SNIA DEVELOPER CONFERENCE
SDC 25
By Developers FOR Developers

Hyatt Regency Santa Clara, CA
September 15-17, 2025

Disaggregated KV Storage: A new Tier for Efficient Scalable LLM Inference

Eshcar Hillel
Principal Research Scientist

PLiOPS
EXTREME DATA PROCESSOR

“Storage and Network Industry developers,

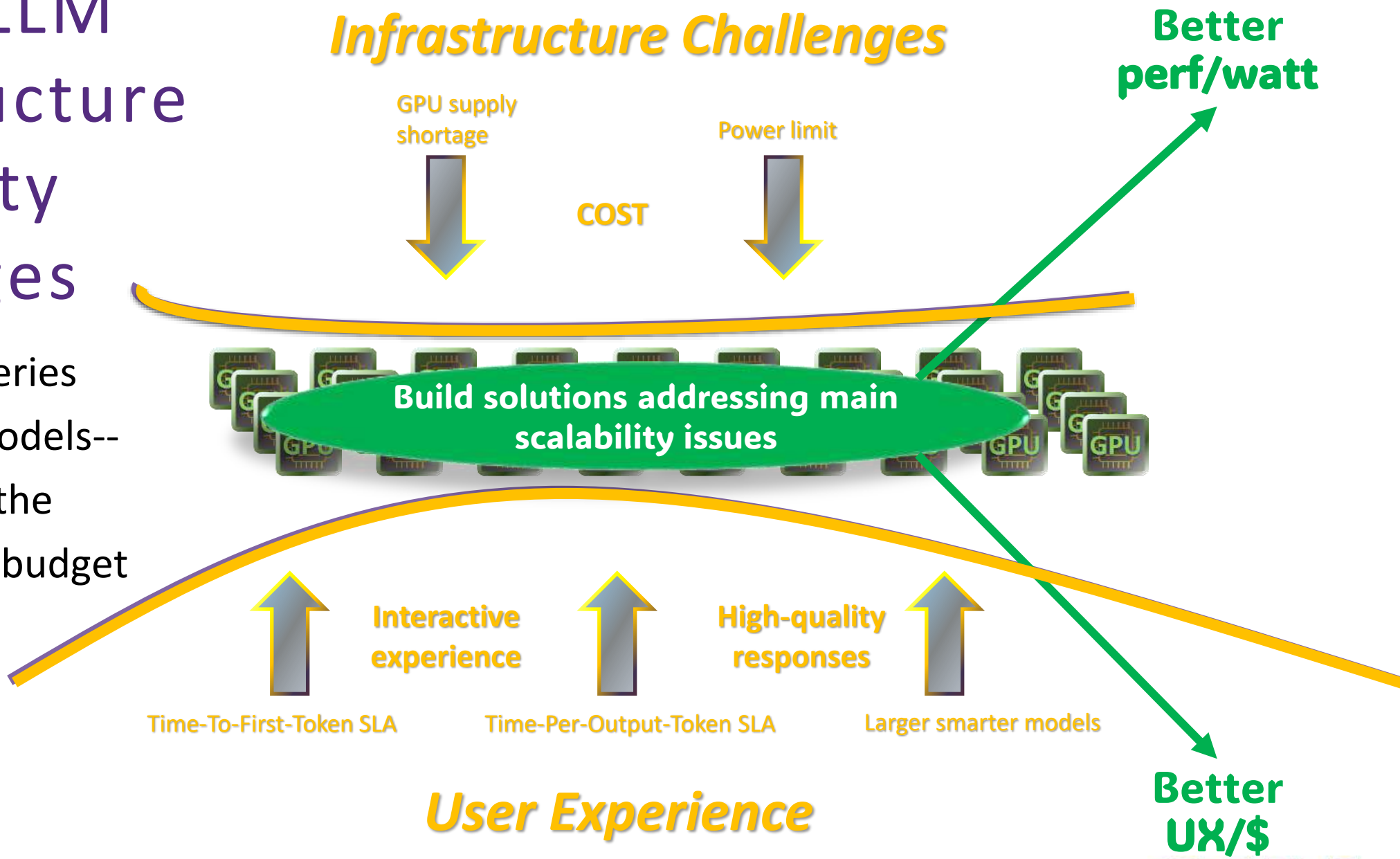
Ask not what AI can do for you.

Ask what you can do for AI!“

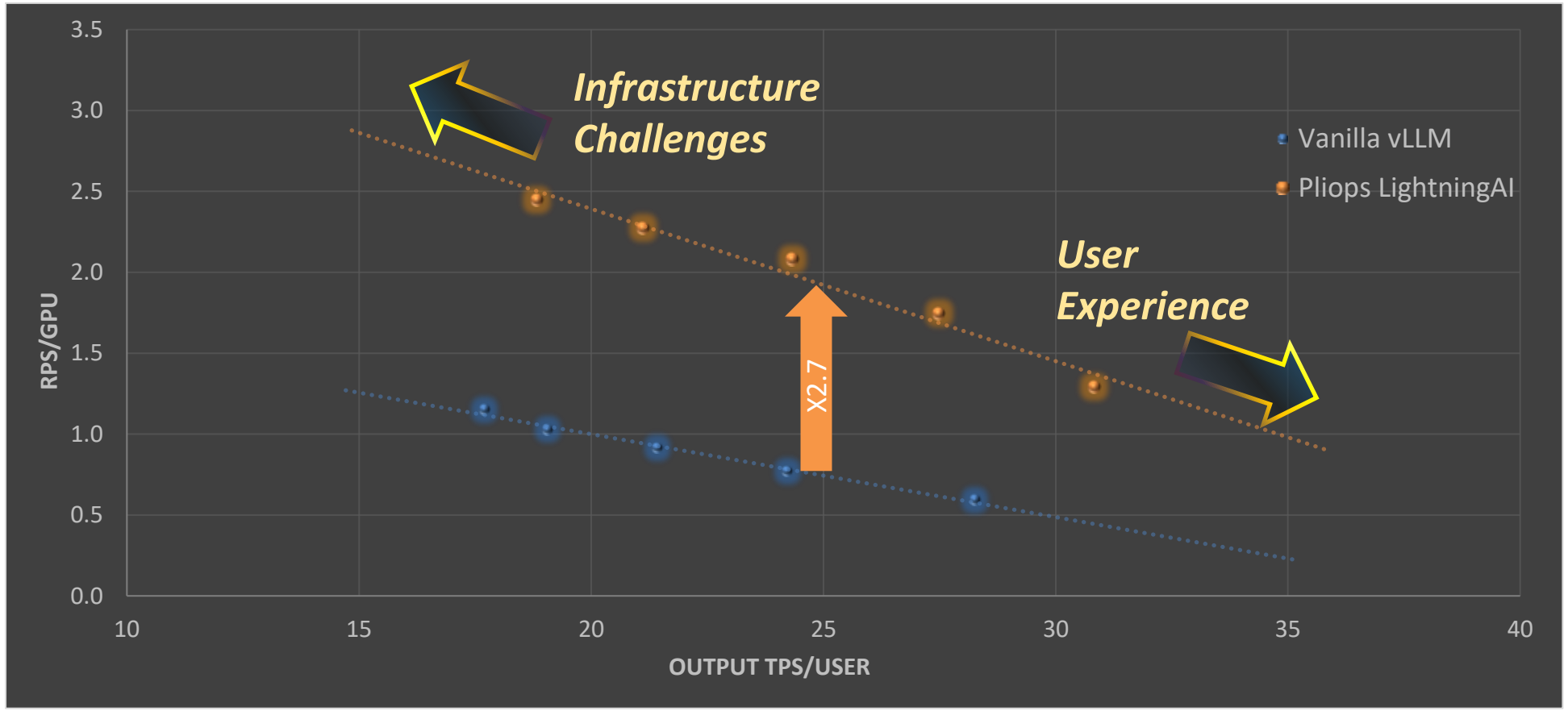
https://www.ted.com/talks/eric_schmidt_the_ai_revolution_is_underhyped

Solving LLM Infrastructure Scalability Challenges

Serve **MORE** queries with **LARGER** models--**FASTER**, under the **SAME POWER** budget



System Efficiency vs User Experience Tradeoff



- HW: H100 gen 5 Dell server
- Model: llama-3-70B, 8FP, GQA 8, TP 2
- Prompt: 100-3500, output: 64
- Different batch sizes

Agenda for Today's Talk

- ❑ Why prefill and KV-cache dominate E2E performance
- ❑ Shared KV-store approach (Disaggregated)
- ❑ Results, theoretical model and implications for system design

