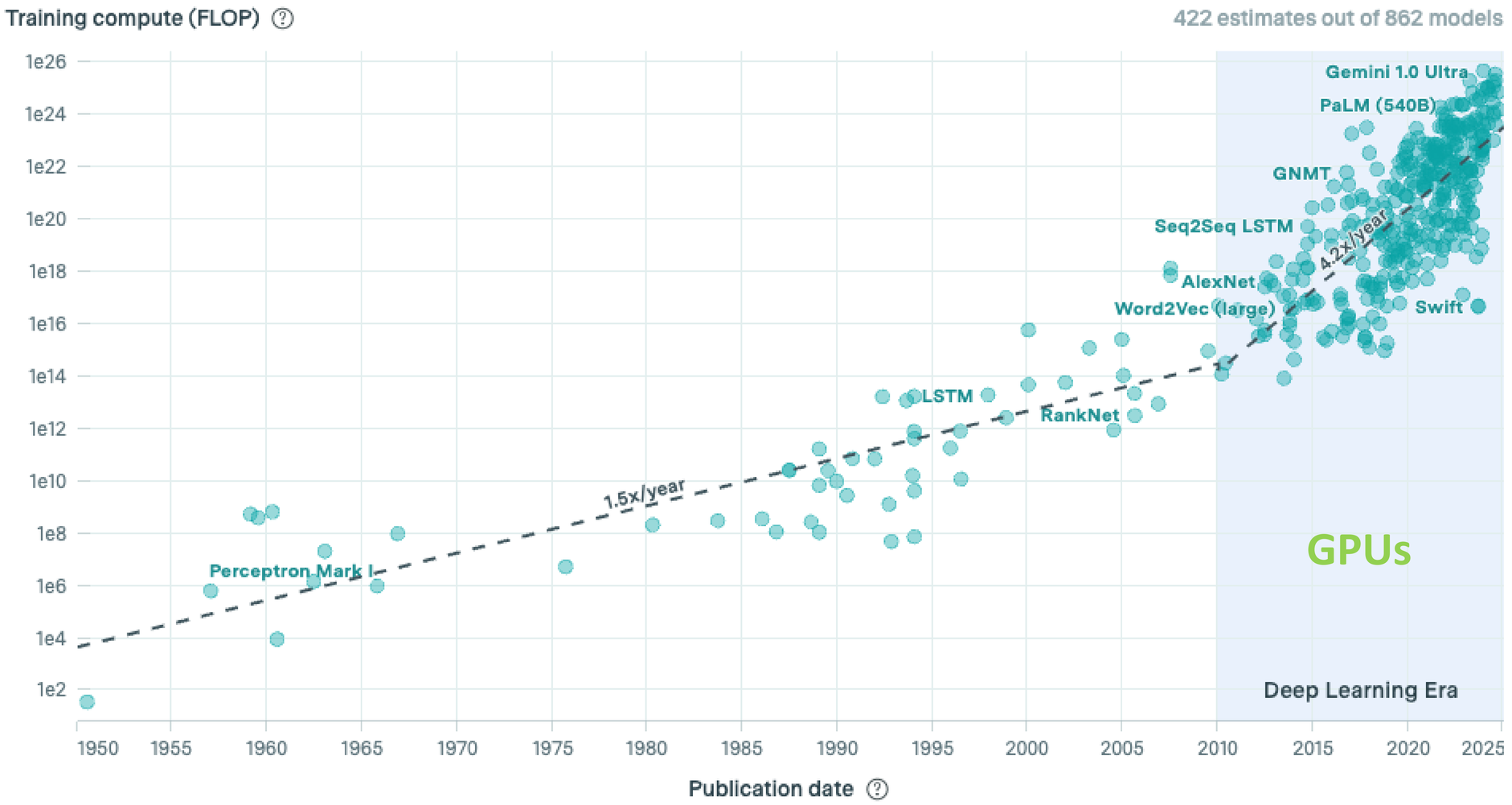


https://www.mckinsey.com/capabilities/quantumblack/our-insights/the-state-of-ai#/

NVIDIA

AI Flops outpacing Moore's law

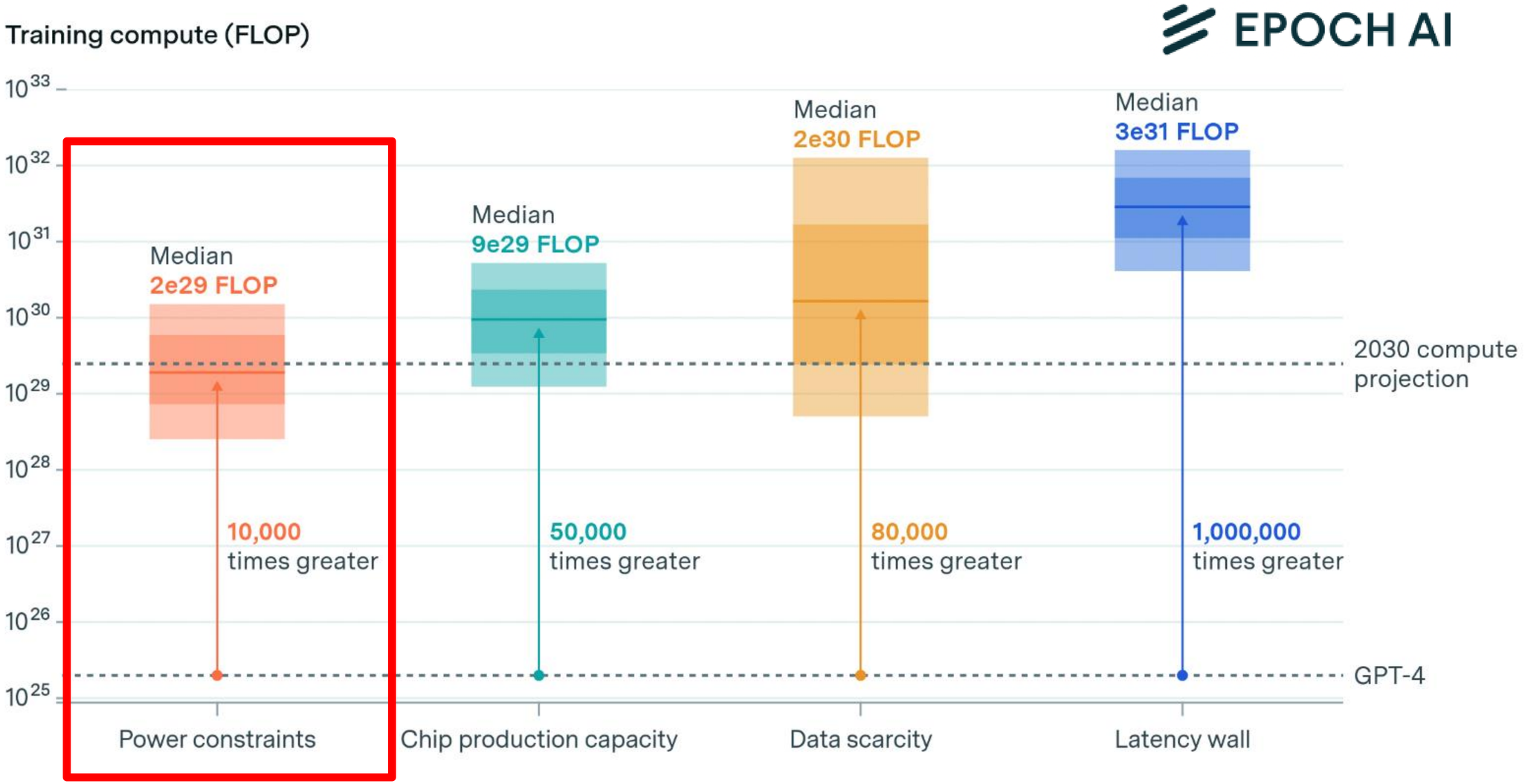
4.2X flops growth per year for LLM models. What about inference needs???



https://epochai.org/blog/compute-trends

Power is THE primary limiter for AI training beyond 2030

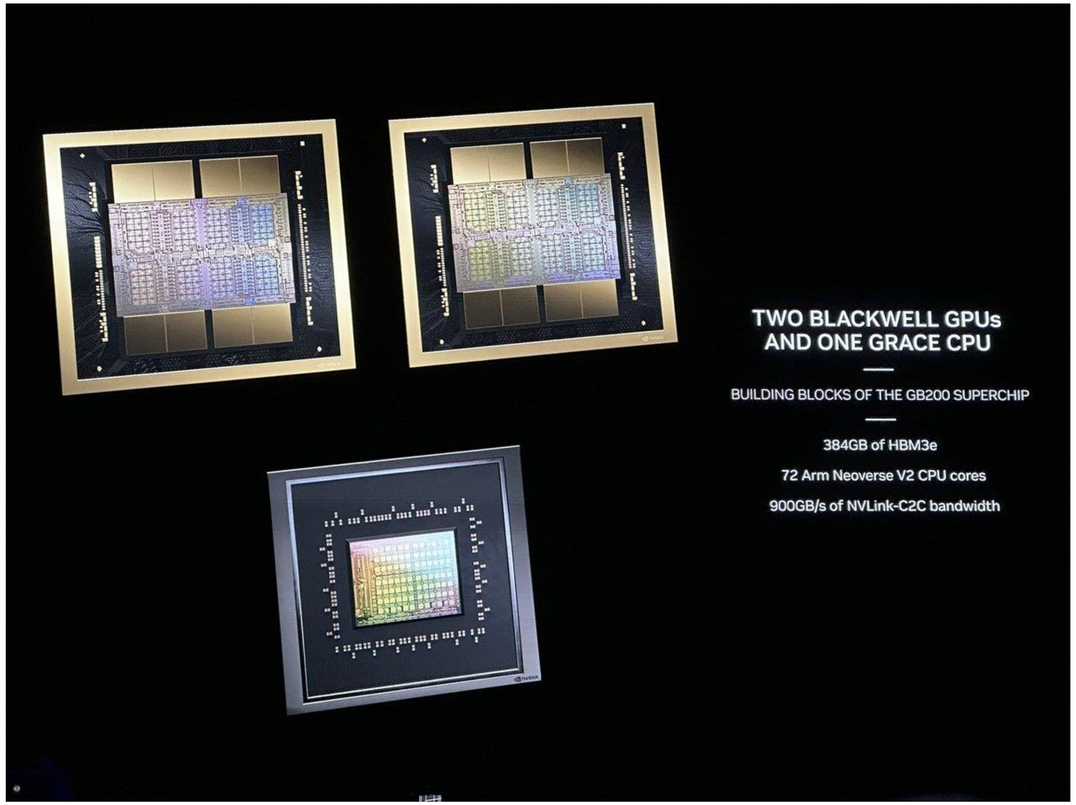
Need to invest in power efficiency AND revamping power generation and delivery



https://epochai.org/blog/can-ai-scaling-continue-through-2030

NVL72 GB200 Pods

For power efficiency and higher performance

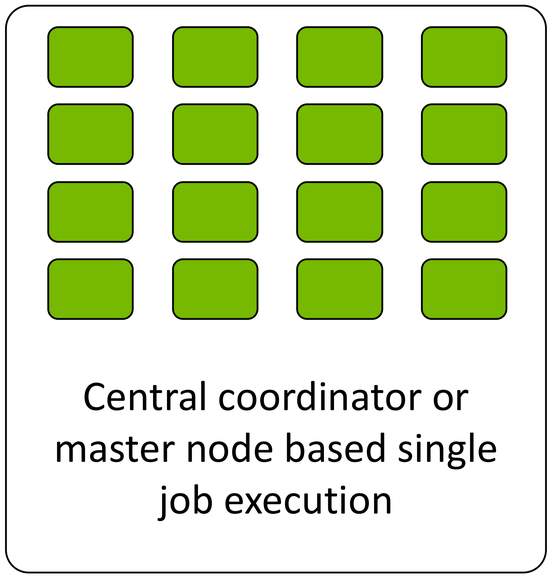


AI Inference

AI Inference

Different beast compared to AI Training

Parallelism strategies
Failure and resiliency
Weak and strong scaling
Topology awareness
Preprocessing data
Optimizers
….

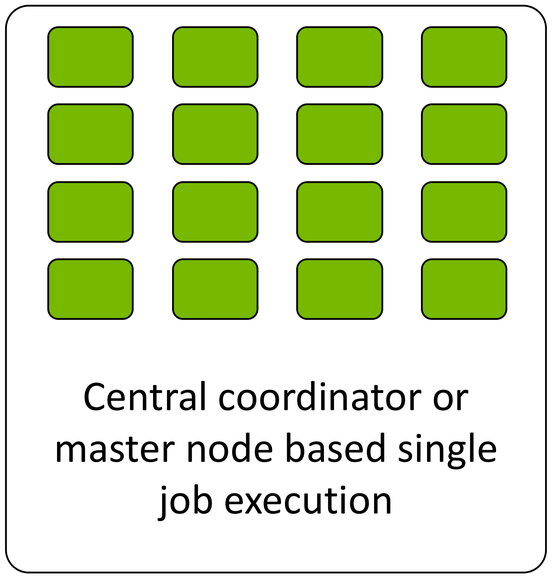


LLM Training

AI Inference

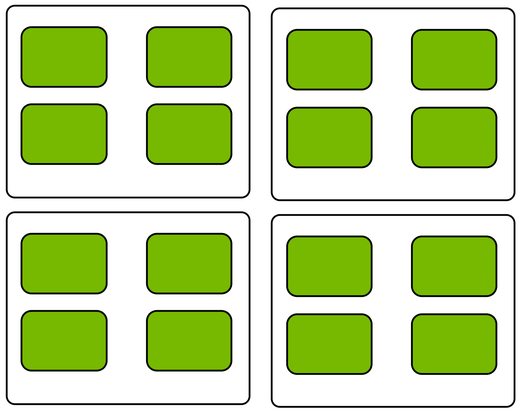
Different beast compared to AI Training

Parallelism strategies
Failure and resiliency
Weak and strong scaling
Topology awareness
Preprocessing data
Optimizers
….



