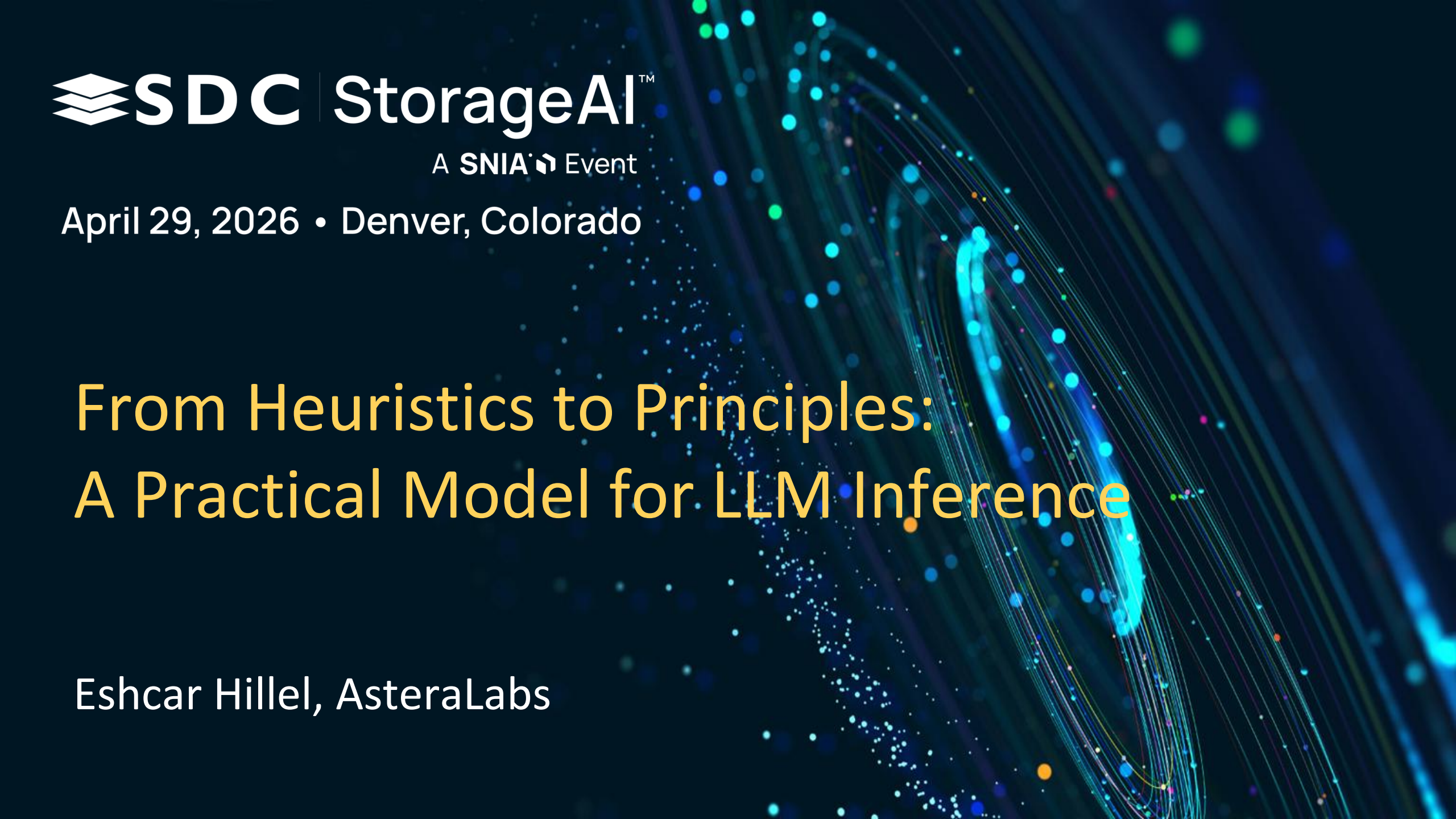


From Heuristics to Principles: A Practical Model for LLM Inference

Eshcar Hillel, AsteraLabs

Today

- What impacts data center efficiency and user experience?
- Mathematical model to capture behavior and performance of contemporary inference architectures
  - *What is the optimal PDD GPU partitioning between prefill and decode?*
  - *How to determine the chunked prefill budget?*
  - *How much efficiency gain over aggregated architecture can be achieved via PDD? Via chunked prefill? Via* KV-CO prefill acceleration*?*
- Focus on insights and predicted gain over baseline