Popular LLM Applications w Repetitive Context

Chatbots

reuse early chat content as context for later chat input

Bug Fixing

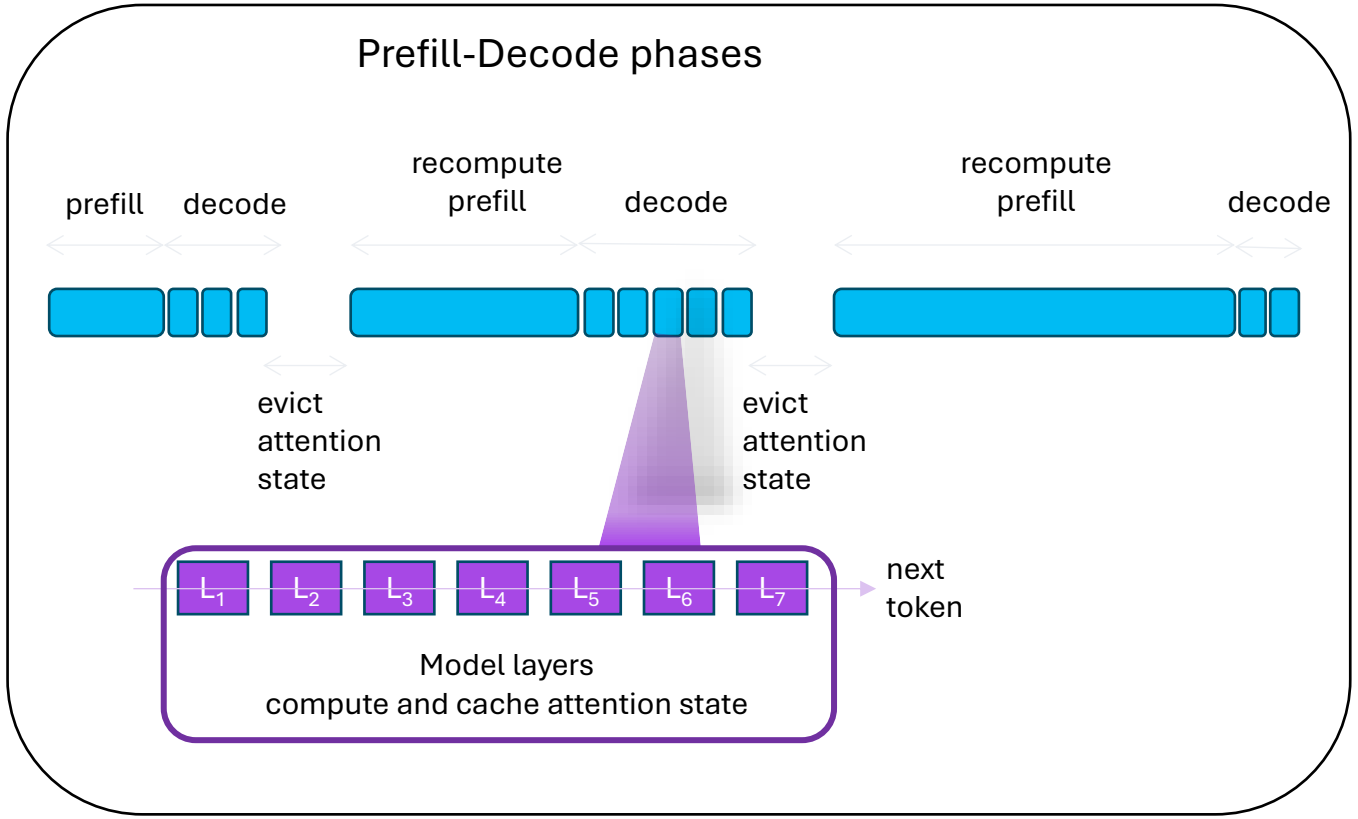
Frequently using entire code repository as context

Insights

Assistant and rec sys query large document sets

Repetitive computations * highly inefficient * up to 99% prefill cost * limit HBM bandwidth utilization and e2e performance

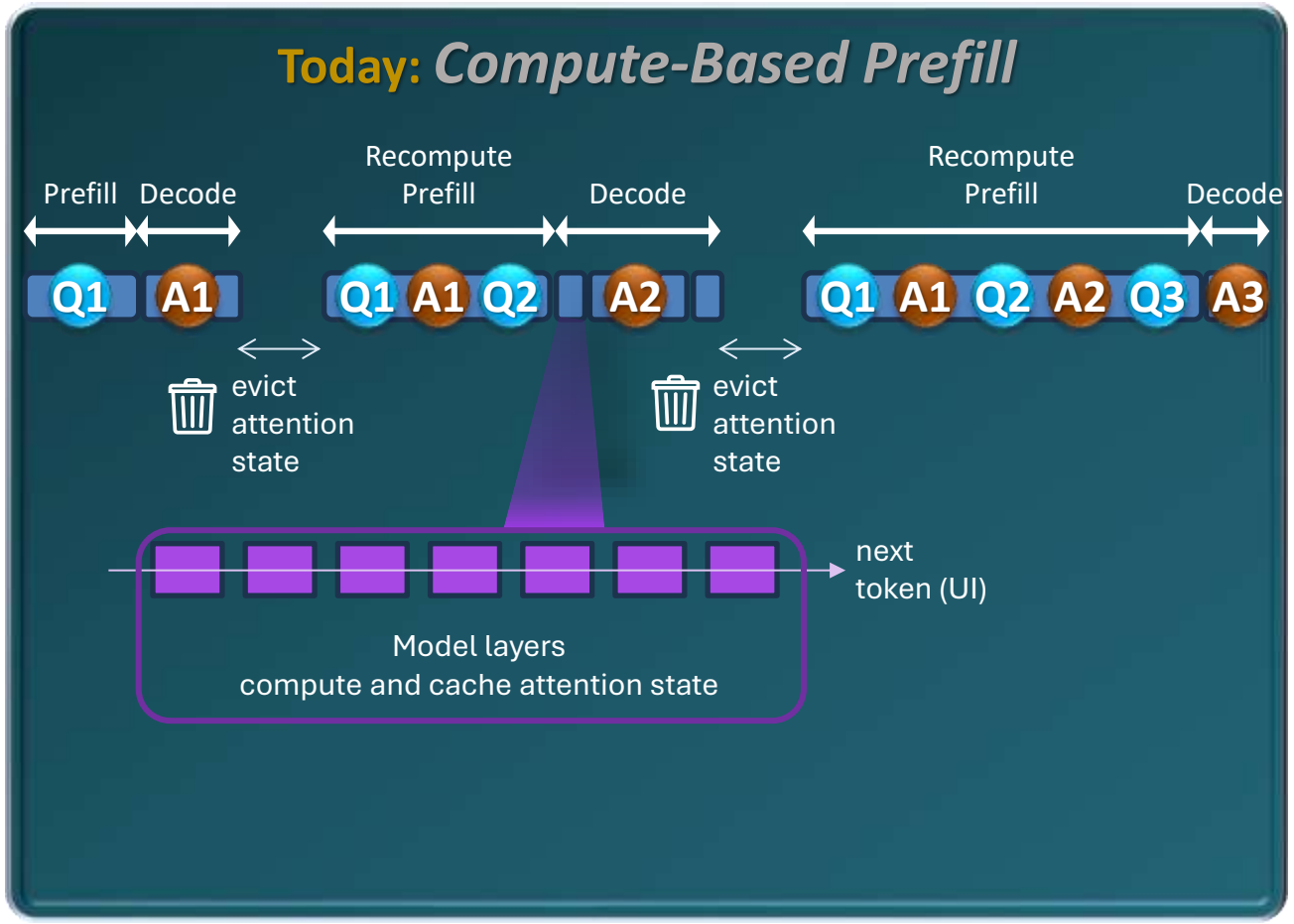
Prompt Computation vs. Token Generation



- Prefill phase (prompt)
  - All input tokens processed in parallel to generate the first output token
  - Time to first token (TTFT)
  - Compute bound
- Decode phase (token)
  - Serialized token generation
  - Time per output token (TPOT)
  - Memory bound

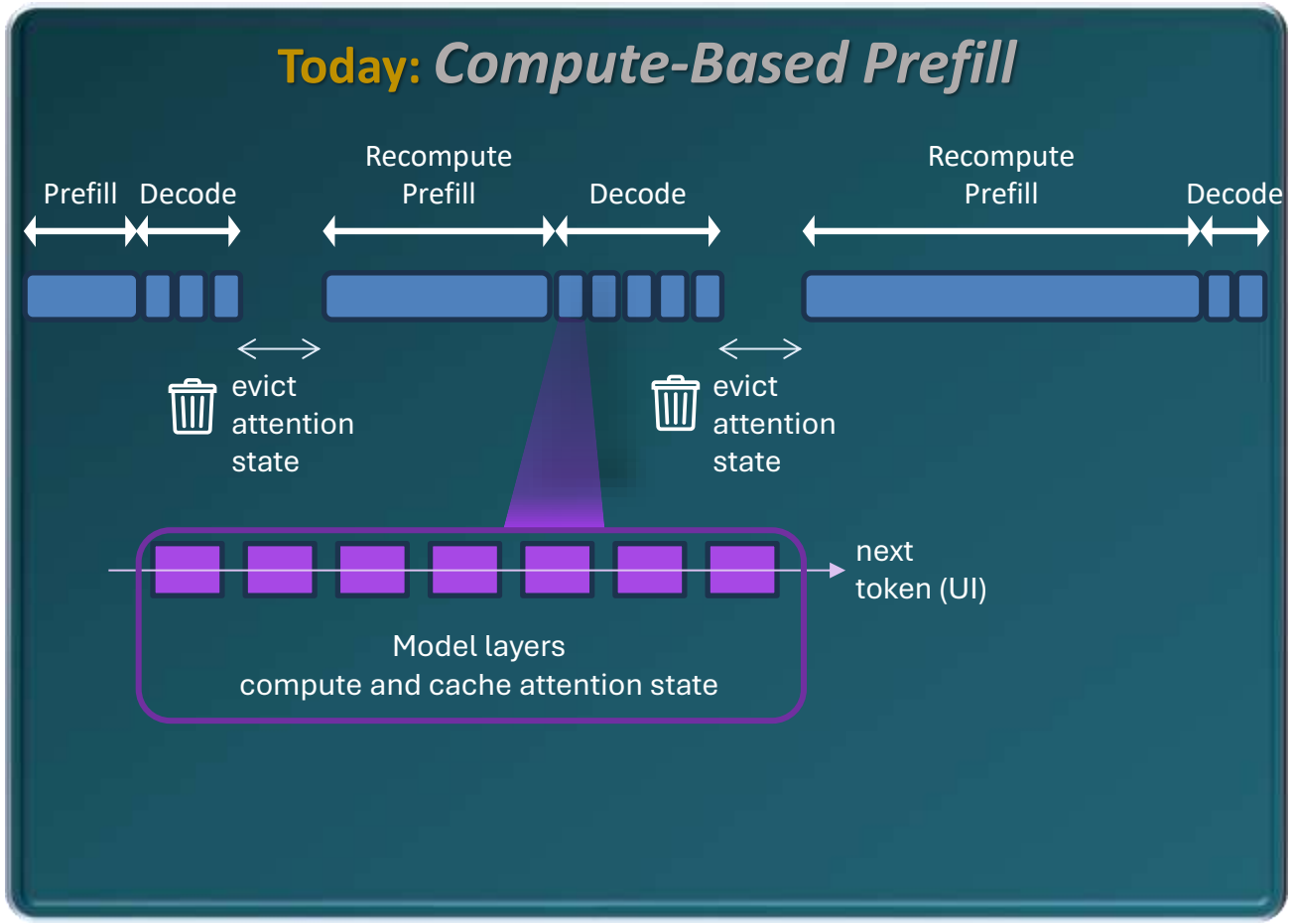
Multi-Turn Chats Evict KV-Cache Between Turns