Central coordinator or master node based single job execution

LLM Training



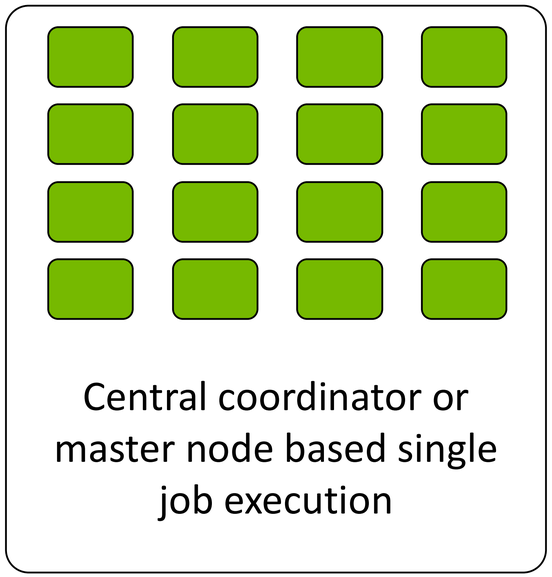
Several small jobs with rapid scale up and down requirements

LLM Inference

AI Inference

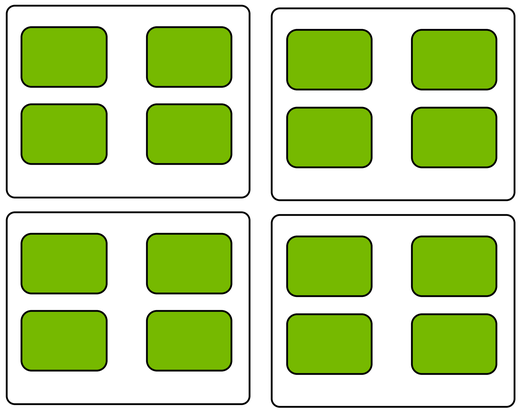
Different beast compared to AI Training

Parallelism strategies
Failure and resiliency
Weak and strong scaling
Topology awareness
Preprocessing data
Optimizers
….



Central coordinator or master node based single job execution

LLM Training



Several small jobs with rapid scale up and down requirements

LLM Inference

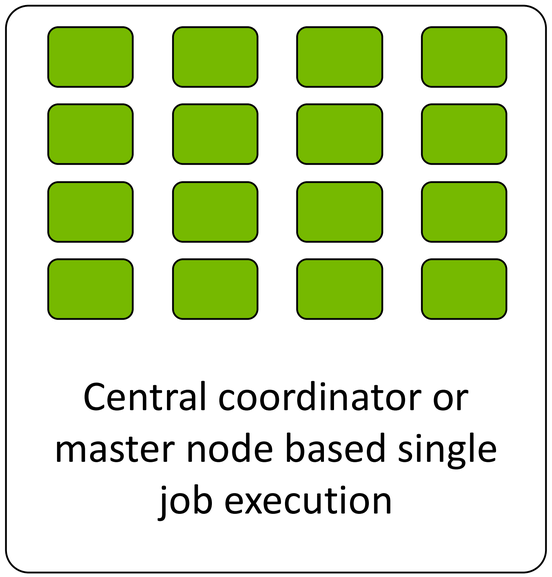
Inflight batching
KV cache optimization
Quantization
Parallelism strategies
User-defined scheduling
…

- Unique challenges and lot of pending optimizations in both stack

AI Inference

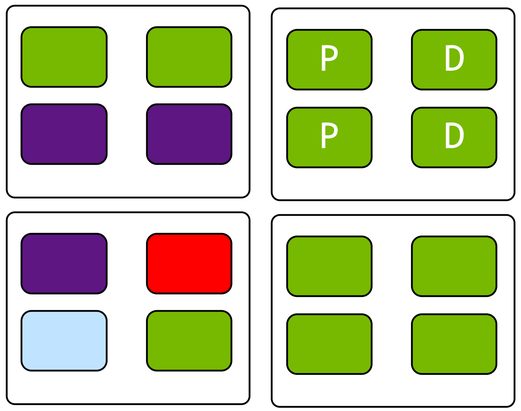
Different beast compared to AI Training

Parallelism strategies
Failure and resiliency
Weak and strong scaling
Topology awareness
Preprocessing data
Optimizers
….



Central coordinator or master node based single job execution

LLM Training



Several small jobs with rapid scale up and down requirements

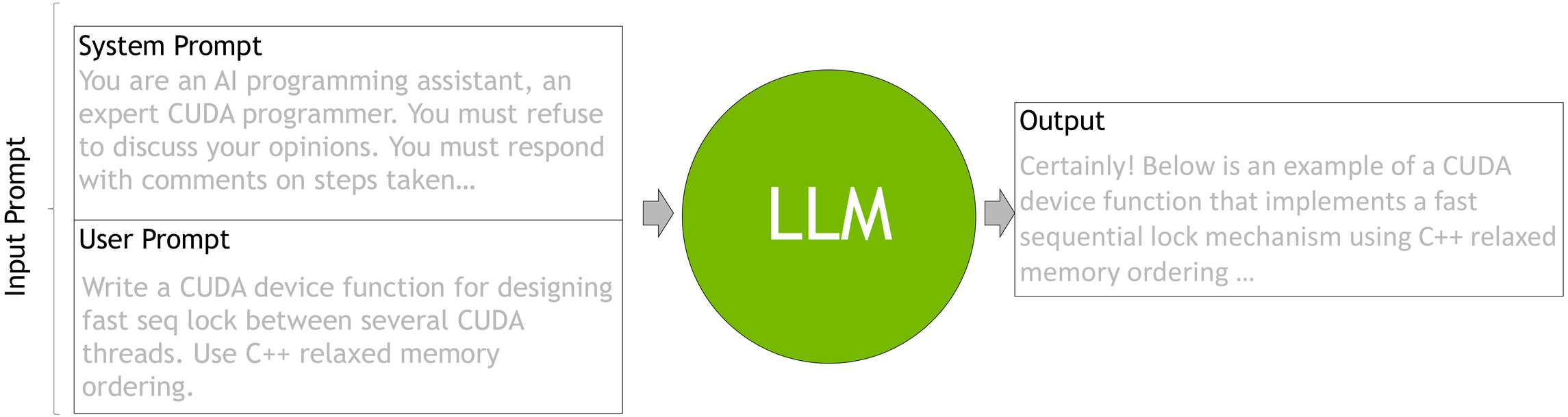
LLM Inference

Inflight batching
KV cache optimization
Quantization
Parallelism strategies
User-defined scheduling
…

- Unique challenges and lot of pending optimizations in both stack

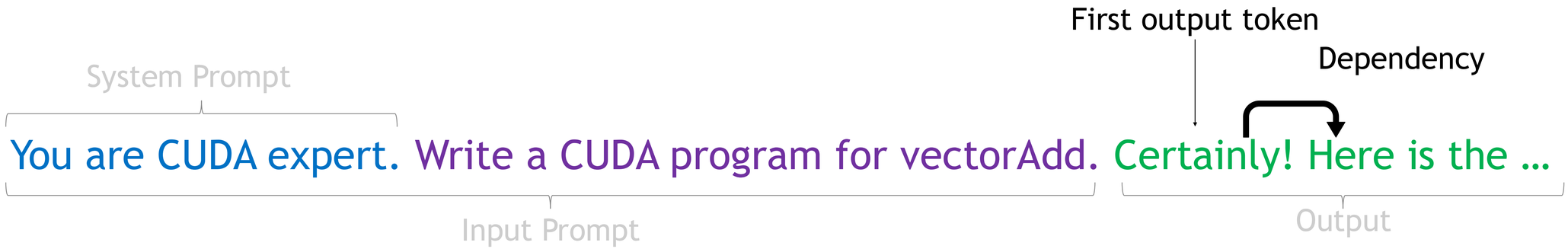
LLM Inference

Typical workflow of LLM inference



- System prompts are instructions to language model that constraints the output generation in a specified manner
- System prompts are common across several user prompts (from the same user or different set of users)
- Input prompt = system + user prompt and can consume even upto 32K tokens.
- Trend is towards longer prompts with ”prompt rules” → leads to better quality response from the model.
- Input prompt processing accounts for a significant-to-dominant fraction of the overall processing time

LLM Inference



$$P(x) = P(x_1).P(x_2|x_1) \dots P(x_n|x_1, \dots x_{n-1})$$

Autoregressive decomposition

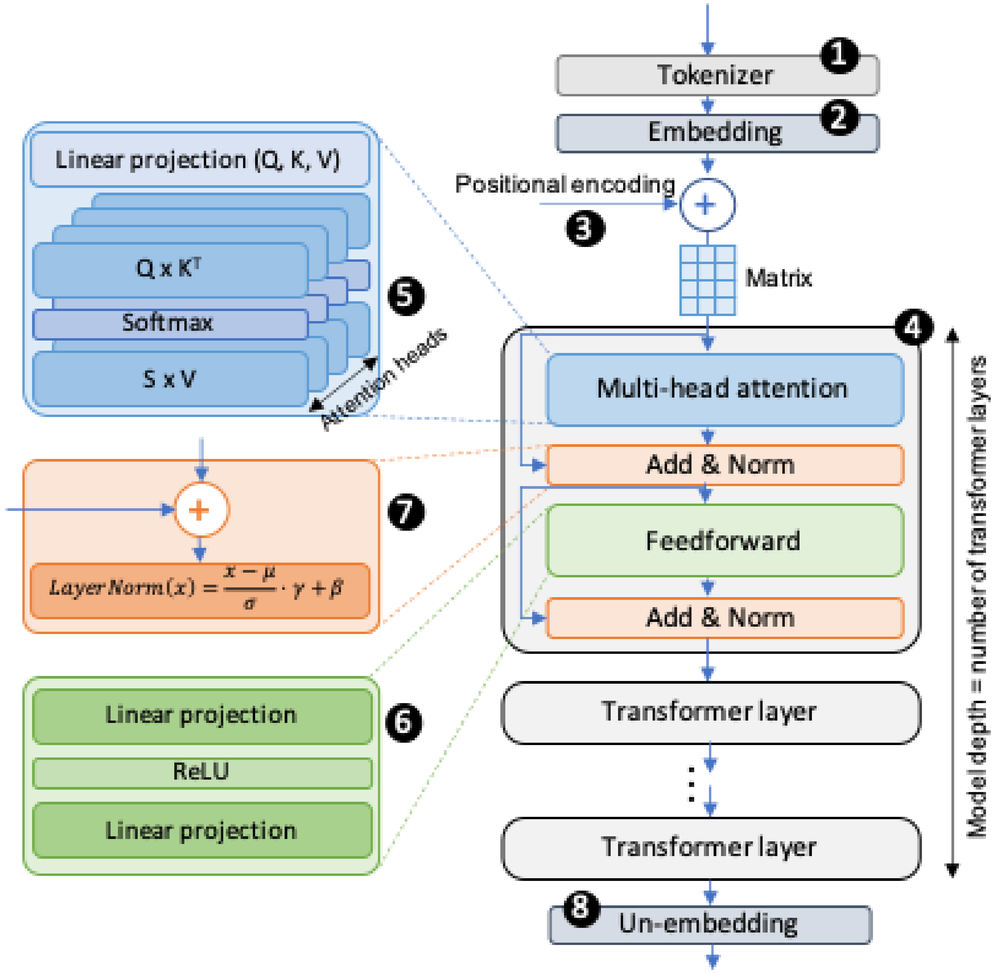
$$P(x_{n+t+1}||x_1, \dots x_{n+t})$$

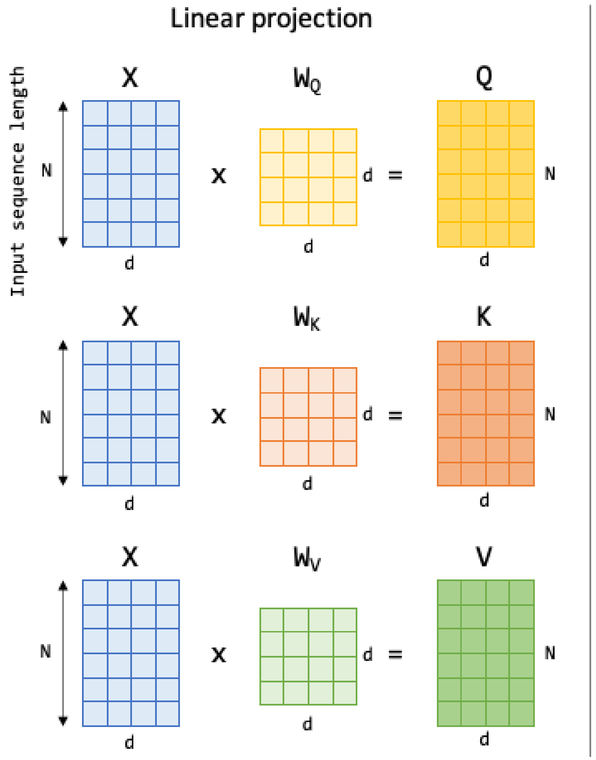
Each output token depends on previous generated token