LLM Applications w Long Repeated Context

RAG

Retrieved chunks from a shared knowledge store

Code Generation

Frequently using entire code repository as context

Long Term Memory

Personal assistant
Personal context history

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

Future of GenAI Systems and Data Centers

- Agentic workloads, reasoning, larger models, larger prompts
- What is expected in 2 years ??

"What it's going to be like when it [AI] really does

***remember every detail of your entire life**, and personalize across all of that, ... not just the facts but also the small preferences .."*

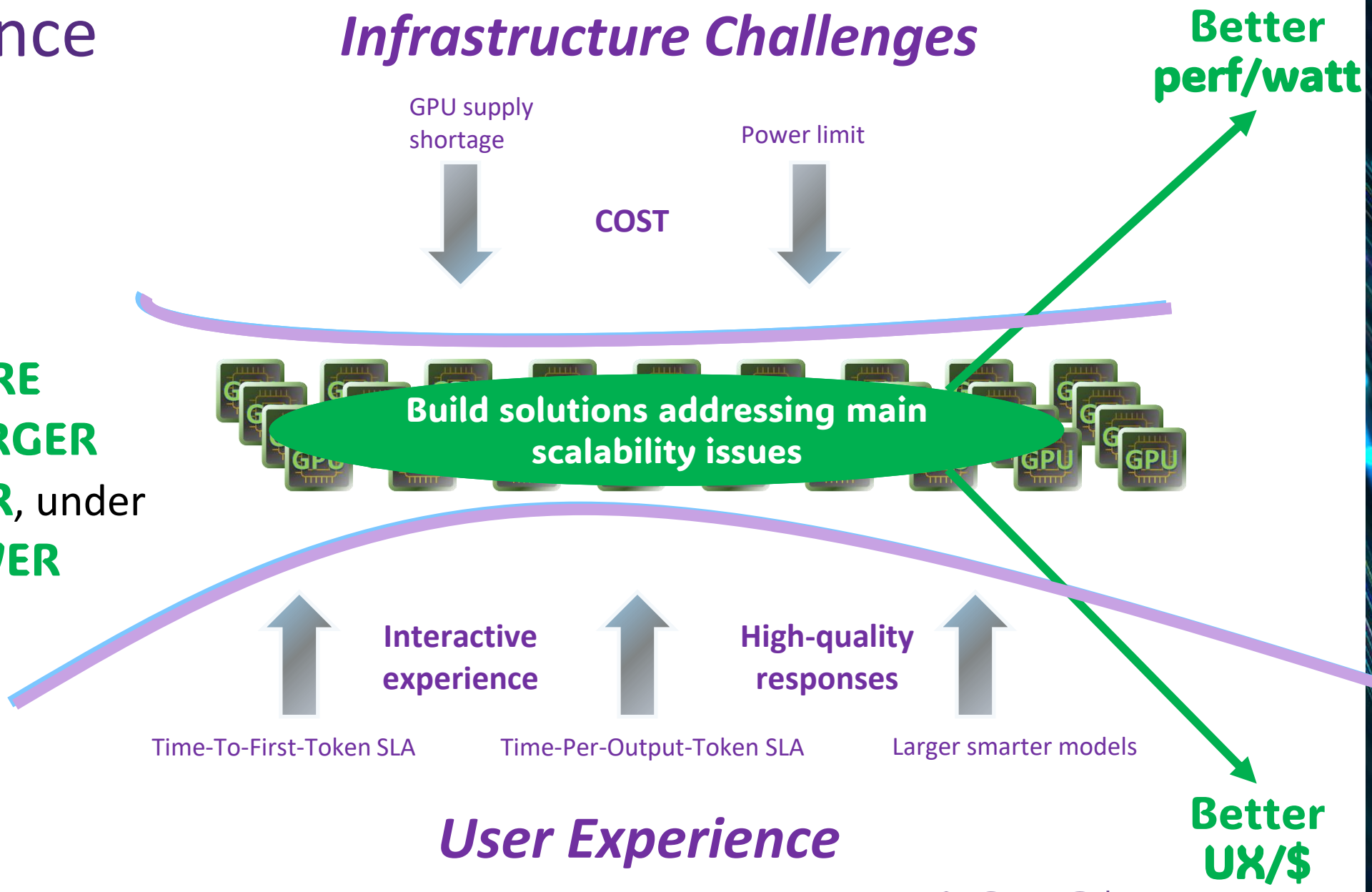
- Sam Altman, OpenAI CEO

https://indianexpress.com/article/technology/artificial-intelligence/openai-ceo-sam-altman-ai-memory-personalised-10430008/