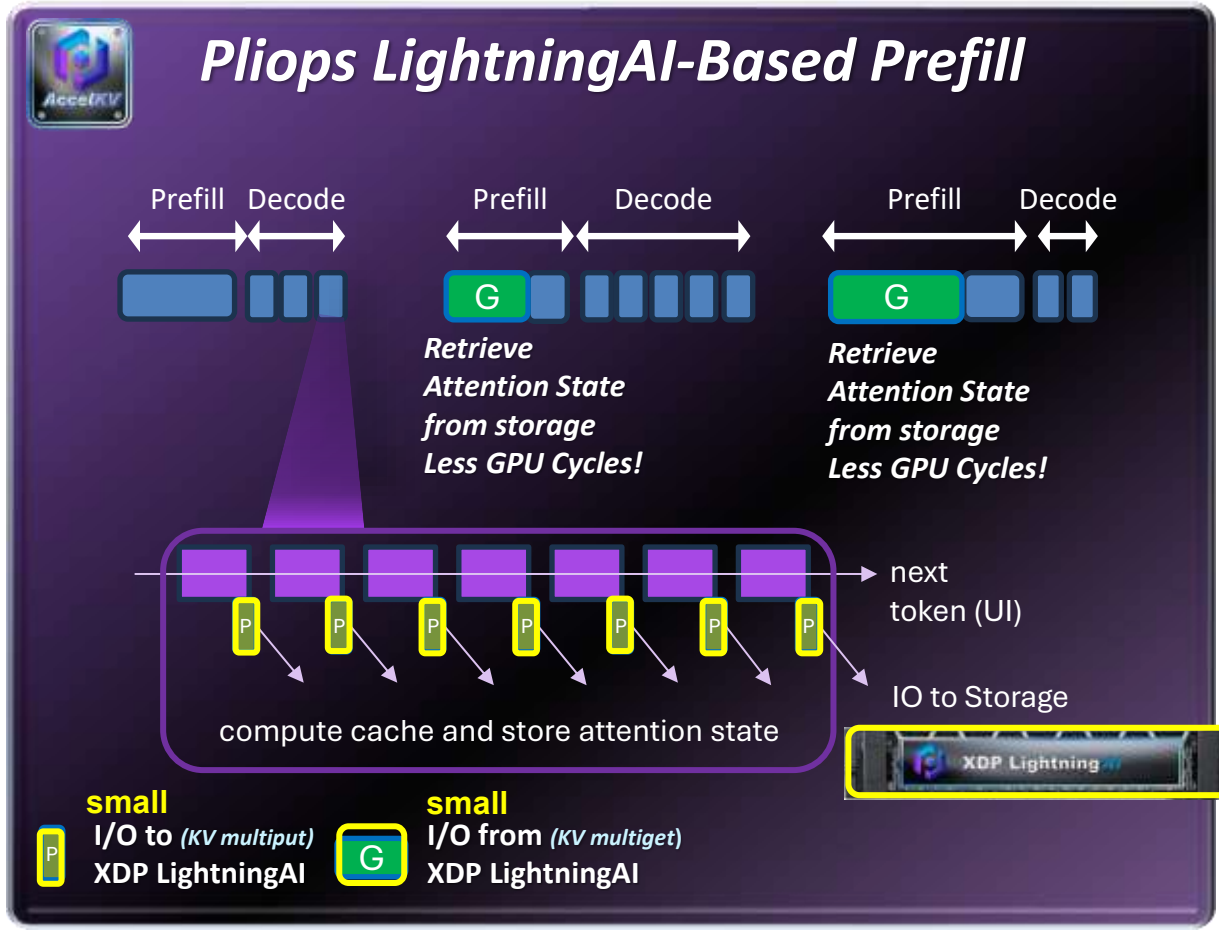
Multi-turn conversation or multi-shot task agent prefill attention kv-cache based on expanding history



KV-Cache Offloading in a Nutshell

Multi-turn conversation or multi-shot task agent prefill attention kv-cache based on expanding history

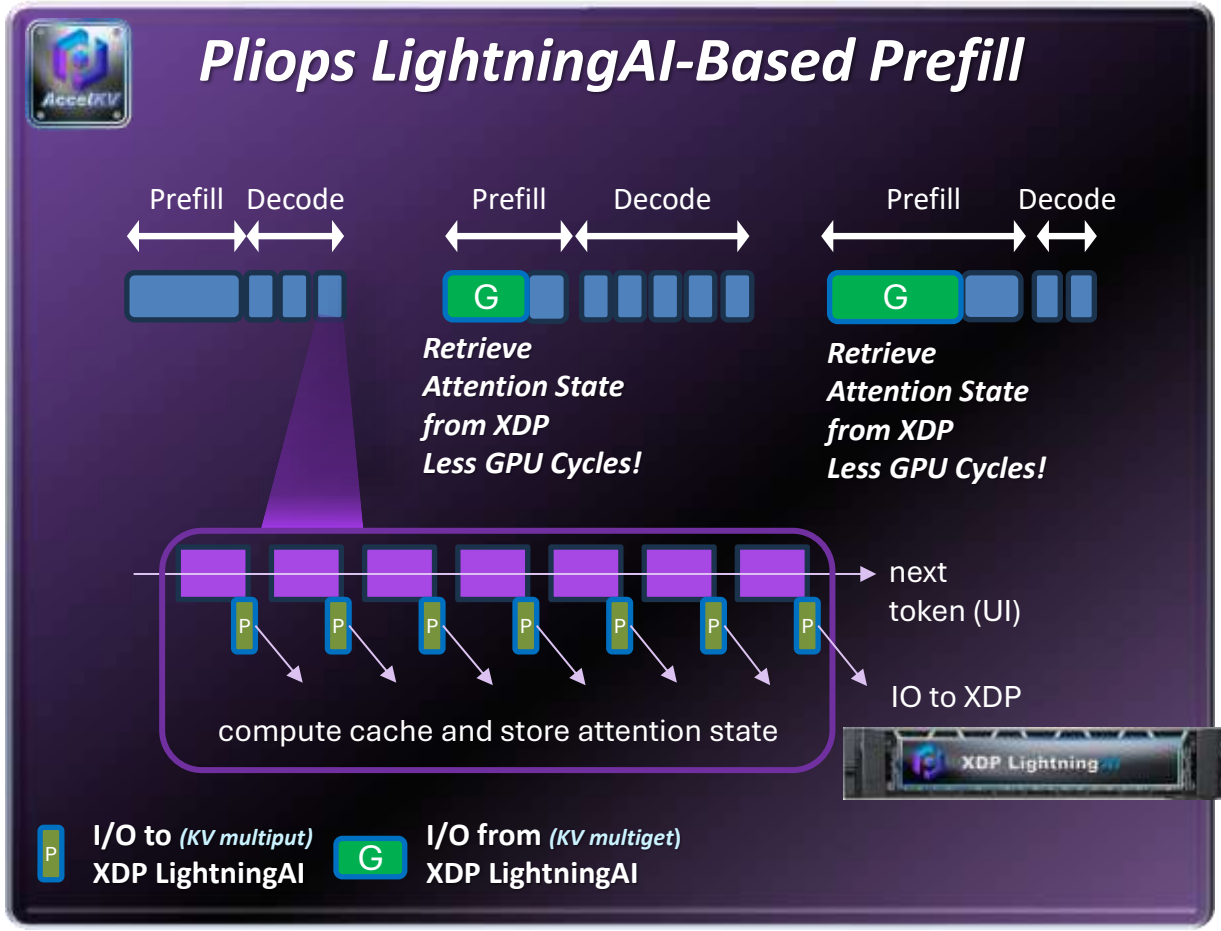




Pliops HW KV Solution is Unique

Multi-turn conversation or multi-shot task agent prefill attention kv-cache based on expanding history

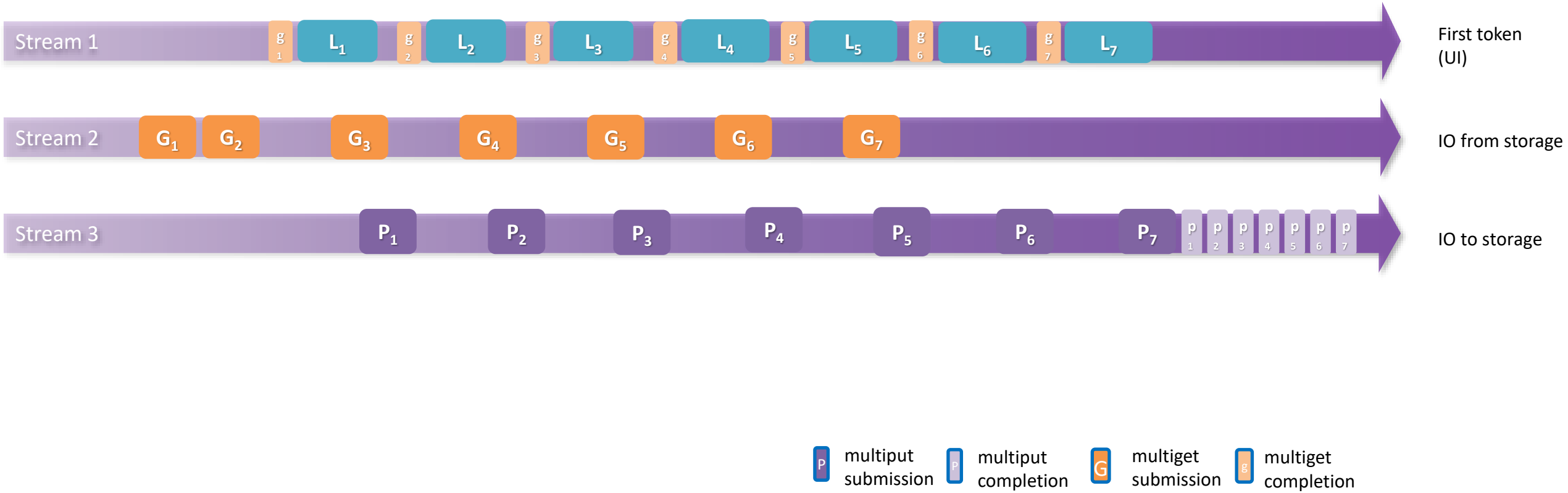
- GPU-initiated IO
- Disaggregated KV-Store
- Avoiding CPU in the critical path