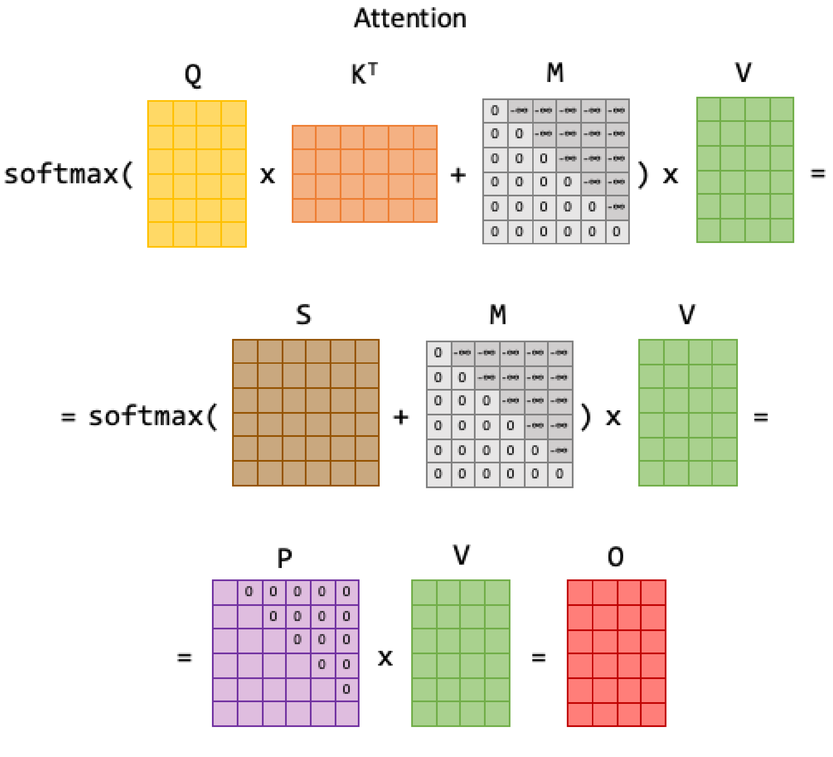
Prefill - Matrix multiplication

Decode - Matrix-vector multiplication

Transformer Computation

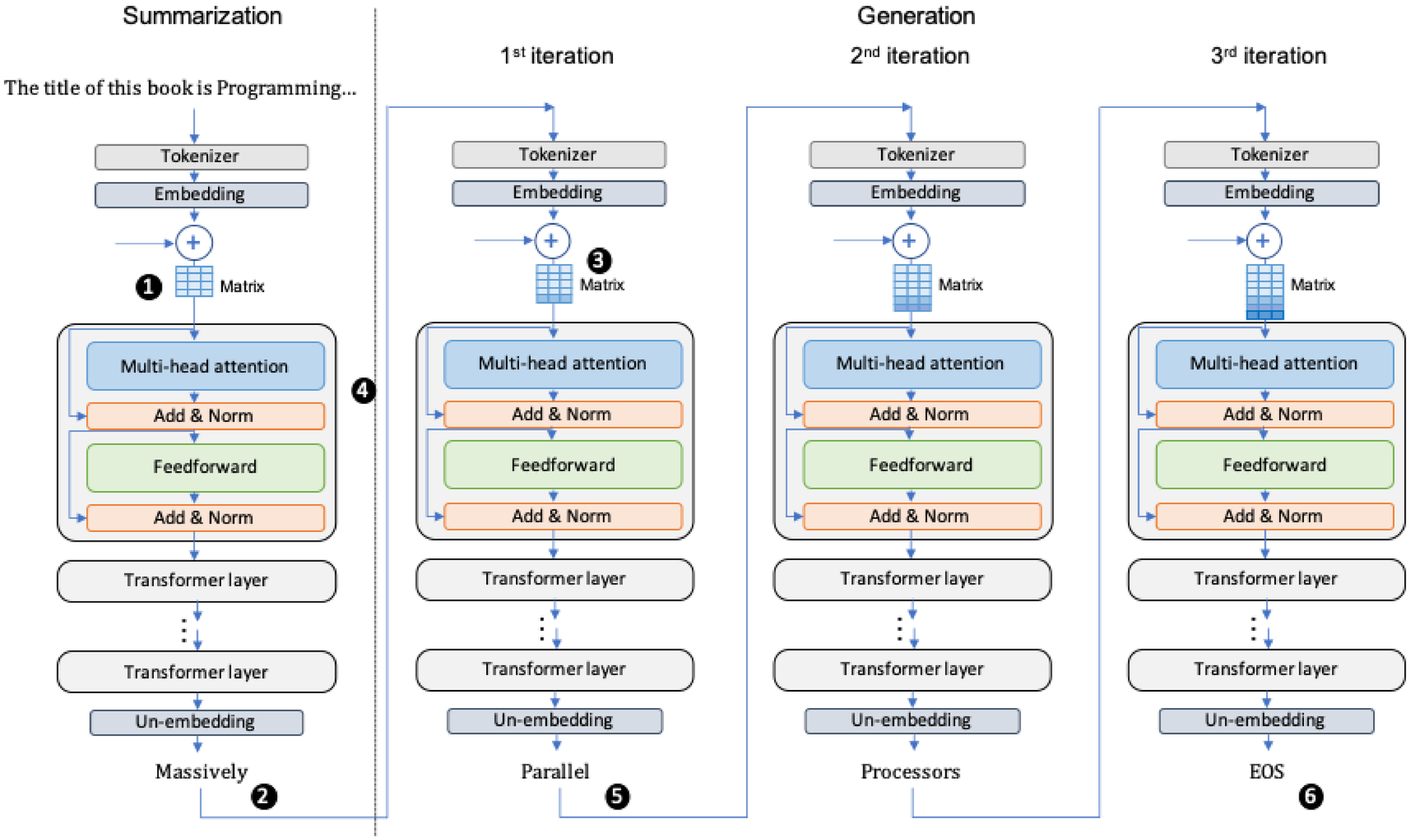






Credits: PMPP textbook v5.0 (yet to be released)

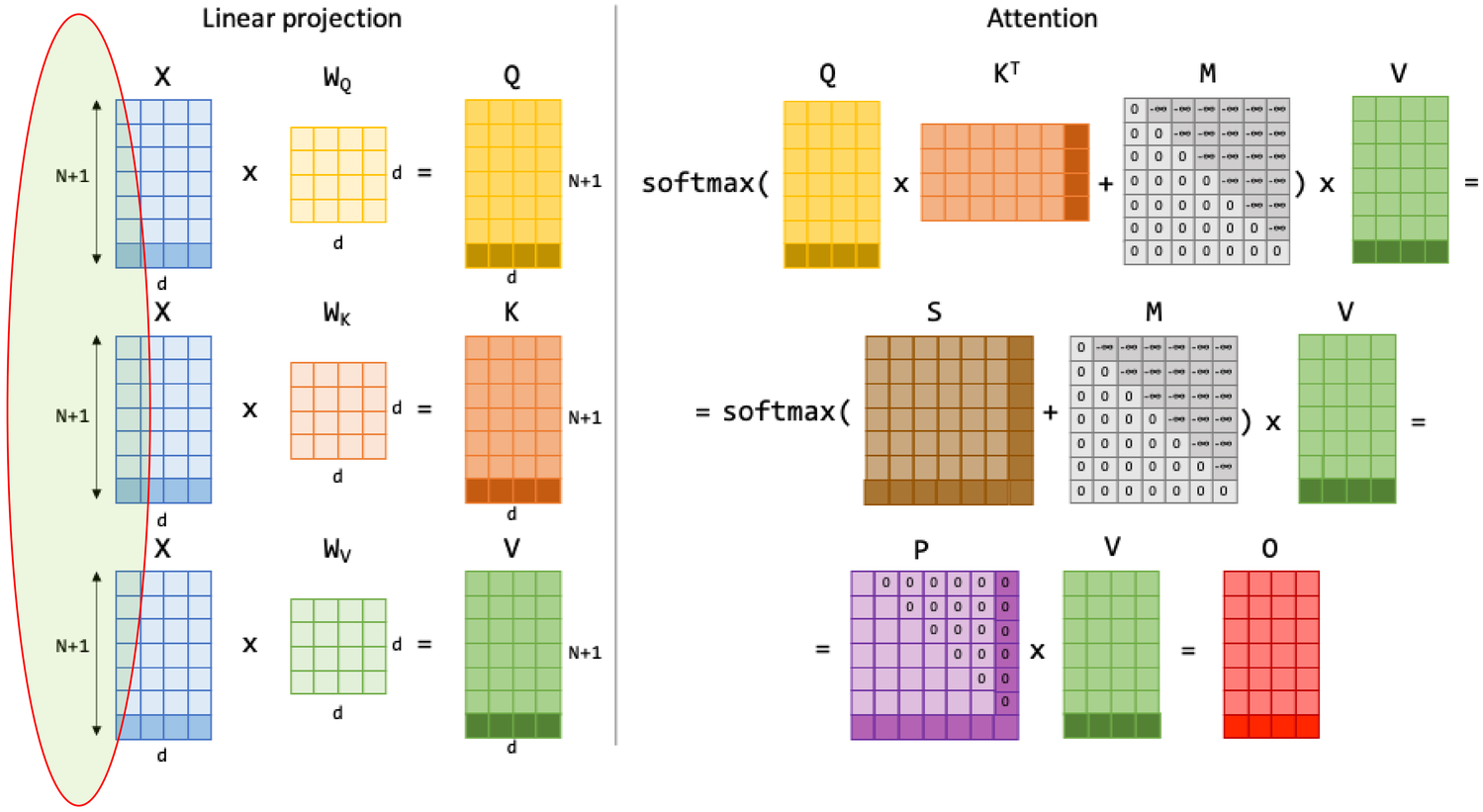
Next token prediction



Credits: PMPP textbook v5.0 (yet to be released)

Transformer Computation

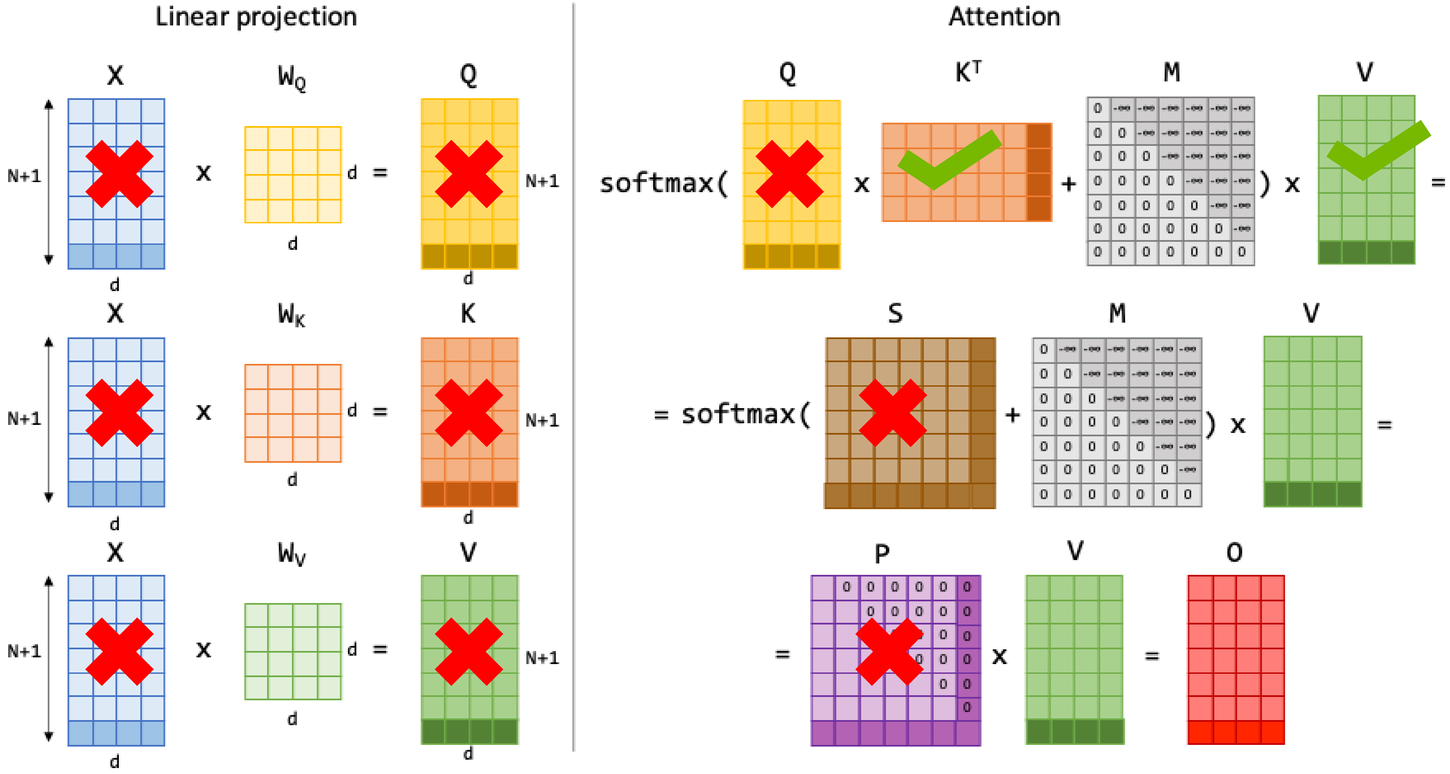
- Five matrix multiplications are performed in each transformer layer
- Seq len (N) can grow to millions
  - GEMM becomes expensive
- Observation:
  - one additional row in the input matrix X from one iteration of the decoding phase to the next
  - Avoid repeating the computation exploiting GEMM property



Credits: PMPP textbook v5.0 (yet to be released)

Transformer Computation

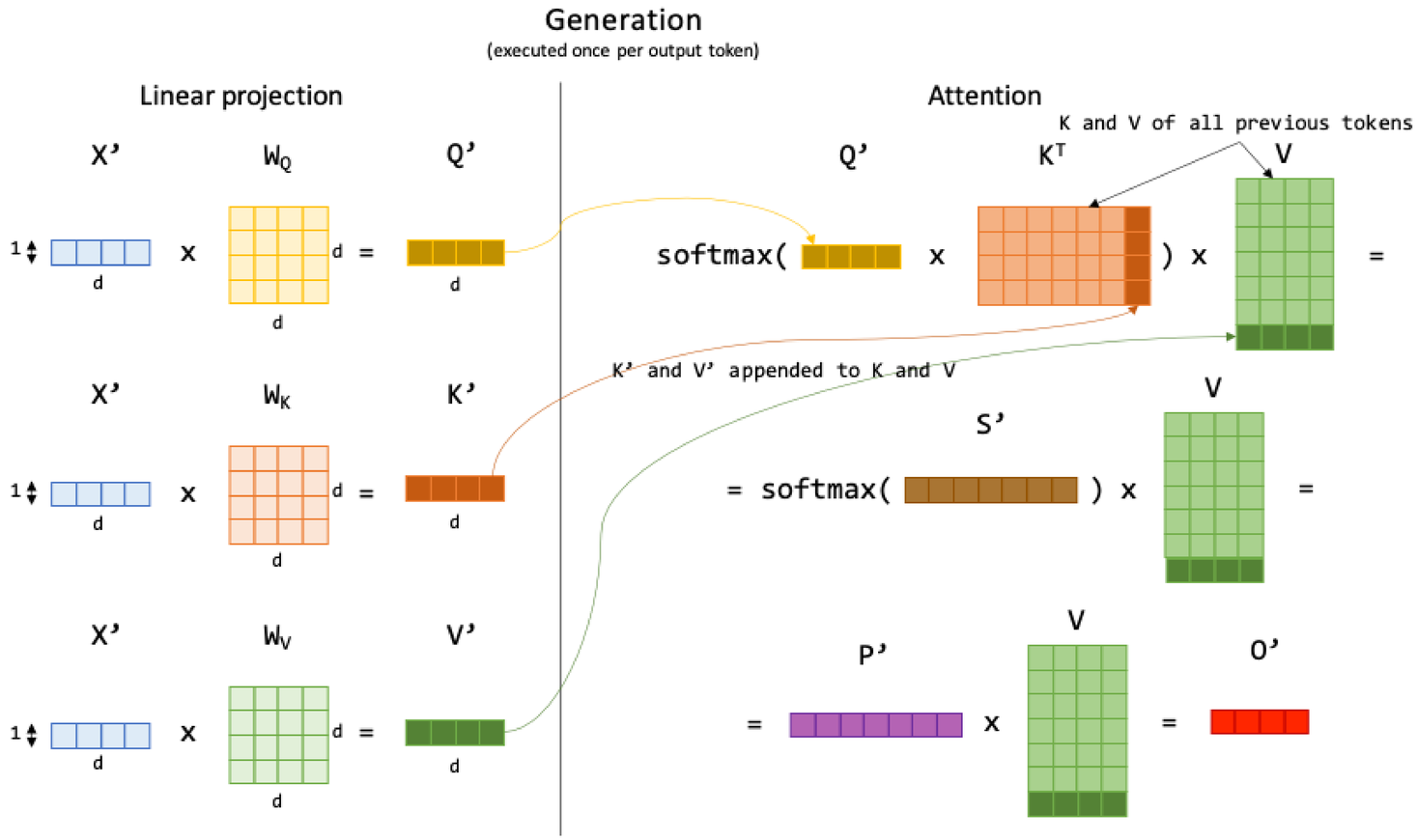
- Five matrix multiplications are performed in each transformer layer
- Seq len (N) can grow to millions
  - GEMM becomes expensive
- Observation:
  - one additional row in the input matrix X from one iteration of the decoding phase to the next
  - Avoid repeating the computation exploiting GEMM property