Infinite memory matters more than smarter AI reasoning

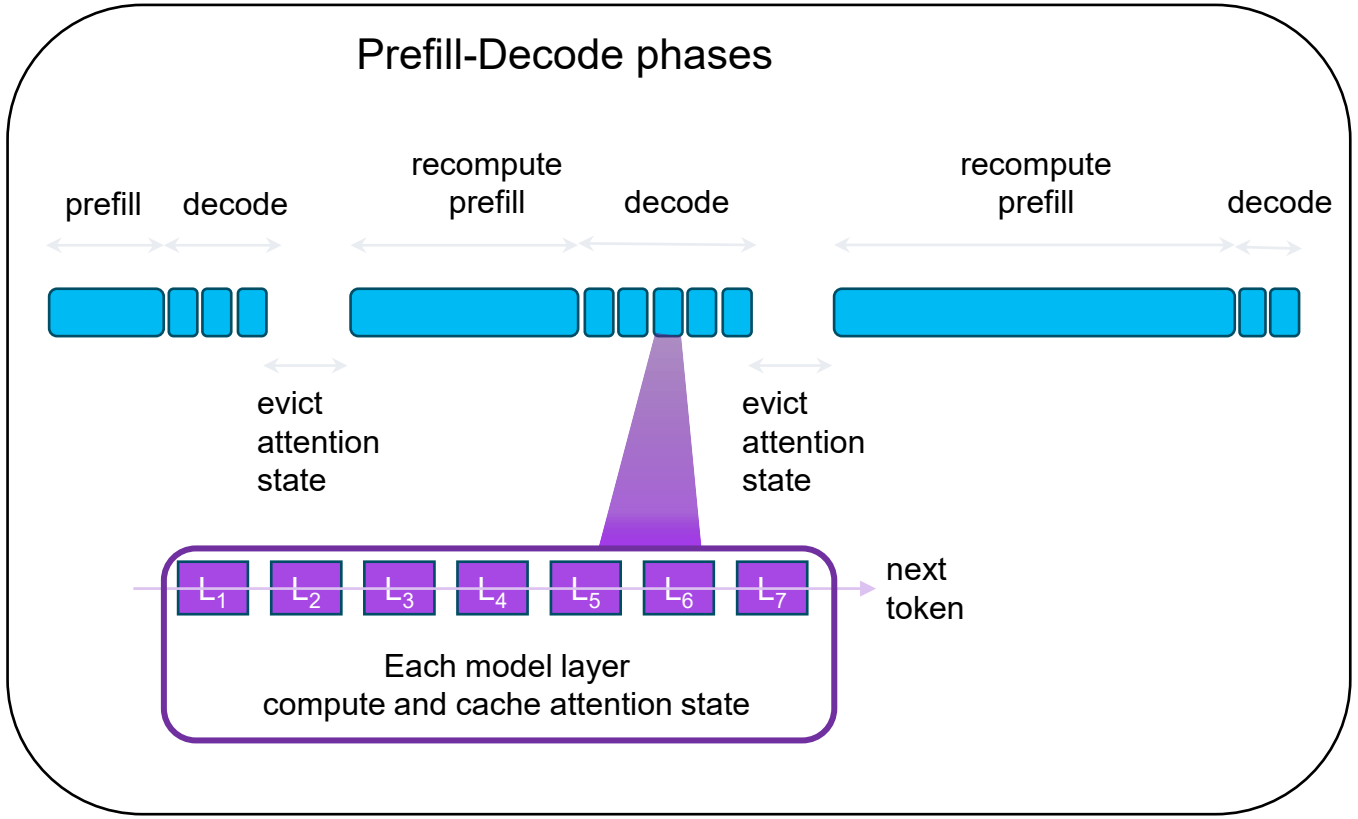
LLM Inference Scalability Challenges

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



SDC | StorageAI™

Background: Prefill and Decode (reminder)



- **Prefill** phase (prompt)