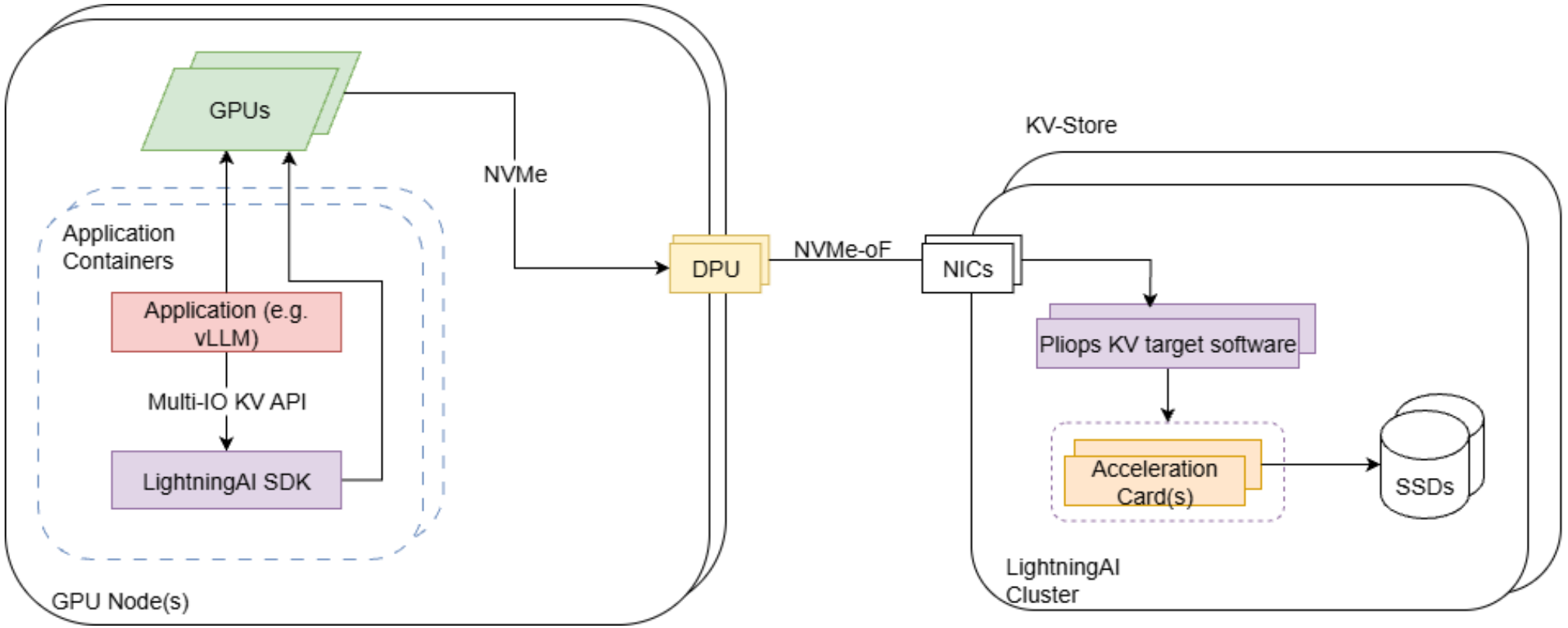
Pliops KV ~~File System~~ **saturates the fabric with small random IO supports variable size compressed tensors**

Overlapping IO w/ Compute

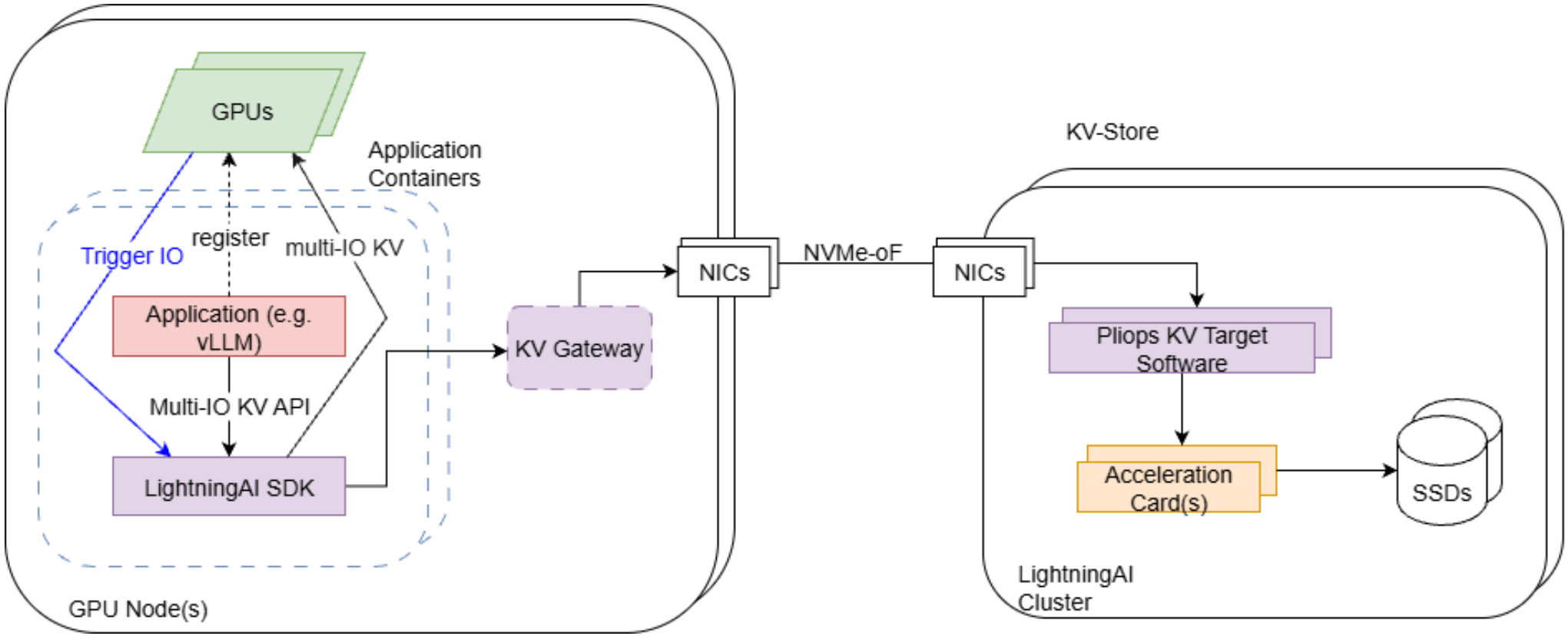
Prefill attention KV-cache: (2) retrieve past (1) compute new interaction (3) store new



GPU-Initiated IO Architecture: DPU-Based



GPU-Triggered IO Architecture: DPU-less



Prefill Acceleration By KV-Cache Retrieval

R - time to retrieve a token from storage

- $R = |KV_{token}|/BW_{IO}$

T - compute time per input token

- $T = (2|model| + N_{in} \cdot O(\sqrt{|model|}))/TFLOPs(*)$

Effective token acceleration $e = T/R$

α – KV-cache fraction cached in storage

N_{in} – number of input token

(*) approximate attention compute; assuming all GPU compute is utilized, not always true

Prefill Acceleration Analysis

$$TTFT^{V} = N_{in} \cdot T$$

Offloading writes the newly generated KV cache

- Let **W** denote the write time of a single token KV

Assuming IO and compute can be overlapped

- At each layer prefetch next layer's KV cache and write the KV cache of the previous layer
- $TTFT^{KV} = \max(\alpha \cdot N_{in} \cdot T/e \ , (1-\alpha) \cdot N_{in} \cdot W, (1-\alpha) \cdot N_{in} \cdot T)$

IO time to fetch existing KV cache from storage

IO time to write the new part of the prompt

Compute time of the new part of the prompt

Prefill Acceleration Analysis (Cont.)

$$TTFT^{KV} = \max(\alpha \cdot N_{in} \cdot T/e \ , (1-\alpha) \cdot N_{in} \cdot W, (1-\alpha) \cdot N_{in} \cdot T)$$

When $W \leq T$ TTFT is not affected by write IOs

$$TTFT^{KV} = \max(\alpha \cdot N_{in} \cdot T/e \ , (1-\alpha) \cdot N_{in} \cdot T)$$