Credits: PMPP textbook v5.0 (yet to be released)

KV Caching

- Reusing the computation saves a lot of compute at the cost of memory
- We move from GEMM → GEMV
  - Compute → memory bound
- But memory needs to be allocated and managed efficiently
  - PagedAttention (vLLM)
  - RadixAttention (SGLang) across requests

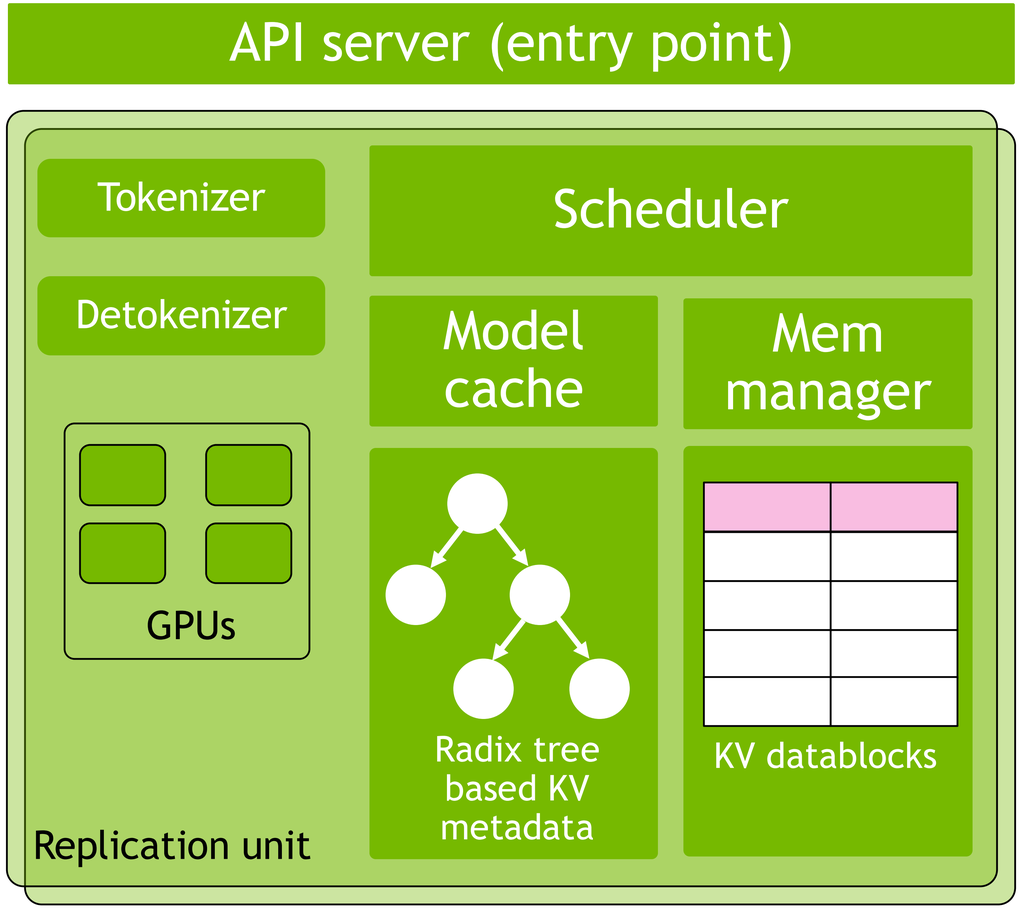


Credits: PMPP textbook v5.0 (yet to be released)

LLM Inference Framework

An example using SGLang

https



Naïve K8s based autoscaling approach

Good Model != Success

Good Model + Good System == Success

^

That hacks the inference pareto frontier

Applications have Diverse SLO

TTFT = Time for First Token, TPOT = Time Per Output Token

	TTFT	TPOT
Chat	Low (< 1s)	Match read speed (~ 100ms)
Search	Very Low (~ 200ms)	Match read speed (~ 100ms)
Program	Very Low (~ 200ms)	Very Low (~ 50ms)