  - All input tokens processed in parallel to generate the first output token
  - KPI = Time to first token (**TTFT**)

➔ Compute bound

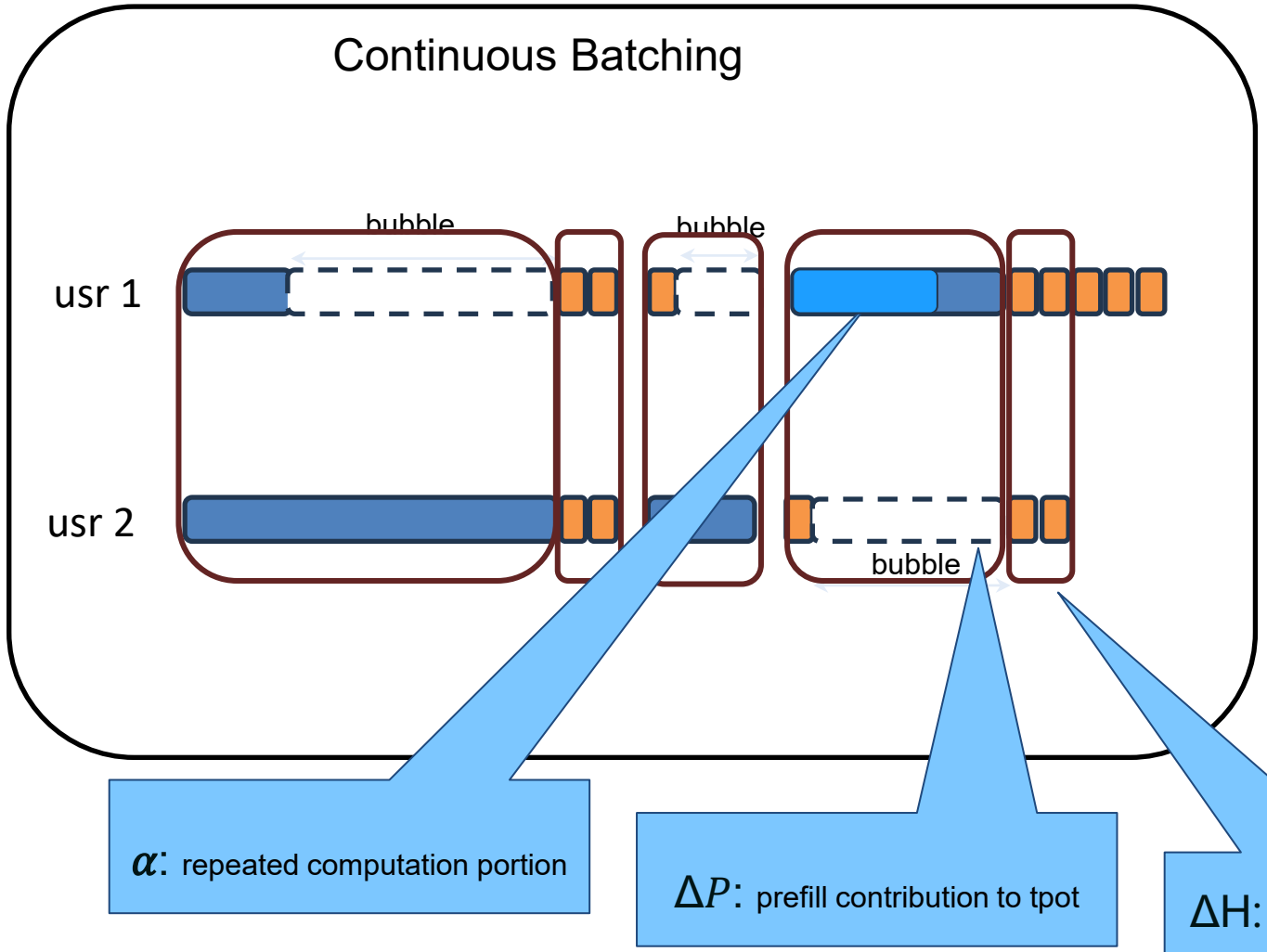
- **Decode** phase (token)
  - Serialized token generation
  - KPI = Time per output token (**TPOT**)