$$TTFT^{V} = N_{in} \cdot T$$

Insight
Write only need to match compute performance

Compute-bound: speedup is x=$1/(1-\alpha)$

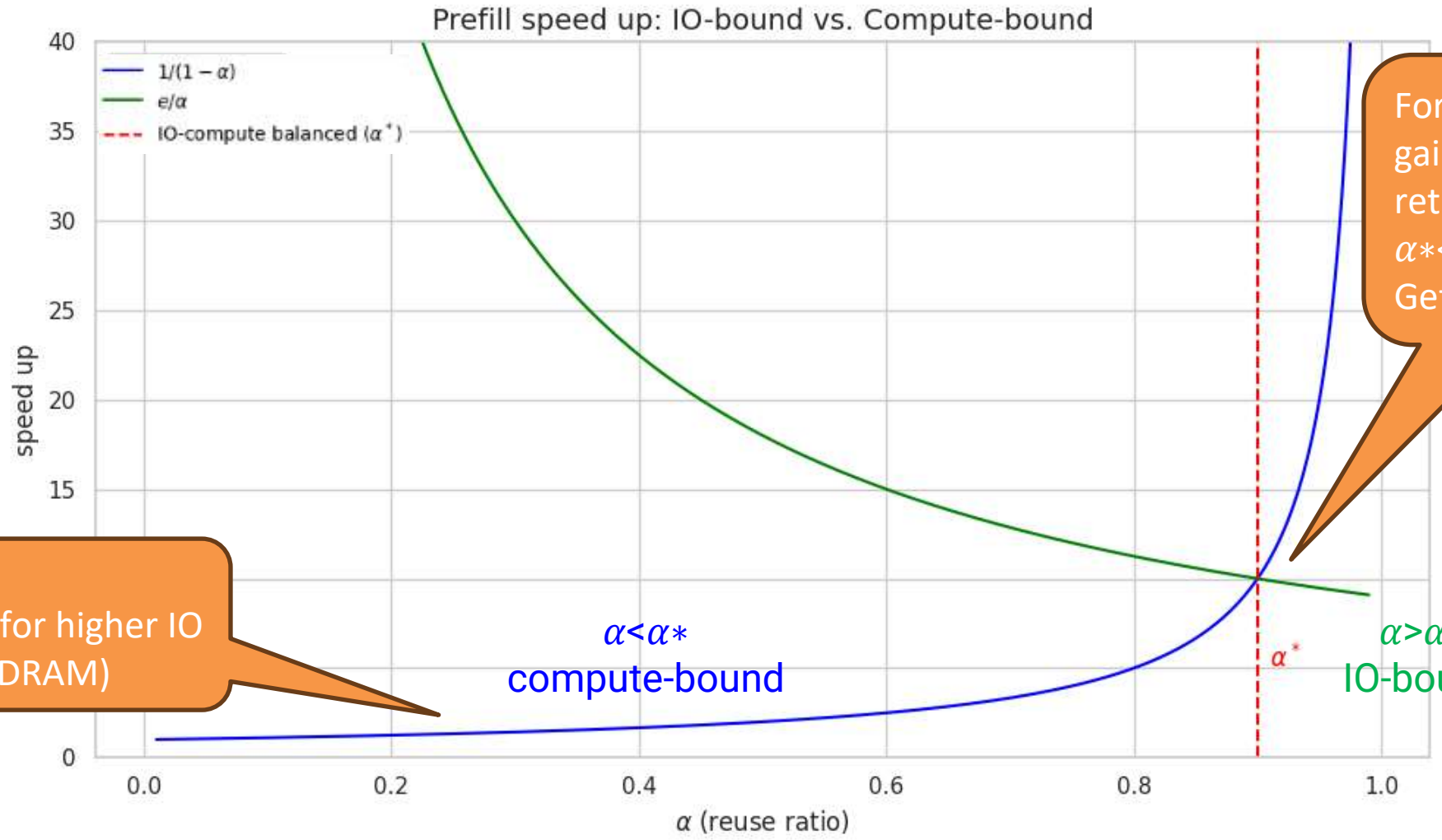
IO-bound: speedup is x=e/α

Acceleration depends on read performance and hit rate

Prefill Acceleration Analysis (Cont.)

For example, **e=9**
e is the acceleration from offloading (T/R)

Crossover point
$\alpha*=e/(1+e)$



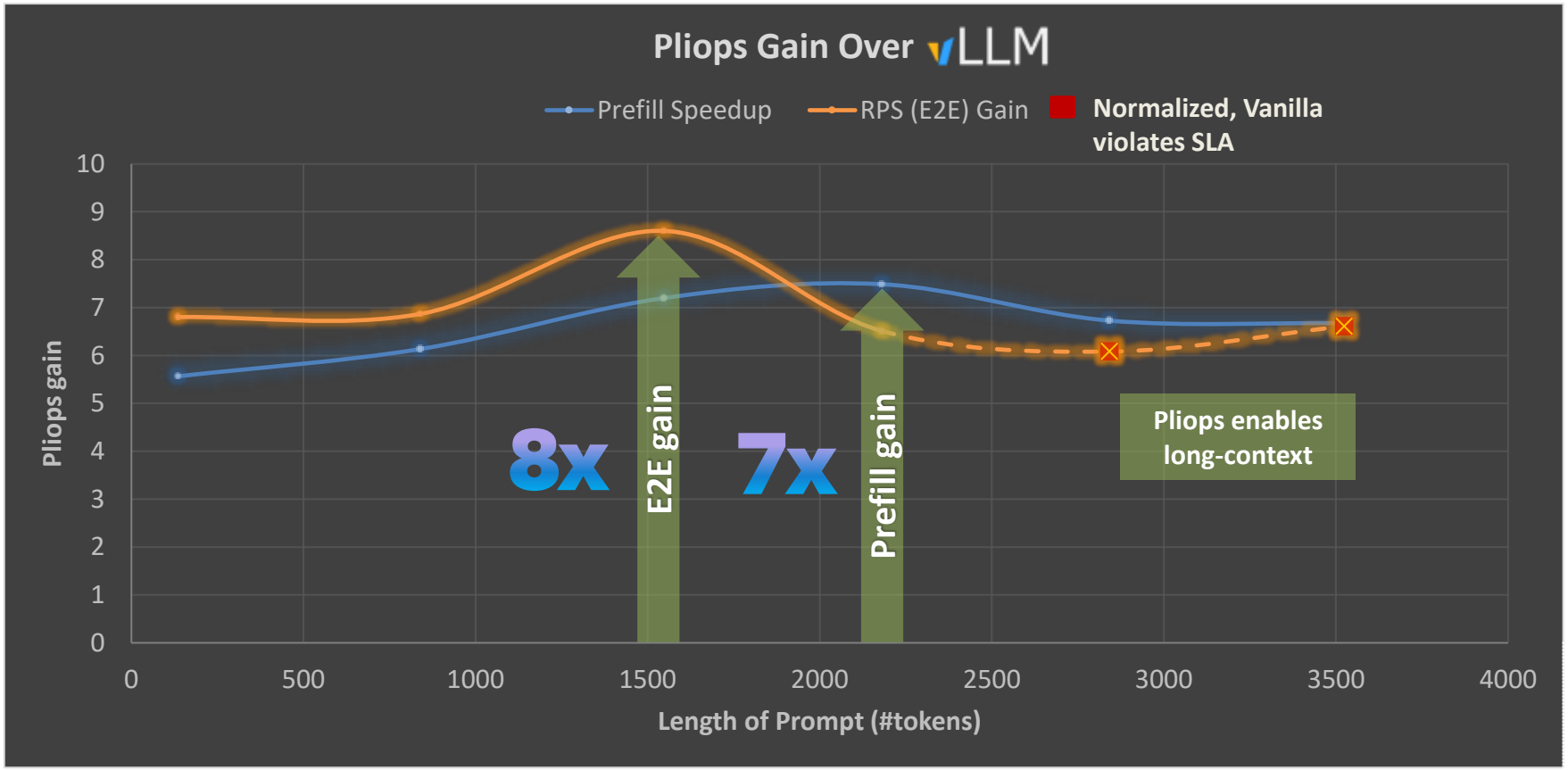
For maximal gain,
retrieve only
$\alpha*< \alpha$
Get ***e*+1** speedup

Insight
No advantage for higher IO speed! (HBM, DRAM)

$\alpha<\alpha*$
compute-bound

$\alpha>\alpha*$
IO-bound

Prefill Acceleration - Compute-Bound Example

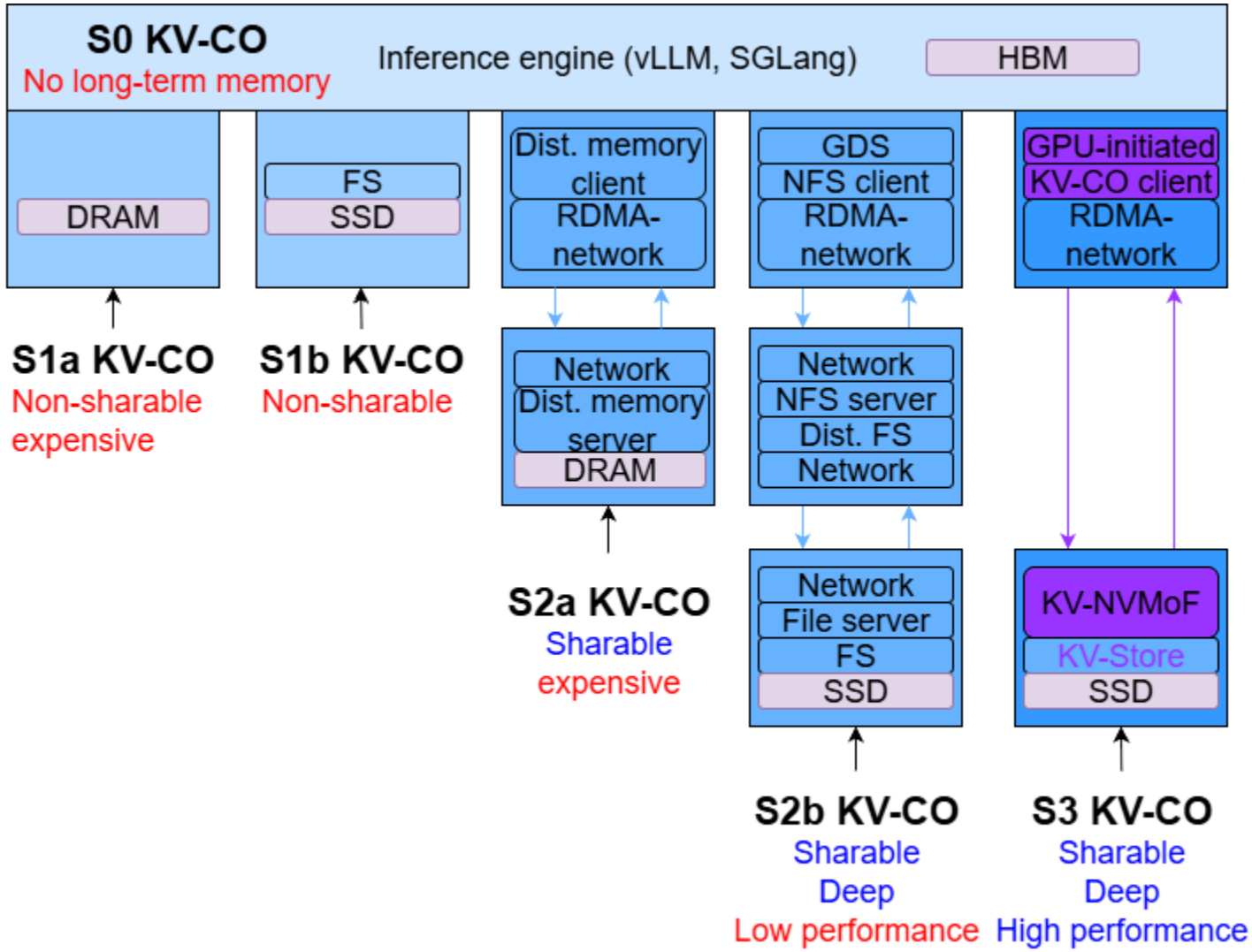


Replacing GPU compute with storage IO in prefill allows higher HBM BW efficiency in decode via larger batch size