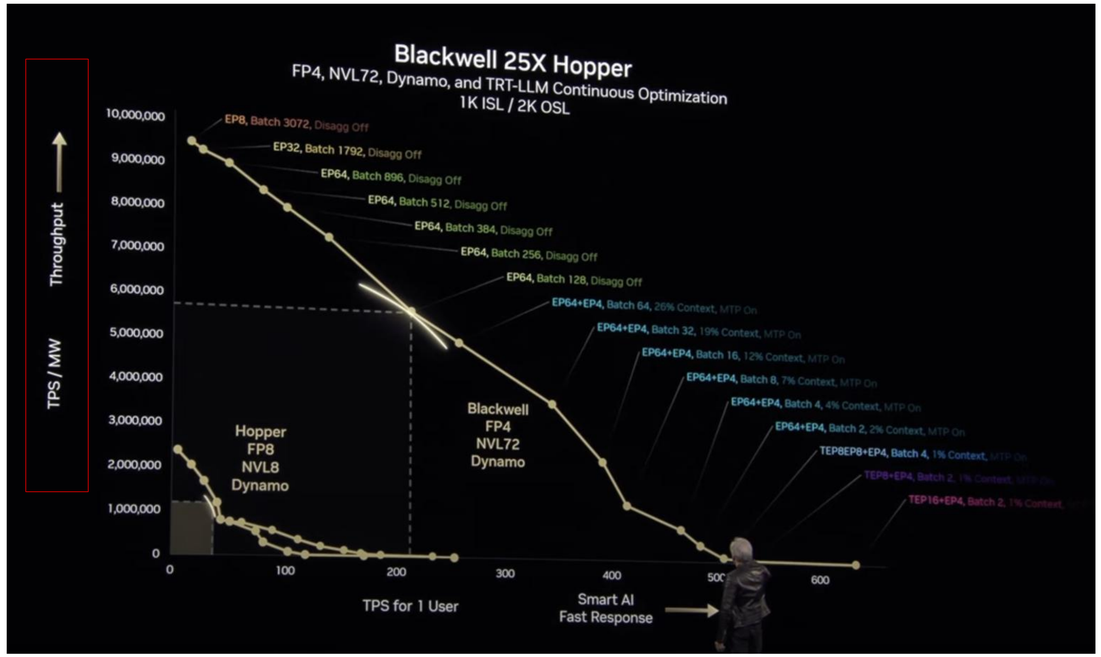
Credits: Junda Chen

Measure Serving Cost

Throughput = # Requests / sec

≈ 💰 per request

Throughput as a proxy of dollar per request



Jensen, GTC'25

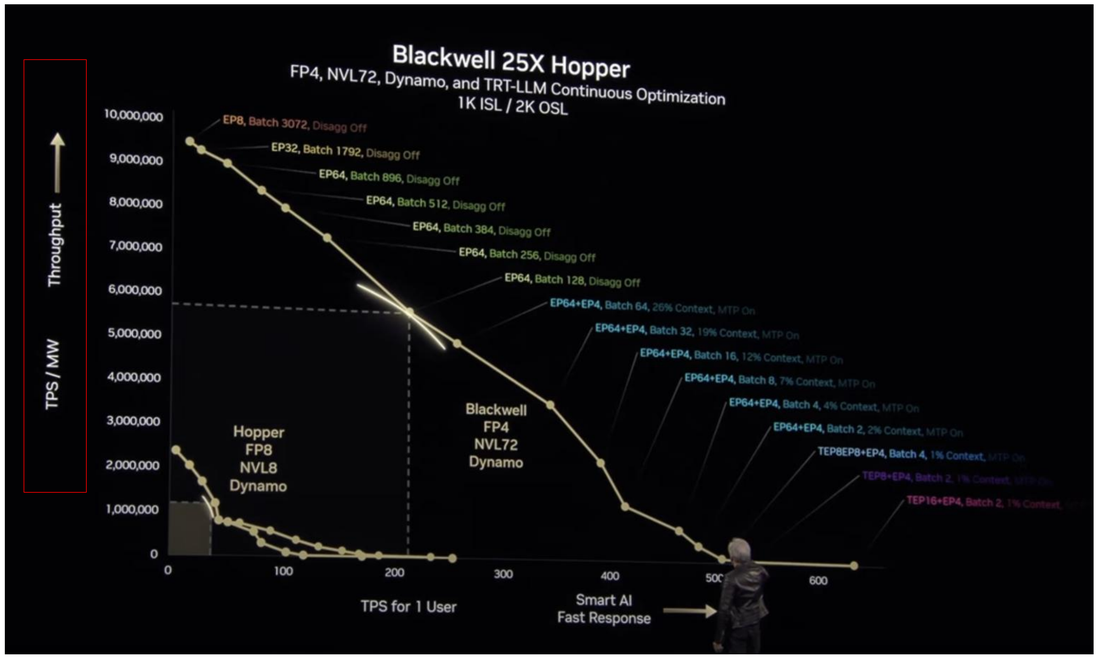
Credits: Junda Chen

Measure Serving Cost

Throughput = # Requests / sec

≈ 💰 per request

Throughput as a proxy of dollar per request



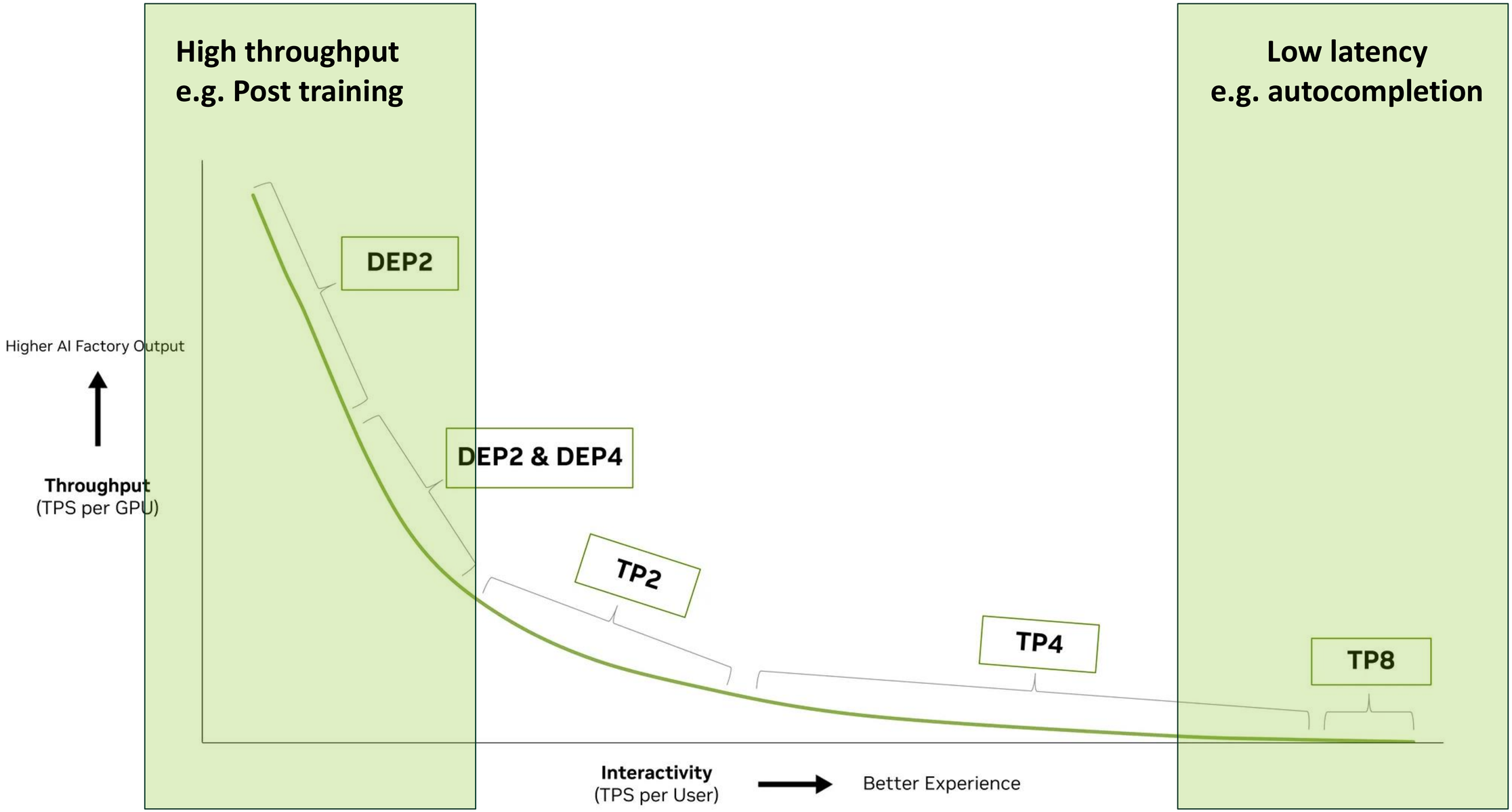
Jensen, GTC'25

Goodput = # Requests / sec satisfying the SLO constraint

≈ 💰 per good request

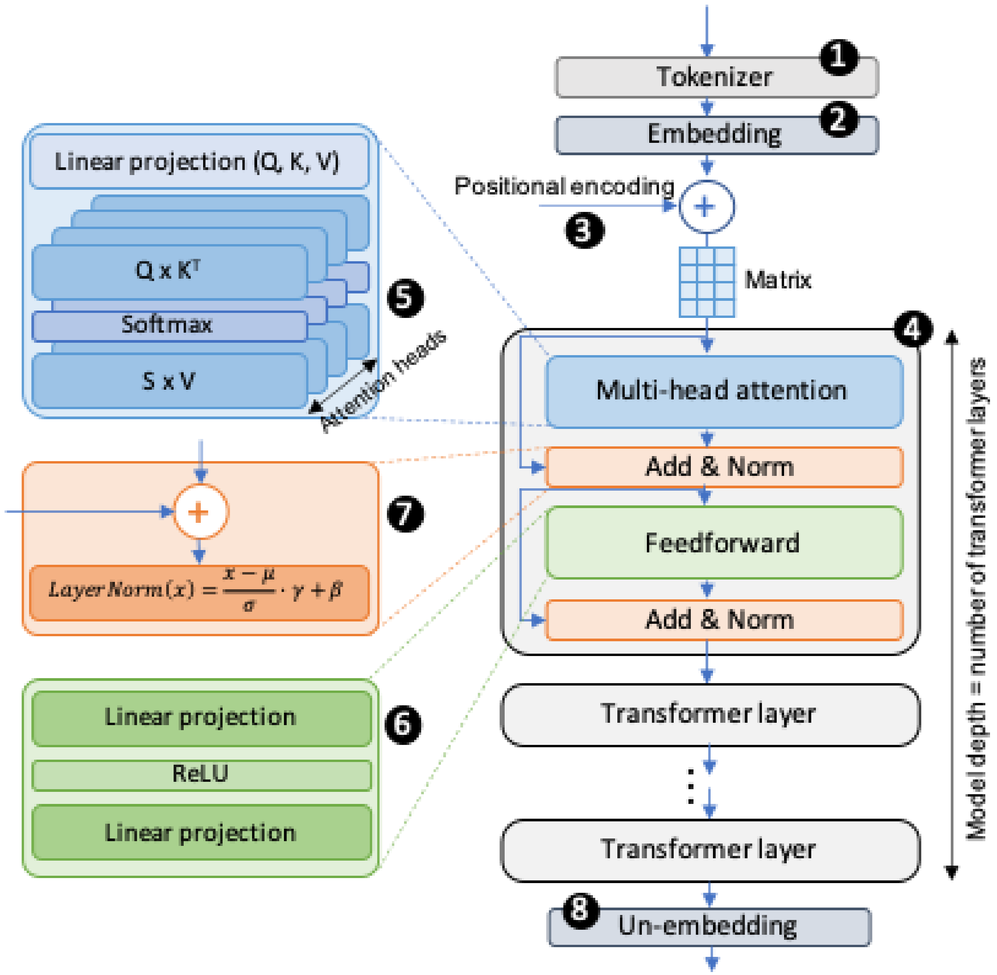
Credits: Junda Chen

Pareto Curve



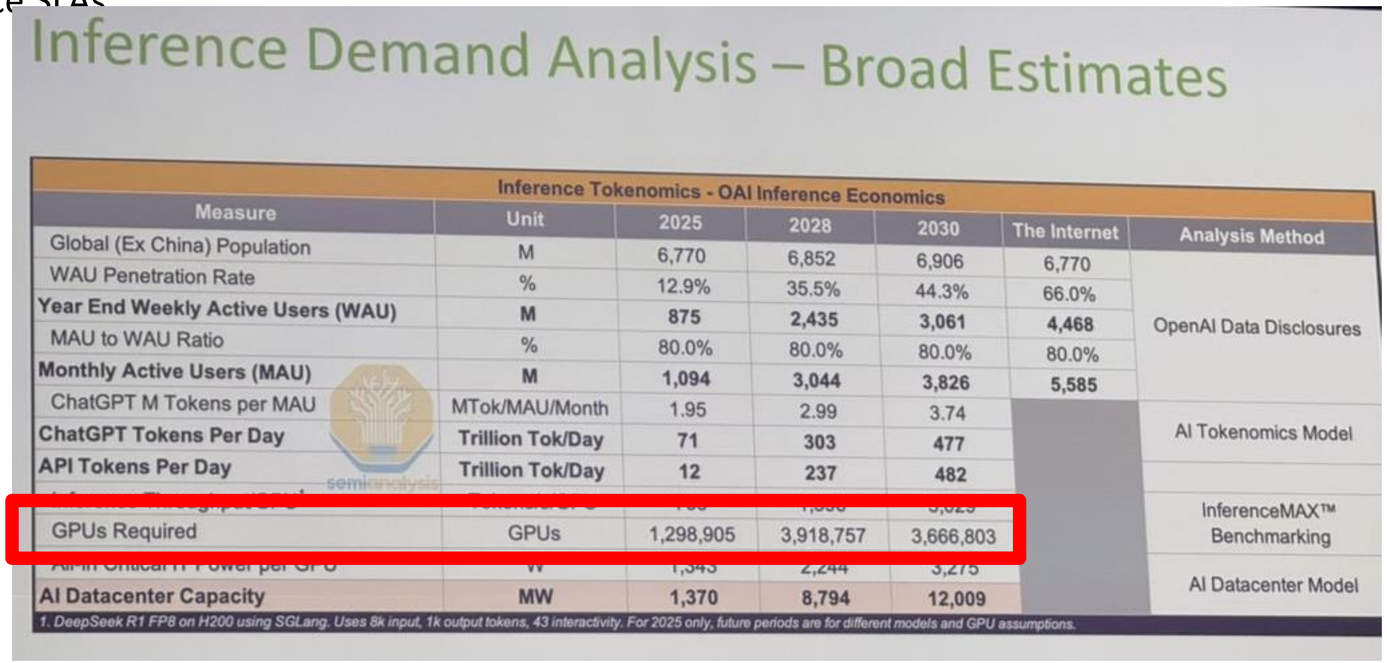
https://developer.nvidia.com/blog/nvidia-blackwell-leads-on-new-semianalysis-inferencemax-benchmarks/

How would you build an inference service at scale?



Challenges Serving AI Inference at scale

- Need to vary the parallelism strategy for hitting performance SLAs
- Multiple requests with varying degrees of ISL/OSLs
  - Some doing multiturn
  - Some from agents
- Heterogenous hardware
  - Different generations and across platforms
- Need for fault tolerance
- Multiple models to serve
- AI factory TCO is key at scale
  - Optimize for goodput
  - Optimize for power
  - Optimize for maximizing KV cache hit rate
- …



Source: Semianalysis OCP’25

LLM Deployments in 2025

Scale Never Seen Before

Input Length

1M+ with RAG, Video understanding, Code…

Output Length

Reasoning, Agentic Workflows…

Model Size

O(100B) – O(1T) parameters

Users + UX

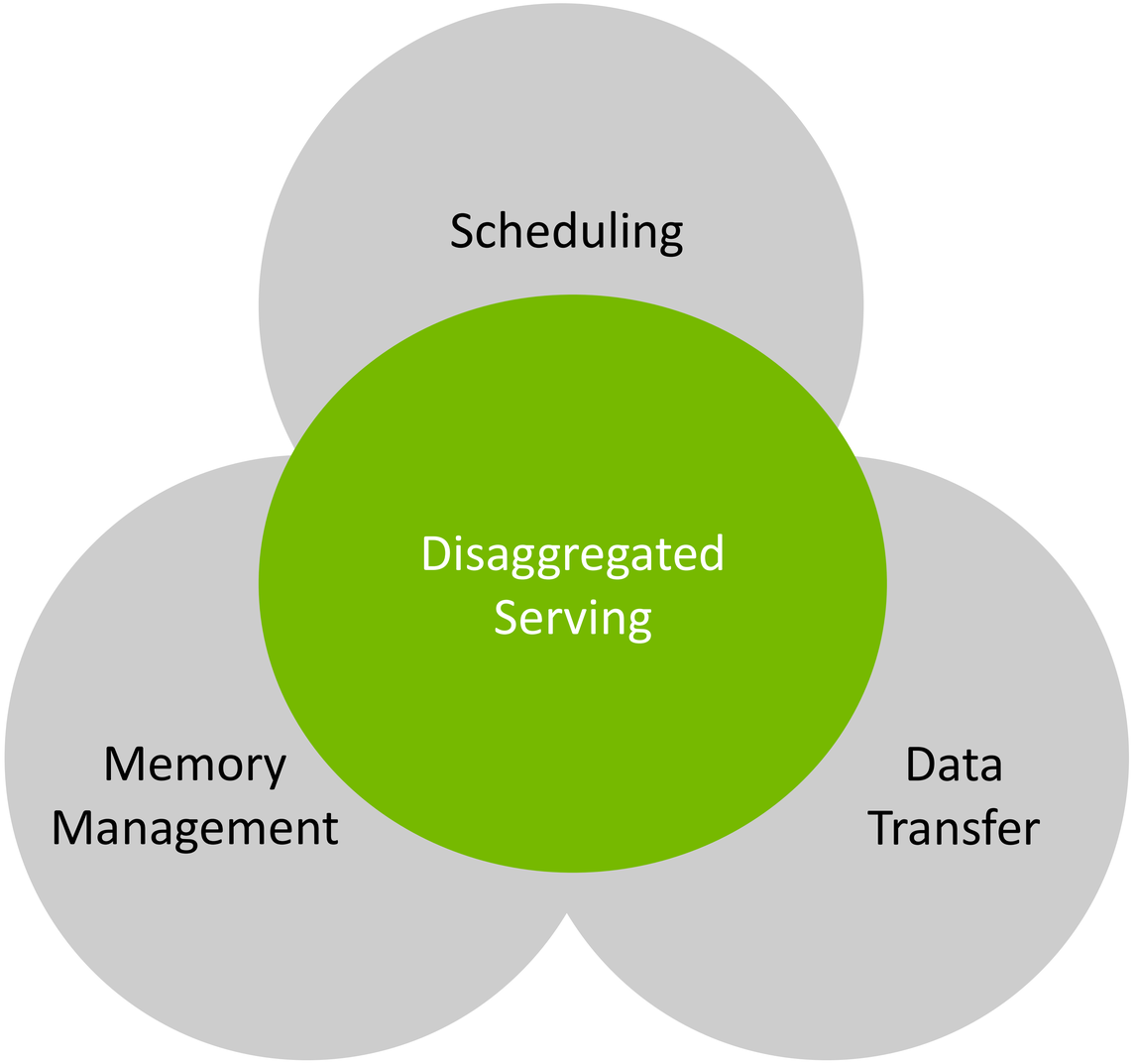
Fast tokens served = Revenue generated

Deployment Constraints

HW availability, power, rack space, cooling…

NVIDIA Dynamo: Distributed Inference At Scale

Systematic approach to AI inference at scale



System level optimization

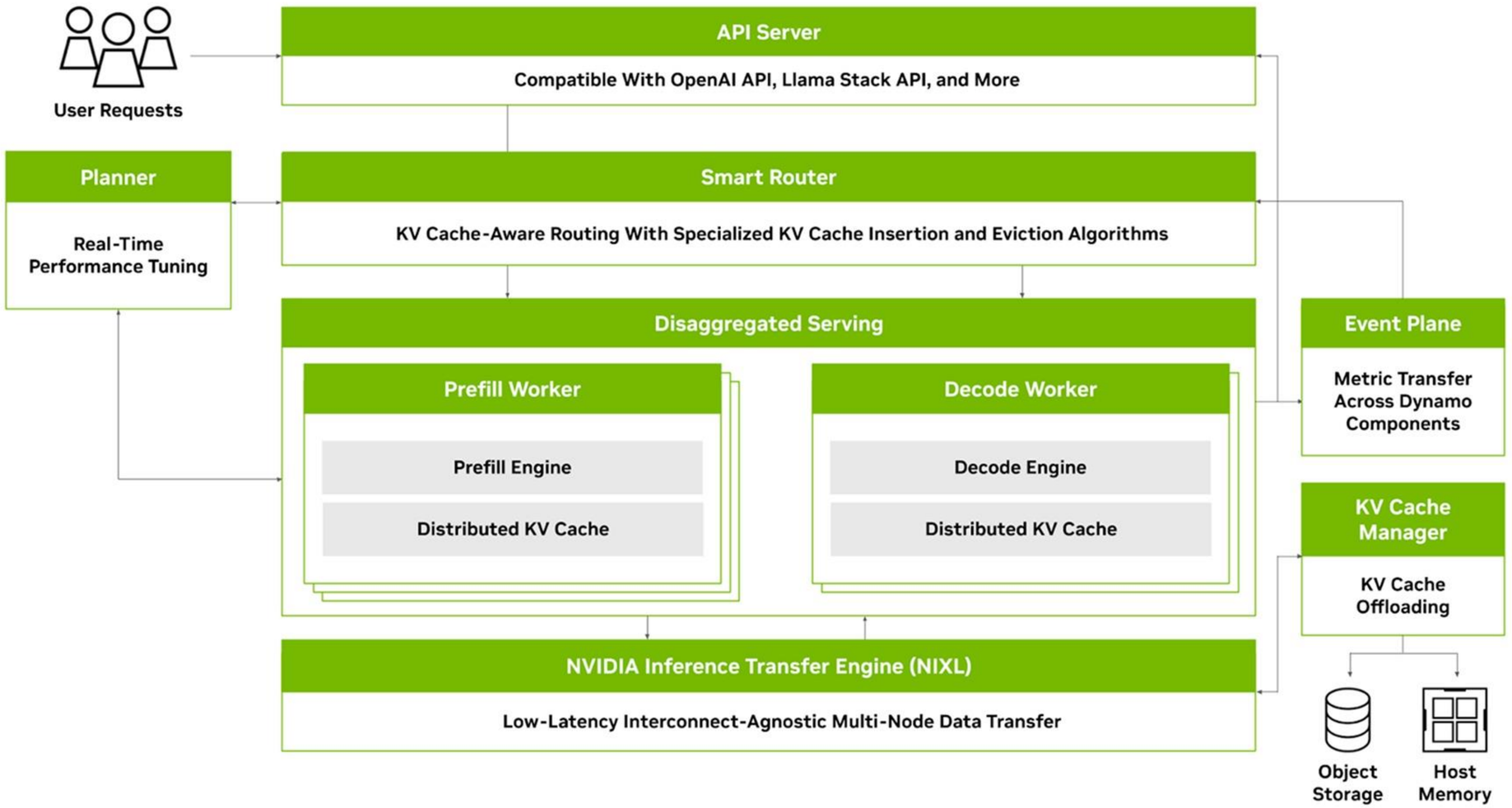
Supports all inference engines

Modular Components

Production grade serving

NVIDIA Dynamo: Distributed Inference At Scale

Modular design, independently scalable yet highly efficient



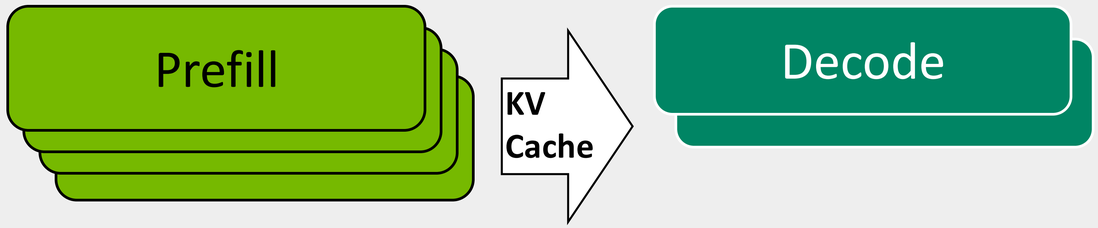
Disaggregated serving

Prefill
Compute bound

≠

Decode
Memory bound

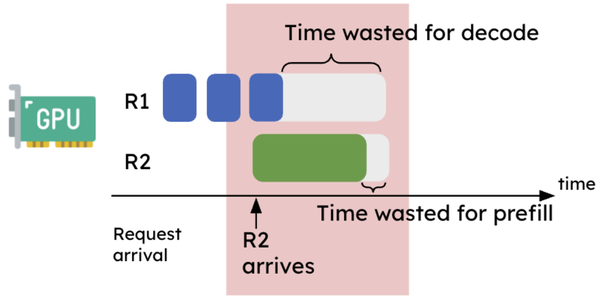
Scale prefill & decode independently on separate GPUs



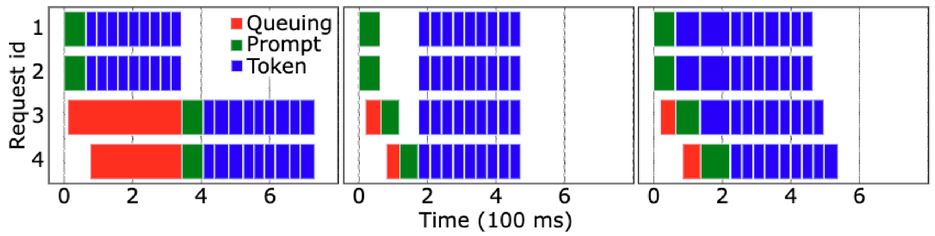
Apply suitable parallelism for prefill and decode separately

Why do Disaggregated Serving?