➔ Memory bound

System Efficiency and UX Factors

- Model and KV-cache
  - Model size (parameters count), model dimension, KV-cache compression (GQA, MLA, TurboQuant), precision
- Workload and app characteristics
  - Input length, output length
  - Time-To-First-Token and Time-Per-Output-Token SLAs
- GPU parameters
  - Peak memory bandwidth (MBW), peak compute (FLOP), memory capacity
  - Network bandwidth

LLM Inference Math Model Building Blocks: ΔP, ΔH

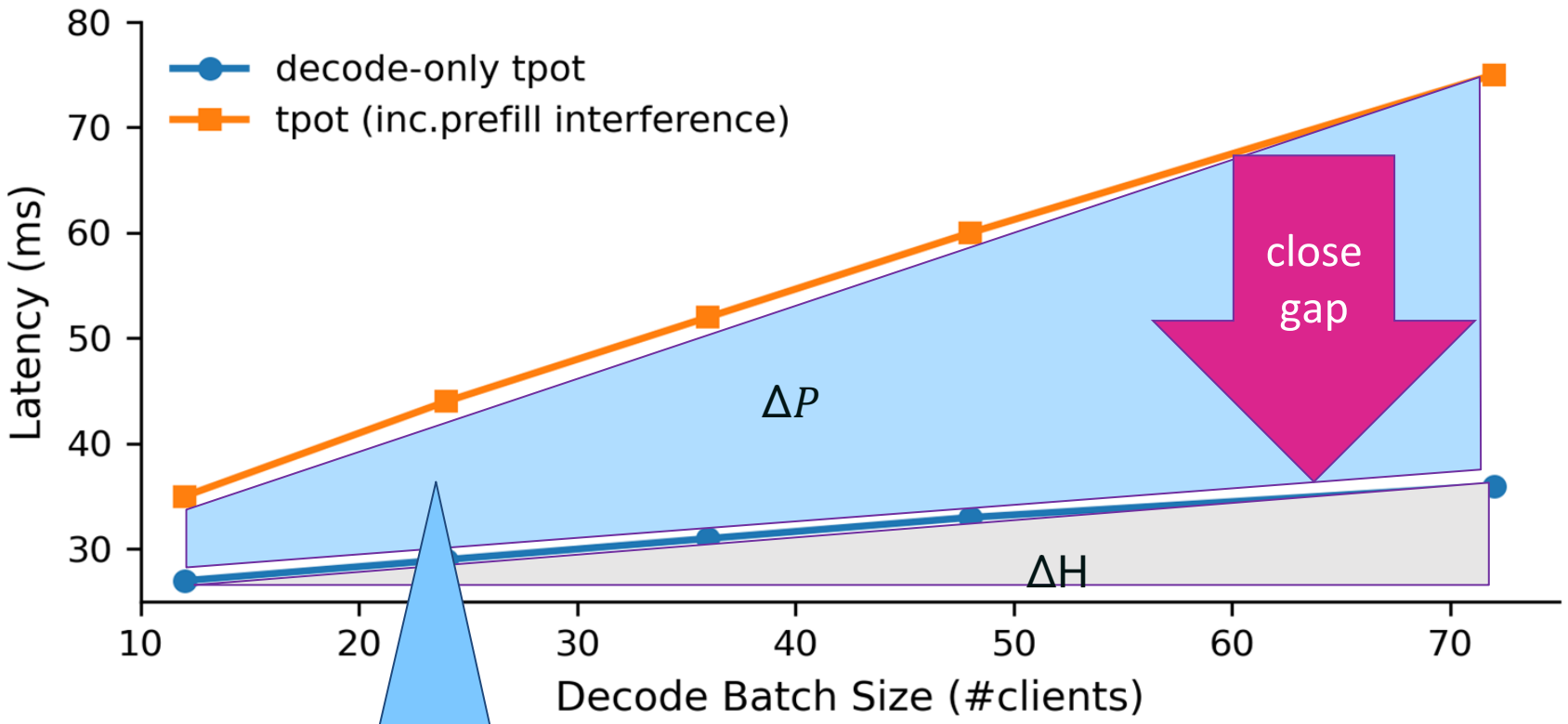


- Prefill and decode steps interleaved across requests
- P-D mutual interference
  - Resource utilization issue, Reduced efficiency, higher TCO
  - Latency implications, exceeding SLAs (UX)
- The model captures these interferences