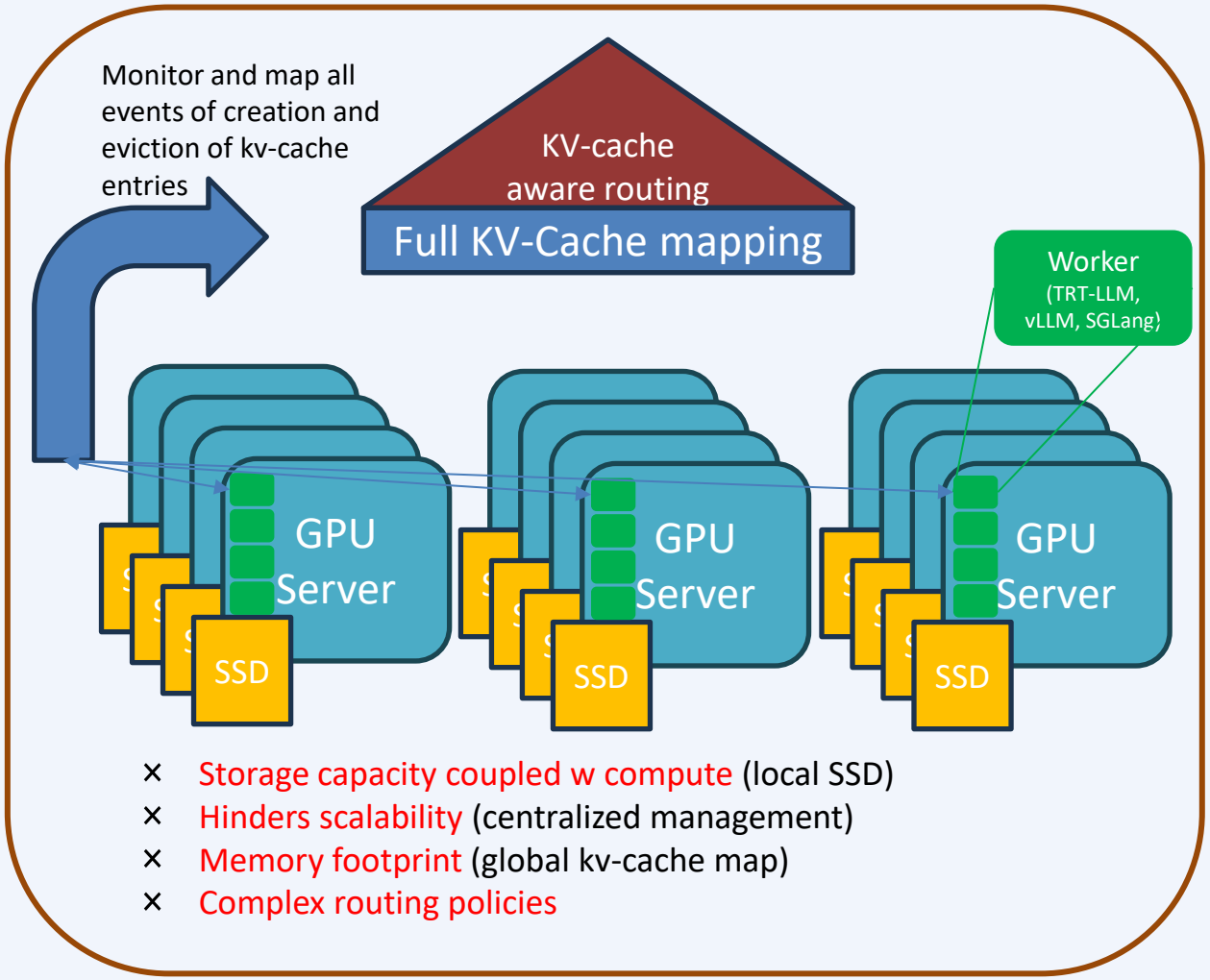
- HW: H100 gen 5 Dell server
- Model: llama-3-70B, 8FP, GQA 8, TP 1
- SLA: TTFT 400ms, TPOT 40ms
- Prompt: 100-3500, output: 64
- Maximum batch size to meet SLA

Stratified classification of KV-cache offloading (KV-CO) Solutions

- **S0:** Simple but tiny capacity
- **S1:** extended capacity, remains non-sharable, incurs higher cost
- **S2:** sharable namespace and deep at the expense of performance or cost
- **S3:** sharable namespace and deep with improved performance