Q1. Prefill and decode interference.



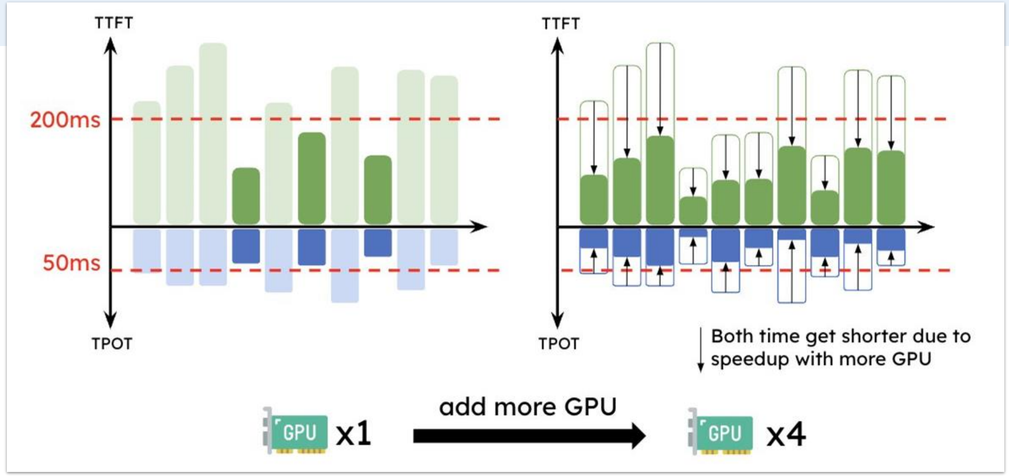
Incoming prefill request elongates the decode 😱



(a) Request-level. (b) Continuous. (c) Mixed.

Fig. 2: Batching mechanisms and their latency impact on the prompt and token phases.

Q2. Resource and Parallelism Coupling



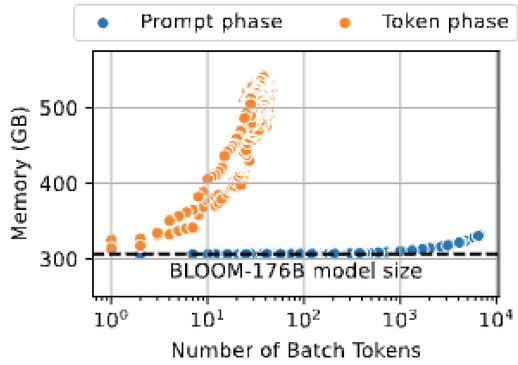


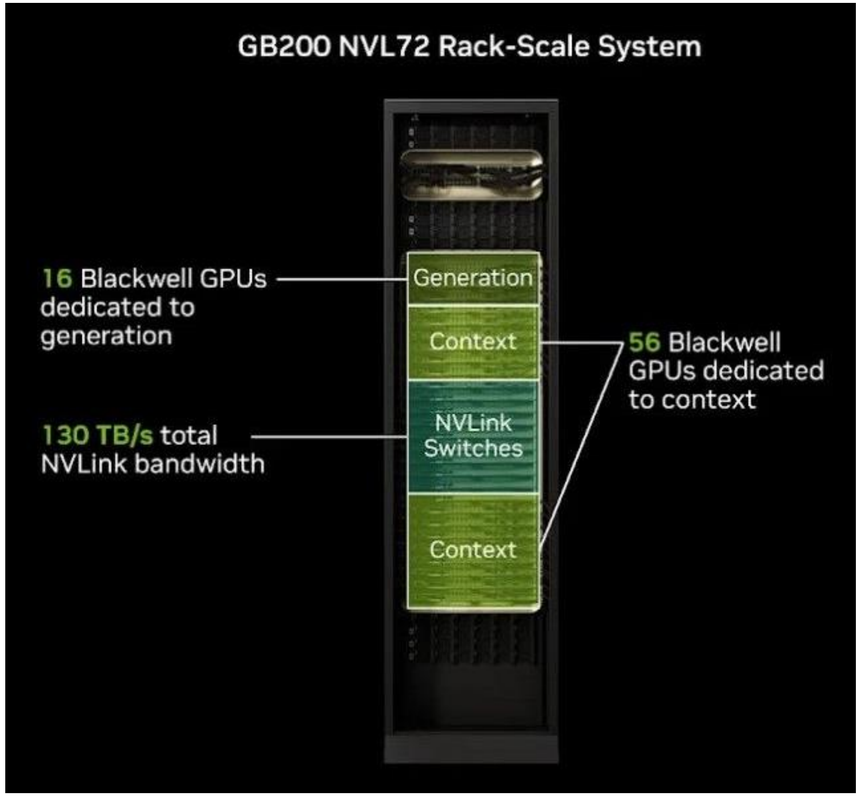
Fig. 7: Required memory with batching in prompt/token phases.

Credits: Junda Chen from DistServe. Some figures from Splitwise

Implementing Disaggregated Serving

- Good proxy example for learning
  - https://docs.vllm.ai/en/latest/examples/online_serving/disaggregated_serving/

- Use for production
  - https://github.com/ai-dynamo/dynamo/blob/main/examples/backends/vllm/deploy/disagg_kvbm_2p2d.yaml

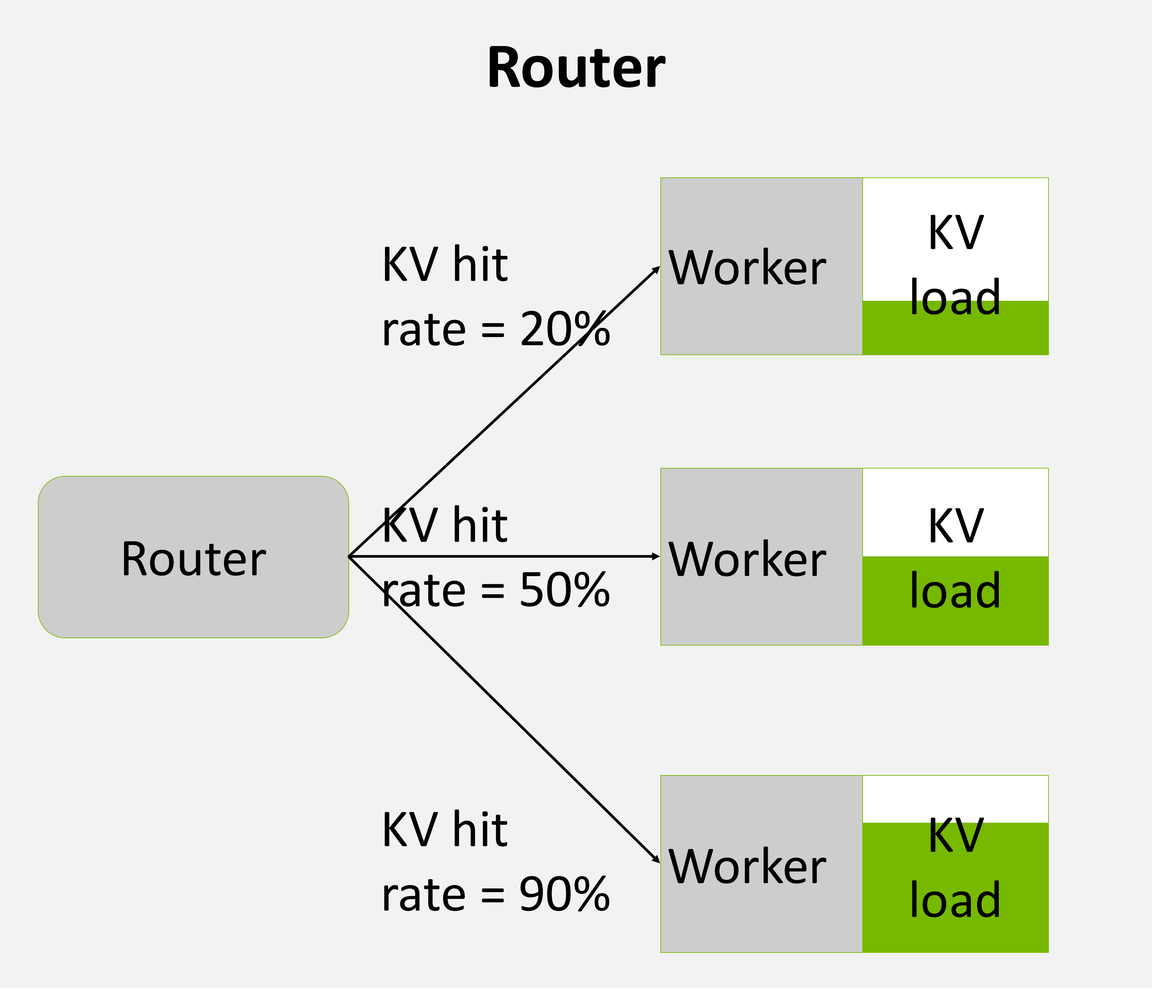


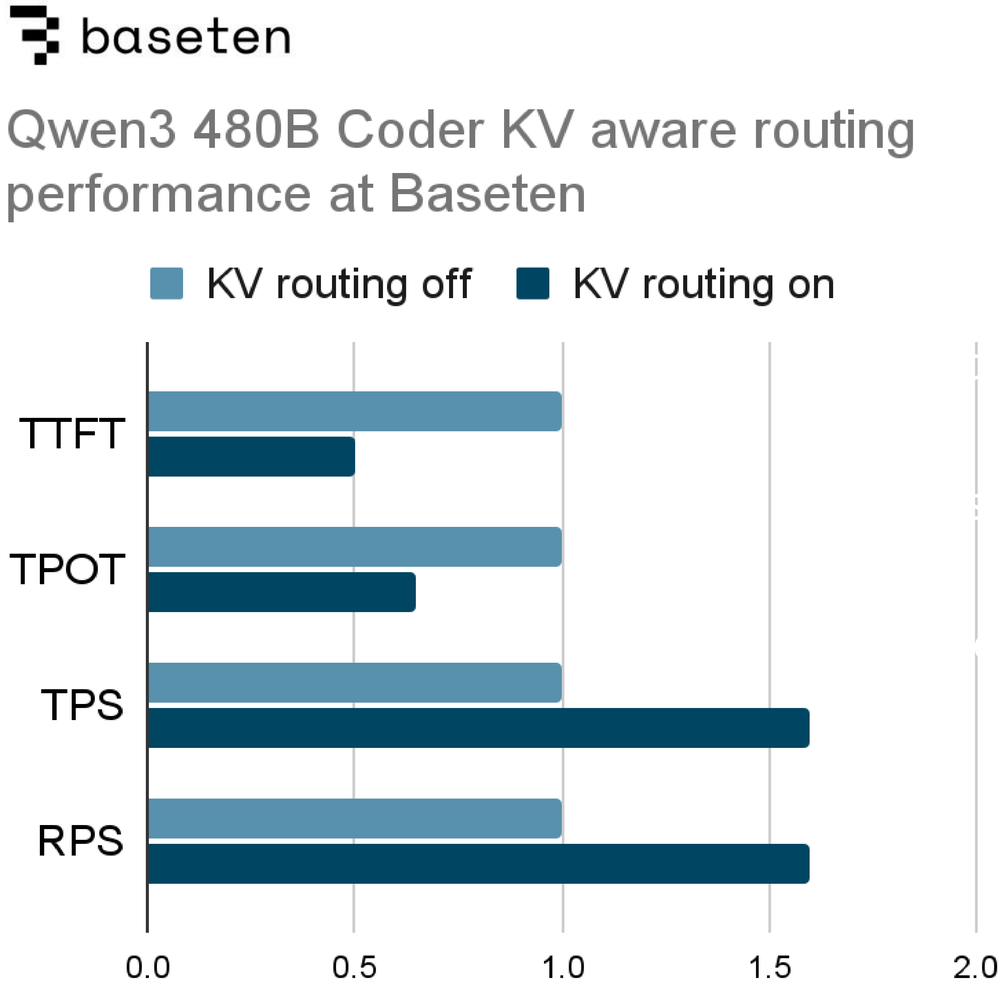
Disaggregated Inference is one feature of Dynamo

At scale inference needs solving the AI factory TCO optimization problem

Scheduling

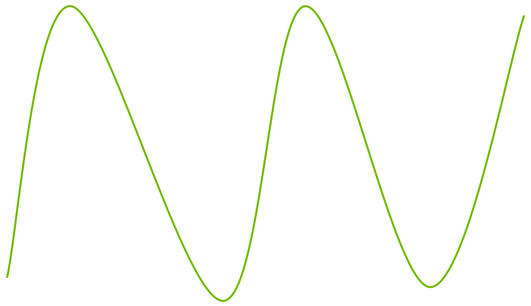
KV aware routing archives 2X TTFT, 1.5X TPOT and 1.6X TPS for Qwen3 480B at Baseten