SNIA SDC 2025 - Disaggregated KV Storage: A New Tier for Efficient Scalable LLM Inference

Measured ΔP and ΔH

- Llama3-70B on H100 GPU
- Decode phase dominates run time
- Closing the gap results in higher efficiency rps/gpu better UX/$



Main goal

The gap grows as prompt gets larger

Towards Cost-Efficient Inference Systems

Prefill-Decode disaggregation (PDD)

- Dynamo, llm-d, AIbrix, mooncake
- Separates prefill and decode execution across specialized hardware pools

Chunked Prefill

- vllm default
- Decomposing the prefill into smaller chunks, processed in parallel to concurrent decode operations

KV-Cache offloading (KV-CO)

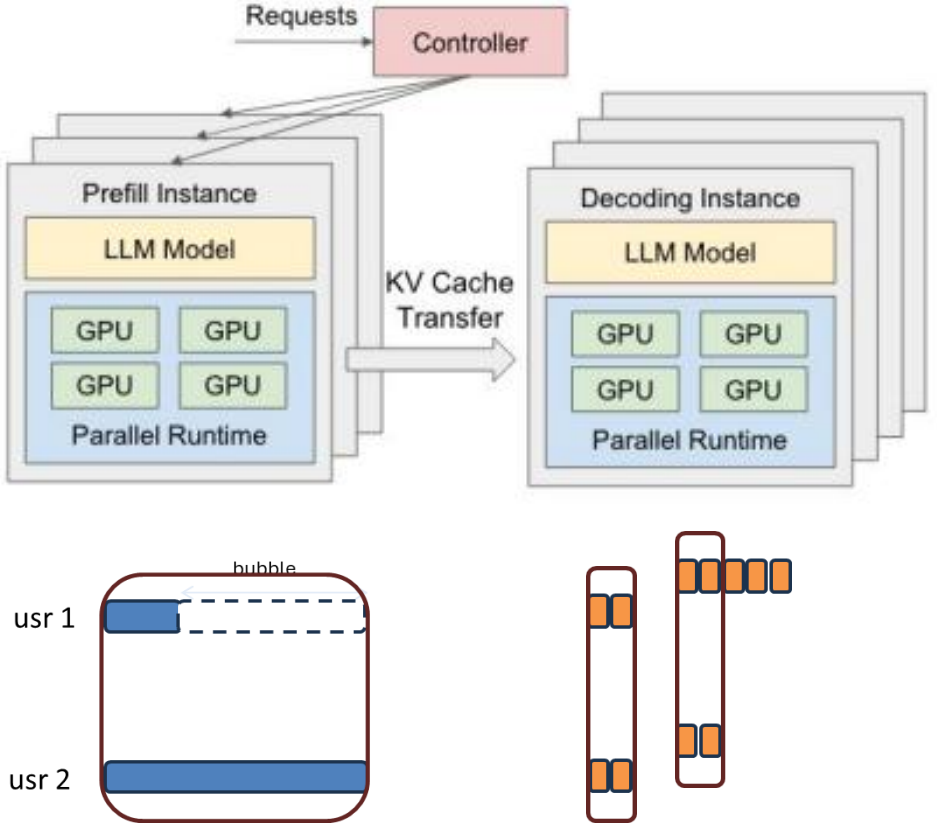
- Compute KV-cache once and reuse
- Local memory and storage
- Disaggregated KV-store: CXL memory expansion, disaggregated storage