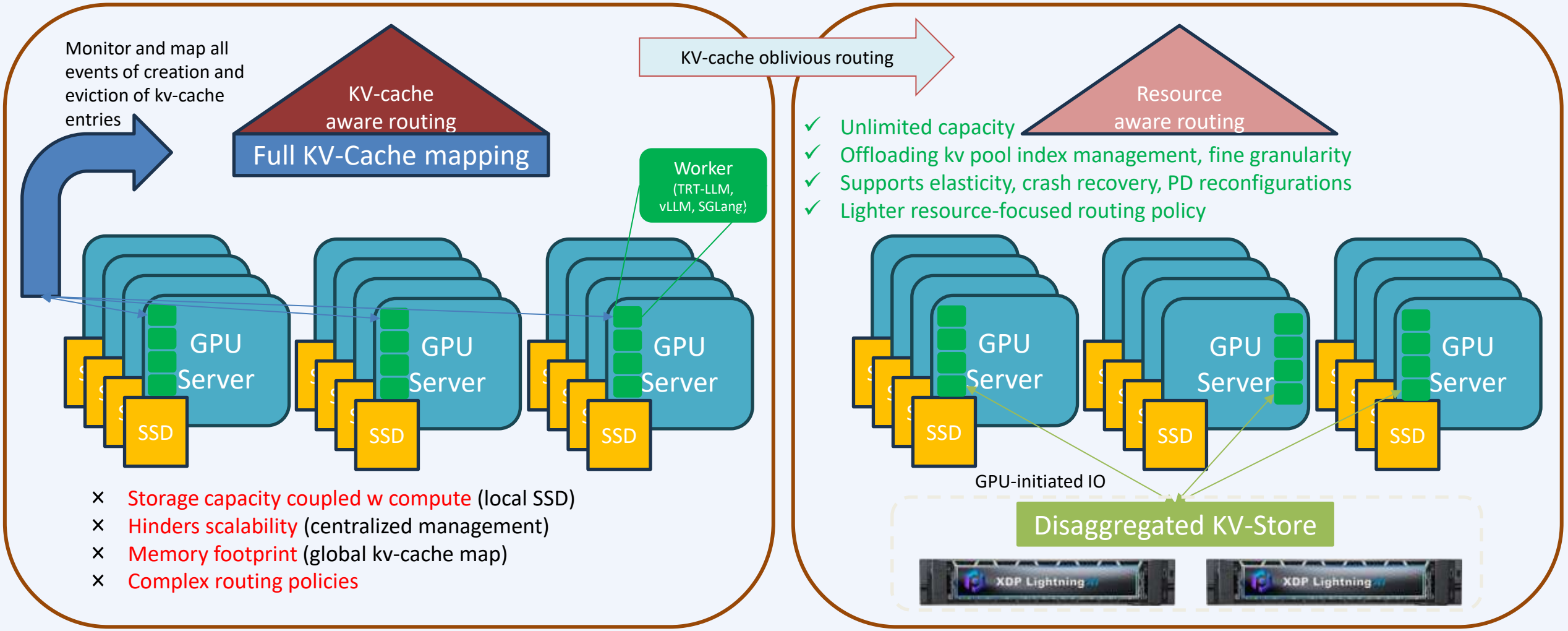


Datacenter Scale LLM Inference Framework

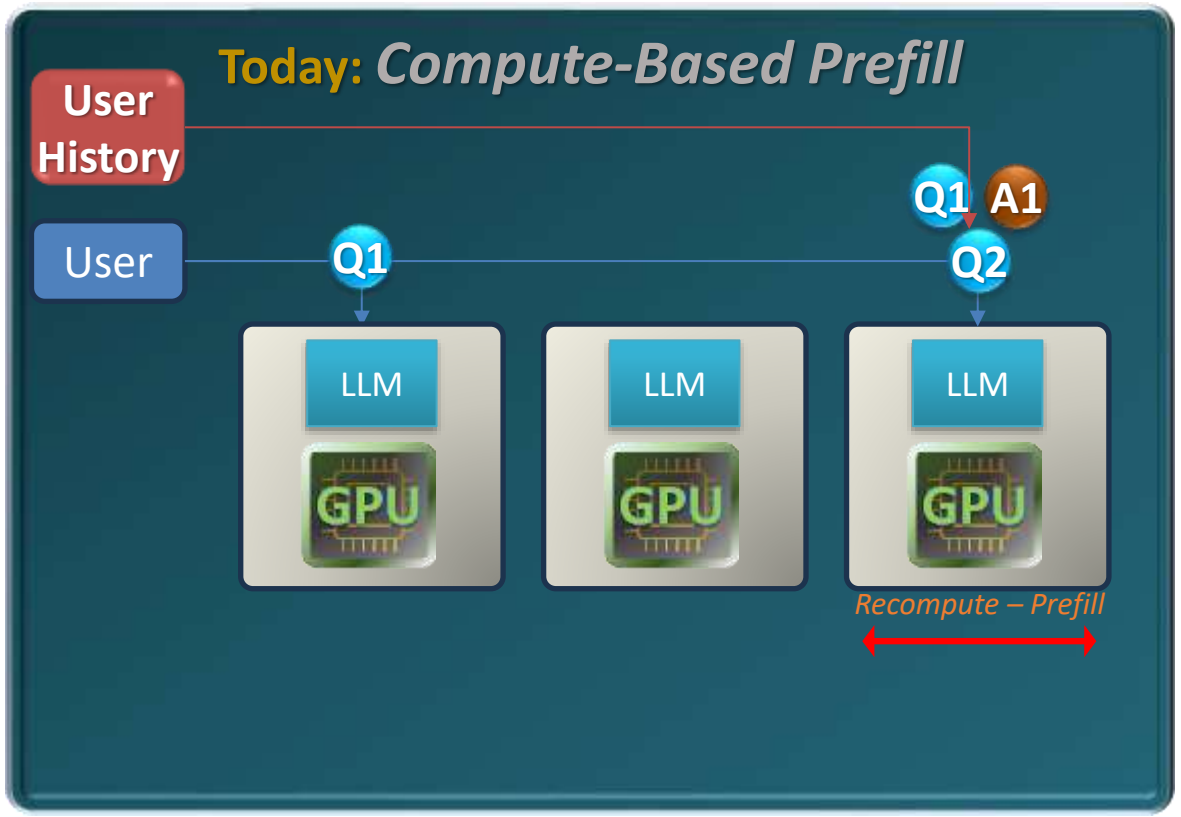


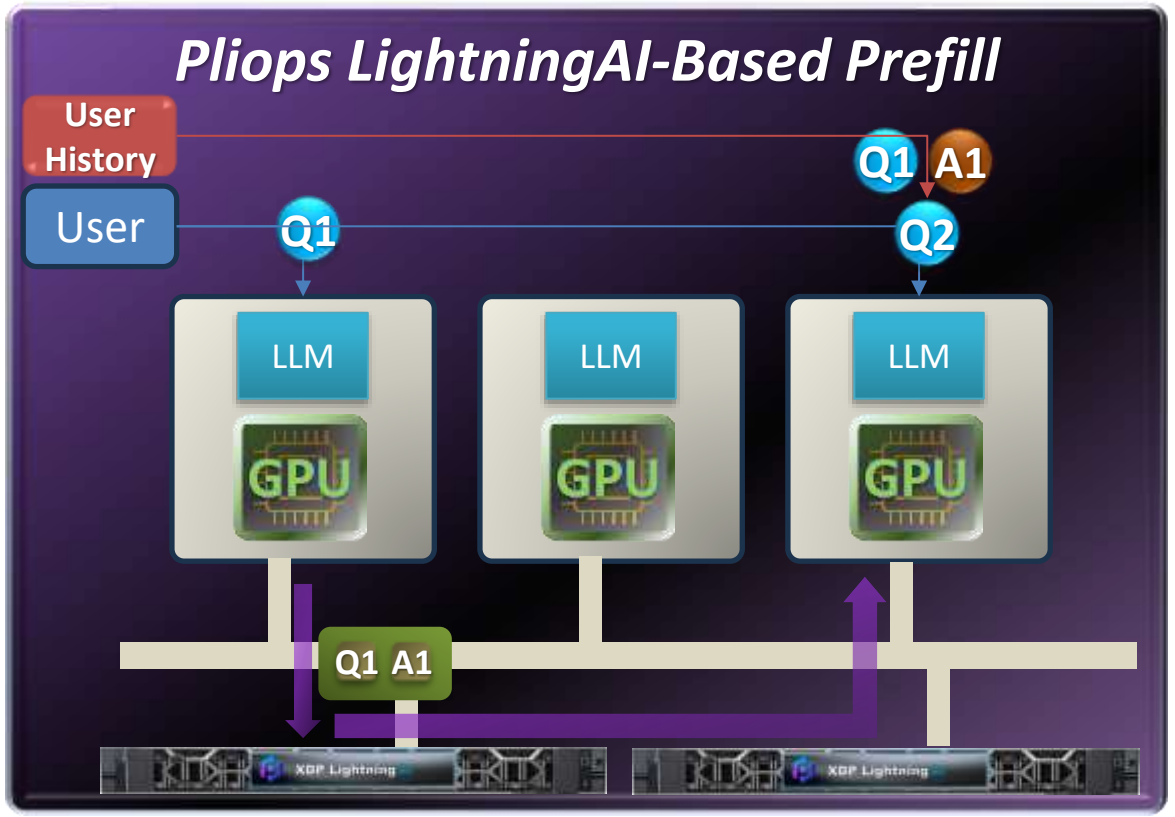
- Dynamo (Nvidia)
- llm-d (IBM/RedHat/Google)
- AIBrix (ByteDance)
- Production Stack (Tensor Mesh)

Simplicity for Performance and Robustness

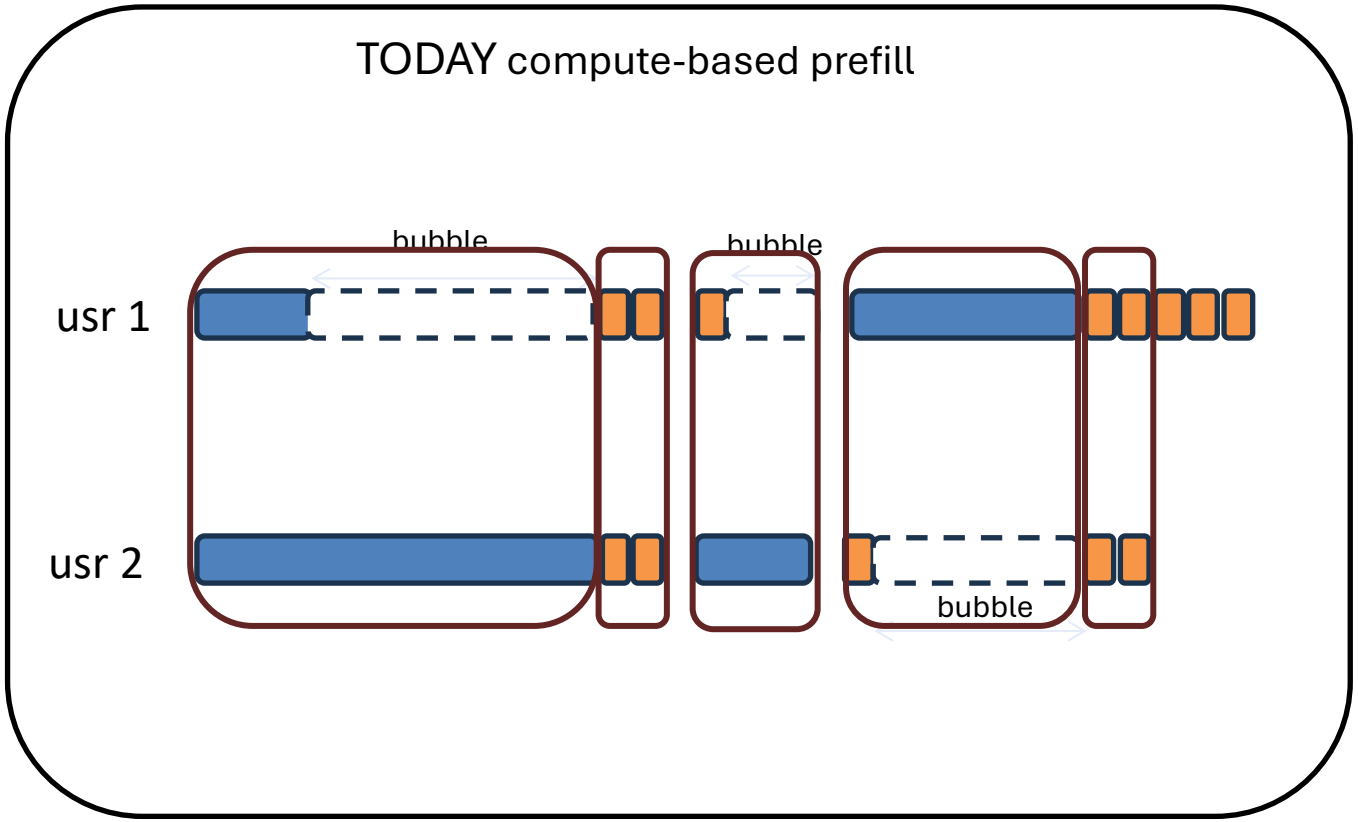


KV-Cache Offloading- System View





Colocating Prefill and Decode



- In-flight batching (continuous)
- Decode “love” batches as it is memory bw bound

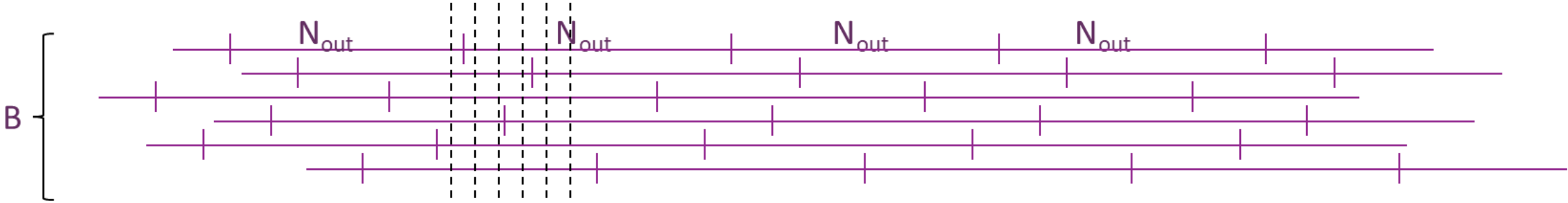
- Prefill-prefill interference
- Prefill-decode interference
- Creating “bubbles”

E2E Gain - Continuous Batching Analysis

B – number of concurrent requests

N_{out} – Average number of output tokens

Number of concurrent prefills~ $Binom(B,1/N_{out})$



E2E Gain - Continuous Batching Analysis

B – number of concurrent requests

N_{out} – Average number of output tokens

Number of concurrent prefills~ $Binom(B,1/N_{out})$

$$E[TTFT(B)] = TTFT(1) + (B-1)/N_{out} \cdot TTFT(1) = (1 + (B-1)/N_{out}) \cdot TTFT(1)$$