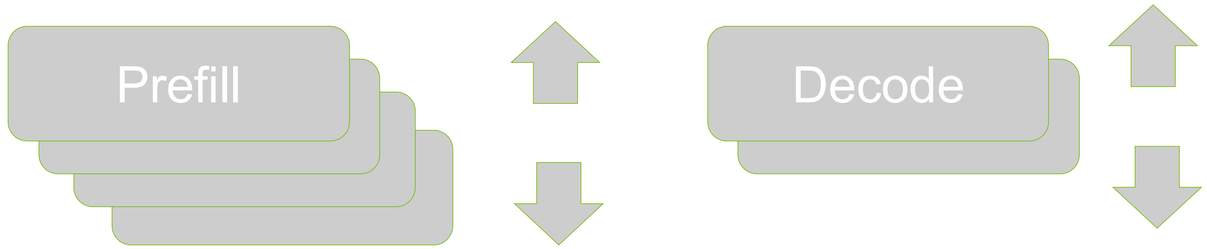


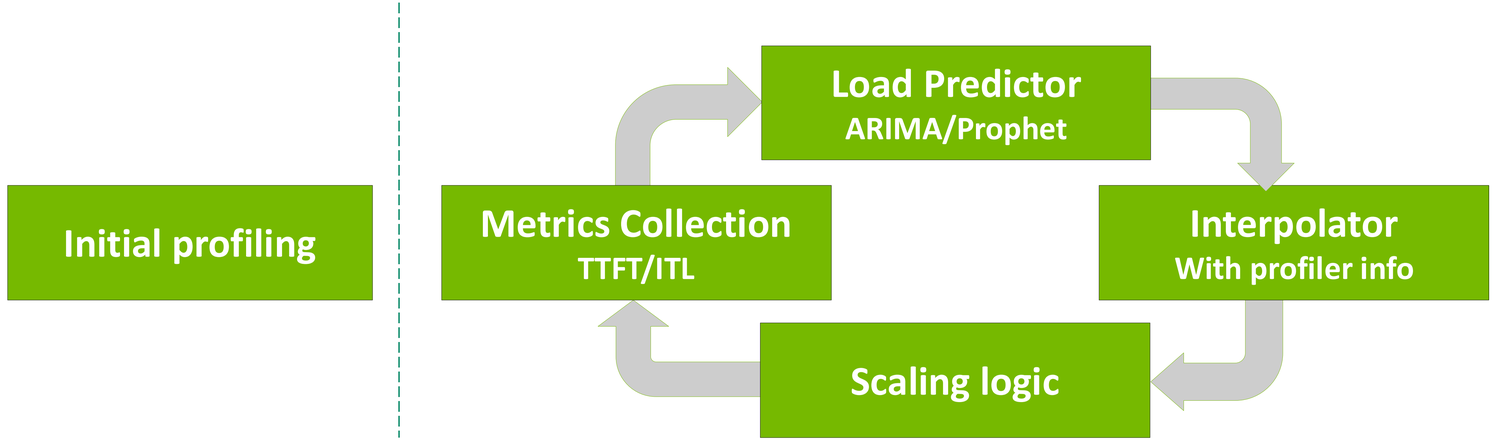
Online perf optimizer

LLM traffic can change rapidly



Planner autoscales in real time w.r.t SLA (TTFT & TPOT)





Alibaba ACK

80% reduction in TTFT
With 5% less GPUs

Based on performance benchmarks presented by Alibaba at APSARA conference on 2025.09.24.
Source: https://yunqi.aliyun.com/2025/session?agendaId=6062 at 2 hr 50 min mark

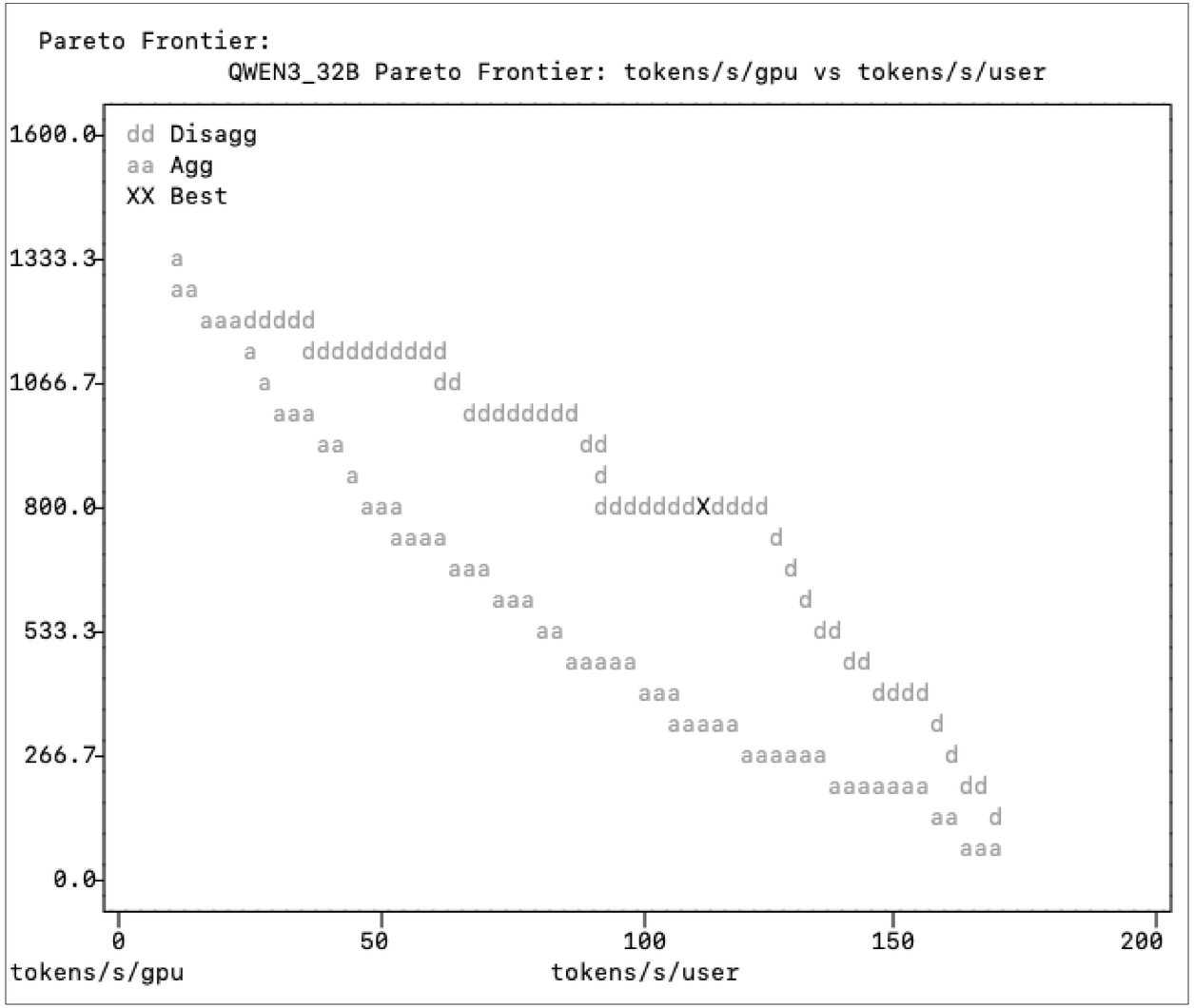
Offline Perf Configurator

AIConfigurator

- Find best config for disaggregated serving
- Compare projected perf gain
- <1 min using laptop vs 2 days using GPUs
- Generate yaml to deploy with Dynamo
- Modular and standalone

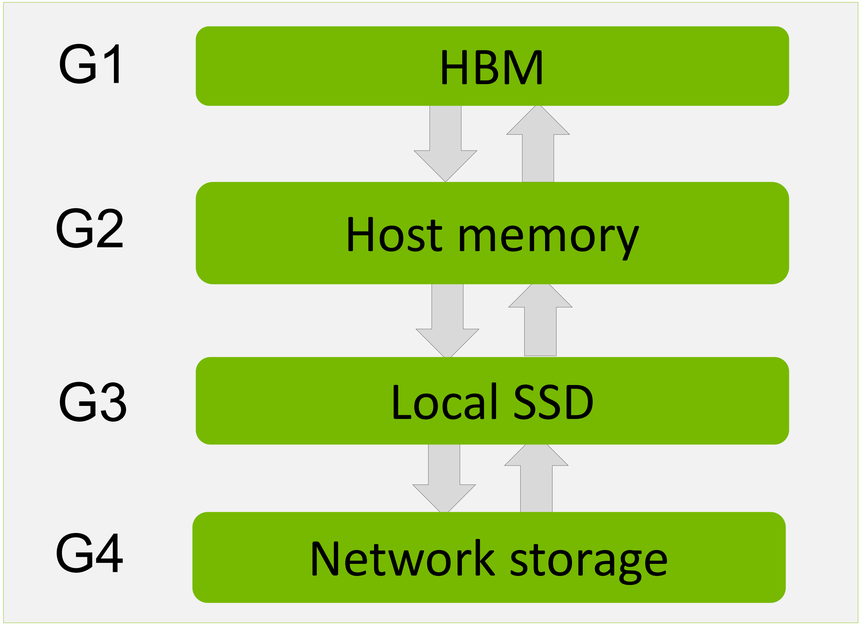
```
aiconfigurator cli

      --model QWEN3_32B
      --total_gpus 32
      --system h200_sxm
      --isl 4000
      --osl 500
      --ttft 300
      --tpot 10
```

SLA requirements



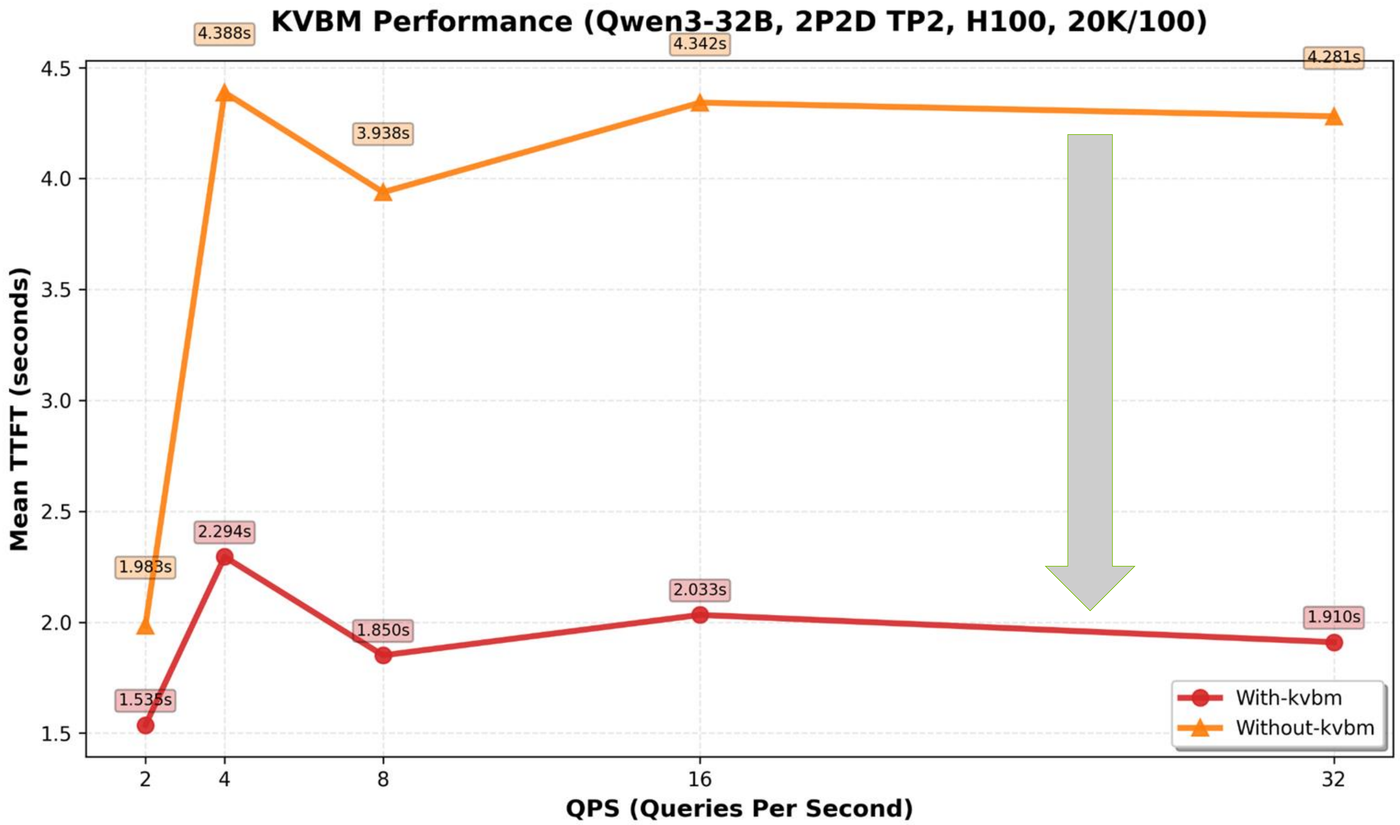
Memory Management

Leverage all memory and storage available in the data center

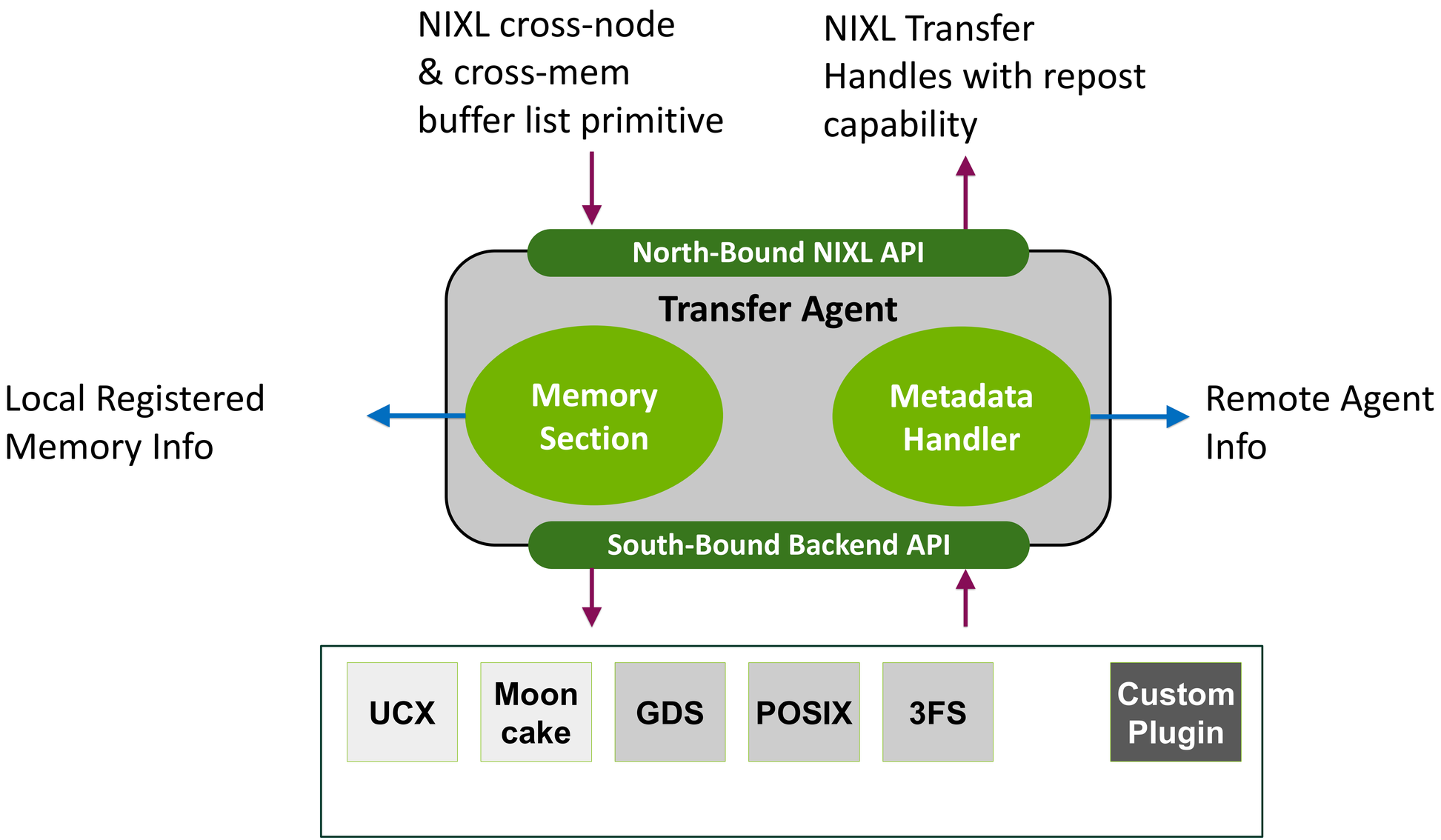


Use cases

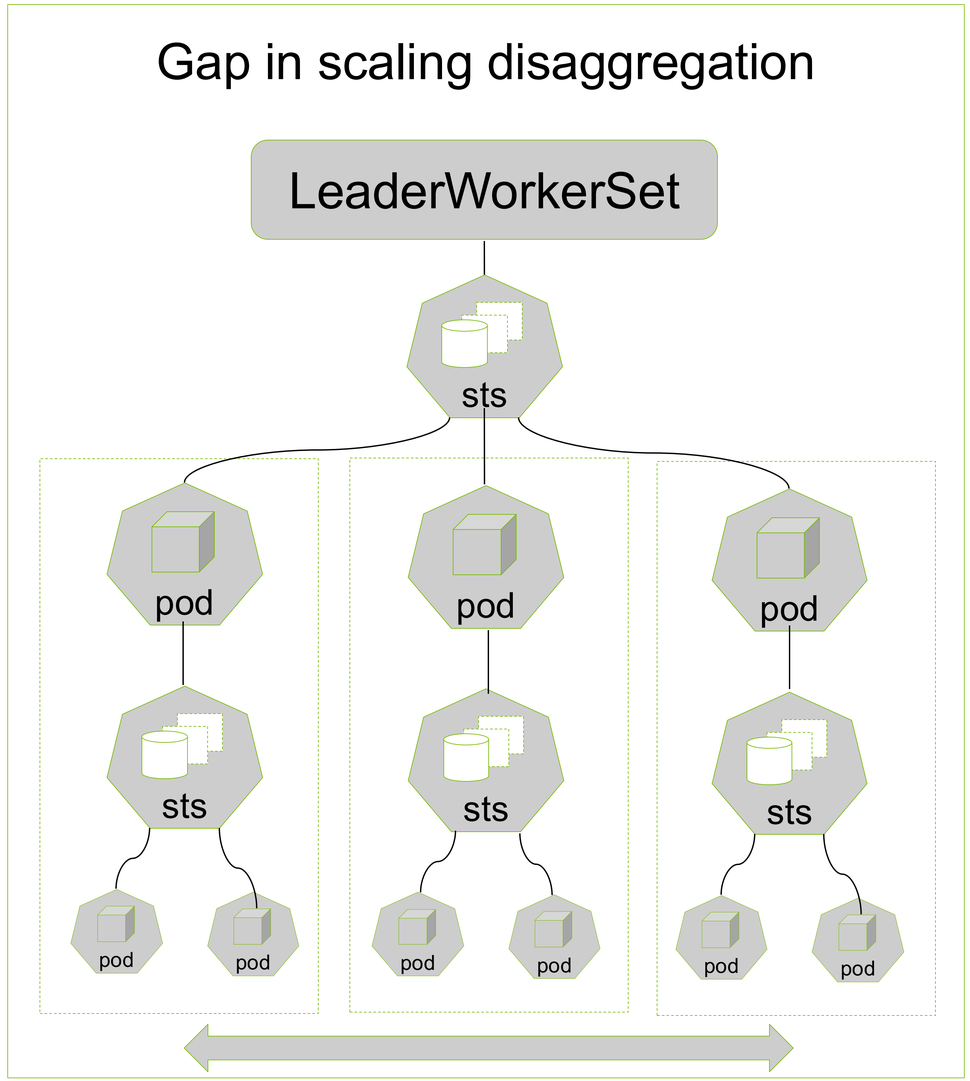
- Multi-turn conversations
- Code generation
- Agent calls
- Generative recommender

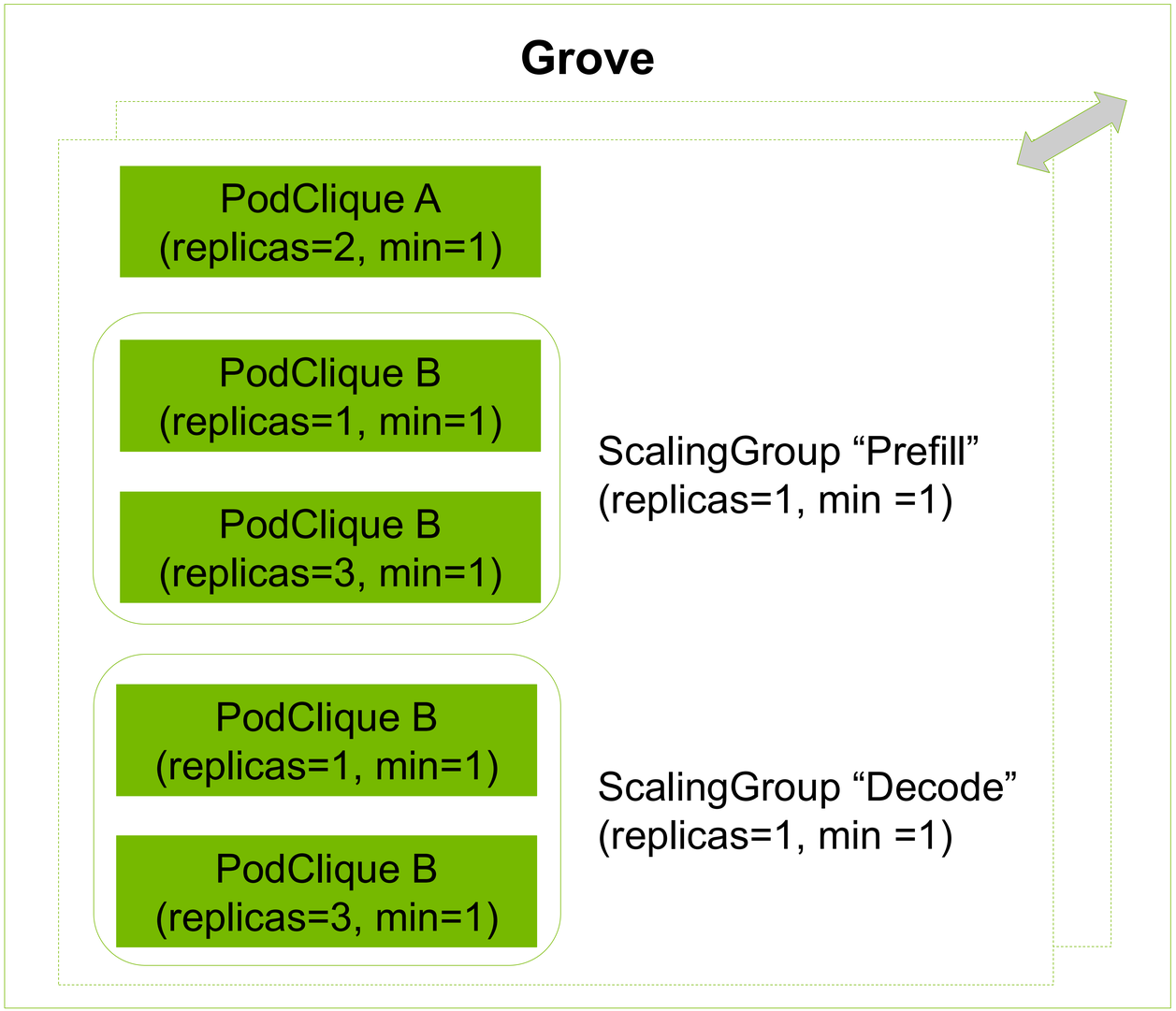


NIXL


NIXL cross-node
& cross-mem
buffer list primitive
NIXL Transfer
Handles with repost
capability
North-Bound NIXL API
Transfer Agent
Memory
Section
Metadata
Handler
South-Bound Backend API
Local Registered
Memory Info
Remote Agent
Info
UCX
Moon
cake
GDS
POSIX
3FS
Custom
Plugin

Topology Optimized Dynamic K8 Scheduling





Fault tolerance

Ensure detection, resilience, and recovery

High availability

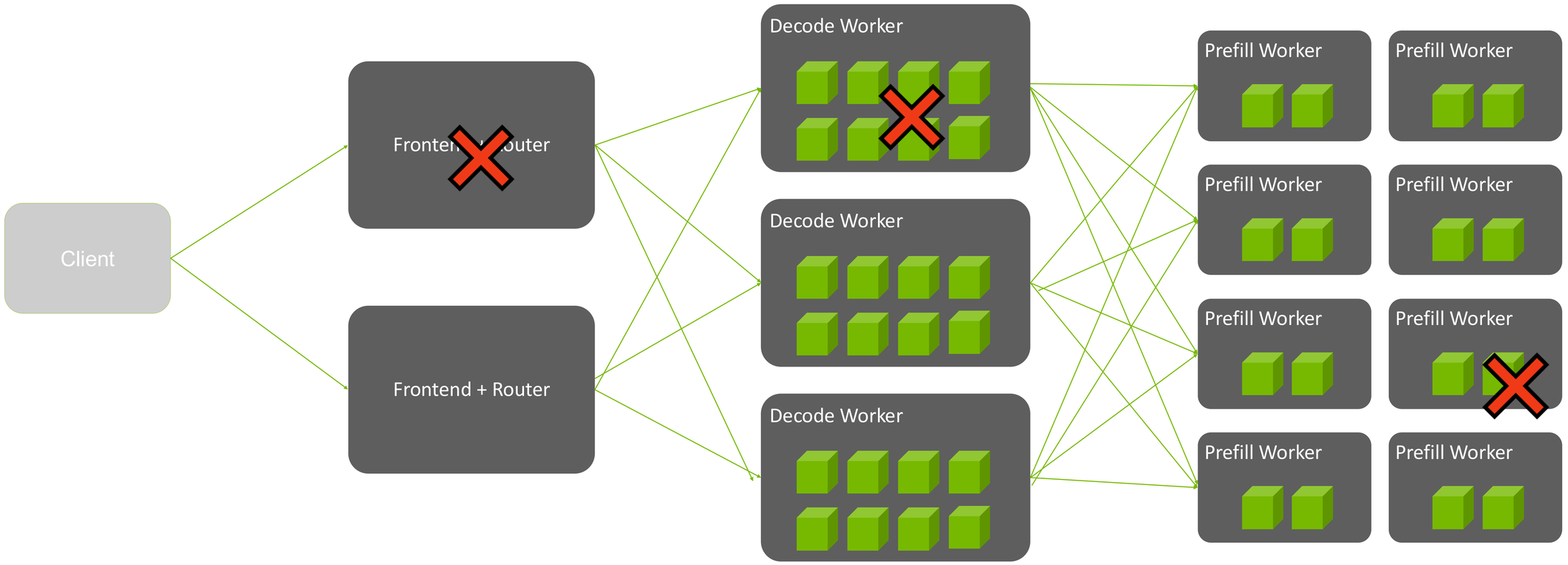
Automatic restart with shared router state

Request cancellation

Cancel request to avoid computation

Request migration

Migrate partial response at token level to avoid Decode



Challenges and opportunities

- We are barely at the start of large-scale AI inference deployment
- Heterogenous model deployment is least explored at scale
  - End to end agentic tasks need multiple models to closely exist together for performing the task
  - KV cache and other efficiency management
  - Fast data transfer mechanism
- Fault tolerance and performance measurement tools for inference at scale do not exist
  - AIPerf – a stab at performance measurement at scale from API standpoint of view
  - Agentic performance modelling?
  - Multimodal performance characterization?
  - Fault injection frameworks for inference?
    - Dynamo is working towards building something here but it is quite a task itself
- Speculative decoding, diffusion-based reasoning model, … all open for exploration
- Enterprise RAG with CAG (cache augmented generation) at scale is unsolved issue
- How to evolve the stack fast with rapid innovations in model development?
- …

Conclusions

- AI inference is unsolved problem, and you can contribute and learn with us!

- https://github.com/ai-dynamo/dynamo （⭐）
- Join the discord group: https://discord.com/invite/D92uqZRjCZ

```
# Create a virtual environment and activate it
uv venv venv
source venv/bin/activate

# Install Dynamo from PyPI (choose one backend extra)
uv pip install "ai-dynamo[sglang]==0.6.1"  # or [vllm], [trtllm]
```

- Getting started with Dynamo: https://www.youtube.com/watch?v=1bRmskFCnqY
- Disaggregated inference demo: https://www.youtube.com/watch?v=_UDJy_5_Czw
- KV Aware router demo: https://www.youtube.com/watch?v=PRCZZKQirN8
- AIConfigurator demo: https://www.youtube.com/watch?v=KuZXeol0fKk