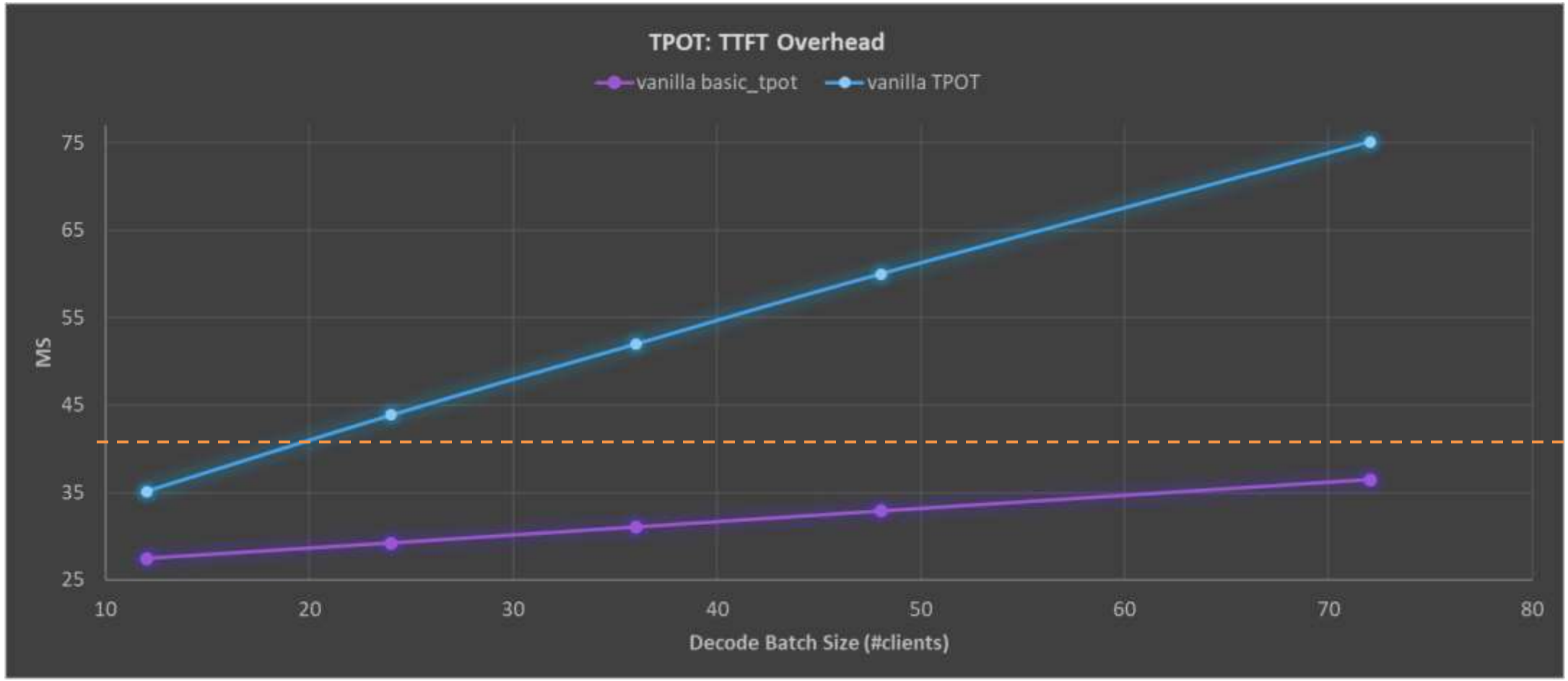
$$E[TPOT(B)] = TPOT(1) + (B-1) \cdot (TTFT(1)/N_{out} + |KV(1)|/BW_{HBM})$$

- $|KV(1)|$ is the average KV cache size of a single input prompt
- BW_{HBM} is the memory BW

Expected number of additional simultaneous prefills in a prefill slot

Time to transfer a single prompt KV cache to compute engines

Measured TPOT Overhead for LlaMa-3-70B



Impact of Prefill Speedup on Decode Speedup

ΔP ΔH

$$TPOT(B) \sim= TPOT(1)+(B-1)\cdot(TTFT(1)/N_{out}+|KV(1)|/BW_{HBM})$$

$TPOT(B) \leq SLA_{TPOT}$

System is required to meet SLA

Maximal B to meet SLA_{TPOT} : $\lfloor (SLA_{TPOT} - TPOT(1))/(\Delta P+\Delta H) \rfloor$

When $x \to \infty$, $B_{KV}/B_{vanilla} = 1+\Delta P/\Delta H$

Insight
TPS gain

Asymptotic E2E Gain Analysis

Model params
Model size
KV-cache compression (GQA, MQA, MLA)

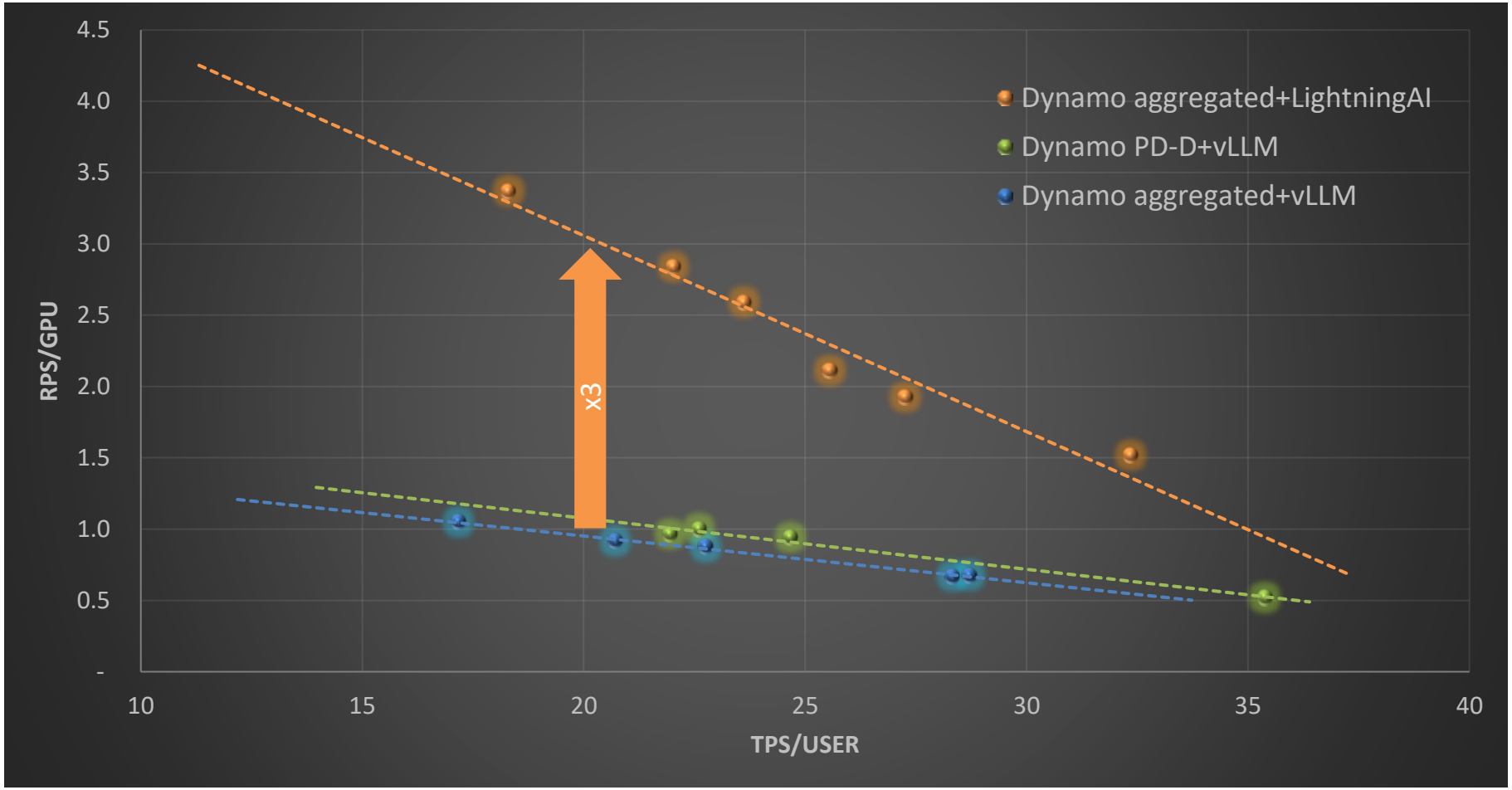
GPU params
Mem bw to compute ratio

App params
in/out ratio

Short prompt regime : $\Delta P/\Delta H \propto |model|/|KV_{token}| \cdot BW_{HBM}/TFLOPs \cdot 1/N_{out}$ ($N_{in} \ll 6d$)

Long prompt regime : $\Delta P/\Delta H \propto \sqrt{|model|/|KV_{token}|} \cdot BW_{HBM}/TFLOPs \cdot N_{in}/N_{out}$ ($N_{in} >> 6d$)

System Efficiency vs User Experience



- HW: H100 gen 5 Dell server
- Model: llama-3-70B, 8FP, GQA 8, TP 2
- Prompt: 2200, output: 170, turns: 15
- Different number of workers, clients

Summary & Conclusions

- ✓KV-cache offloading increases efficiency and reduces cost while maintaining user experience constraints
- ✓Throughput gains depend on the model, the GPU, and the workload
- ✓In compute-bound cases, no benefit for faster-than-SSD memory
- ✓Speedup depends mainly on read performance
- ✓GPU-initiated IO achieves full IO-compute overlap with zero CPU overhead

Eshcar Hillel
eshcar@pliops.com
linkedin.com/in/eshcar

PLiOPS
EXTREME DATA PROCESSOR

Thank you for attending!

Please remember to rate this session. You get access the presentations at
http://sniadeveloper.org/conference

SDC 25