Prefill-Decode Disaggregation

Separates prefill and decode execution

Insight
Optimal GPU partitioning
$\Delta P^p : \Delta H^d$
PDD ratio $SLA_{ttft} : Nout*SLA_{tpot}$

DistSereve https://www.usenix.org/system/files/osdi24-zhong-yinmin.pdf



False no-interference illusion

Insight
No gain w heterogeneous GPUs

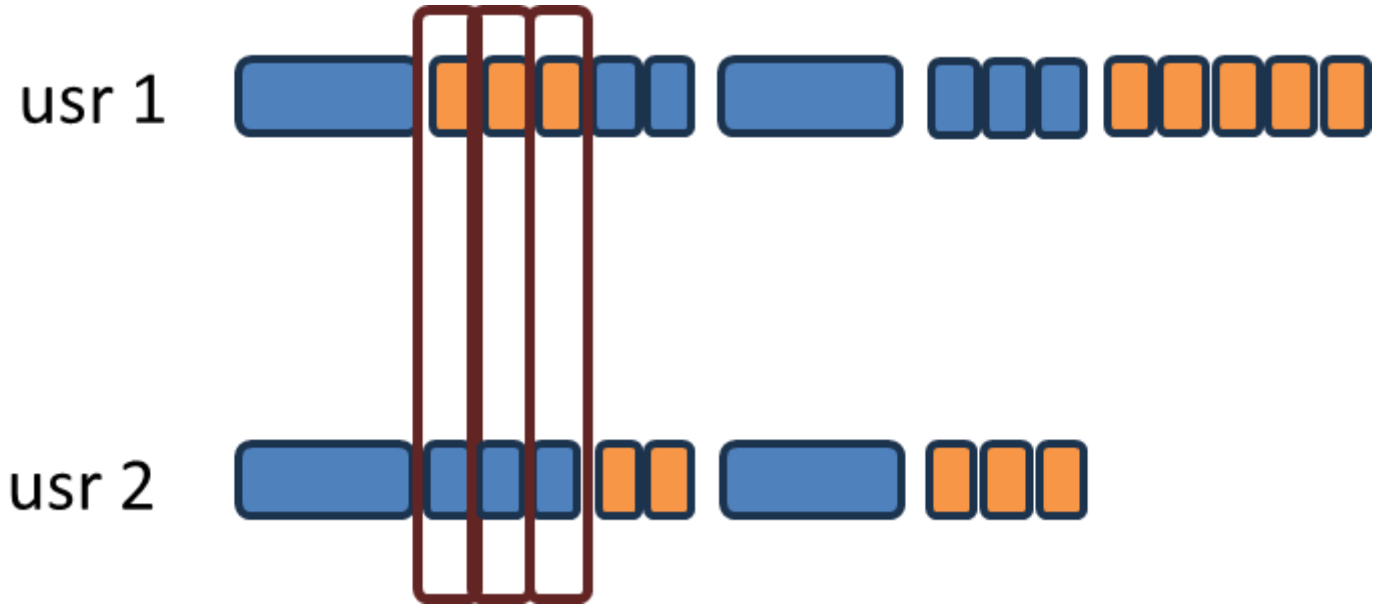
Efficiency gain only w specialized hw

Insight
Gain prop $FLOP^p*MBW^d : FLOP^p*MBW^v$

Chunked Prefill

Split large prefills into smaller chunks

just enough to consume the leftover compute budget in decode batch



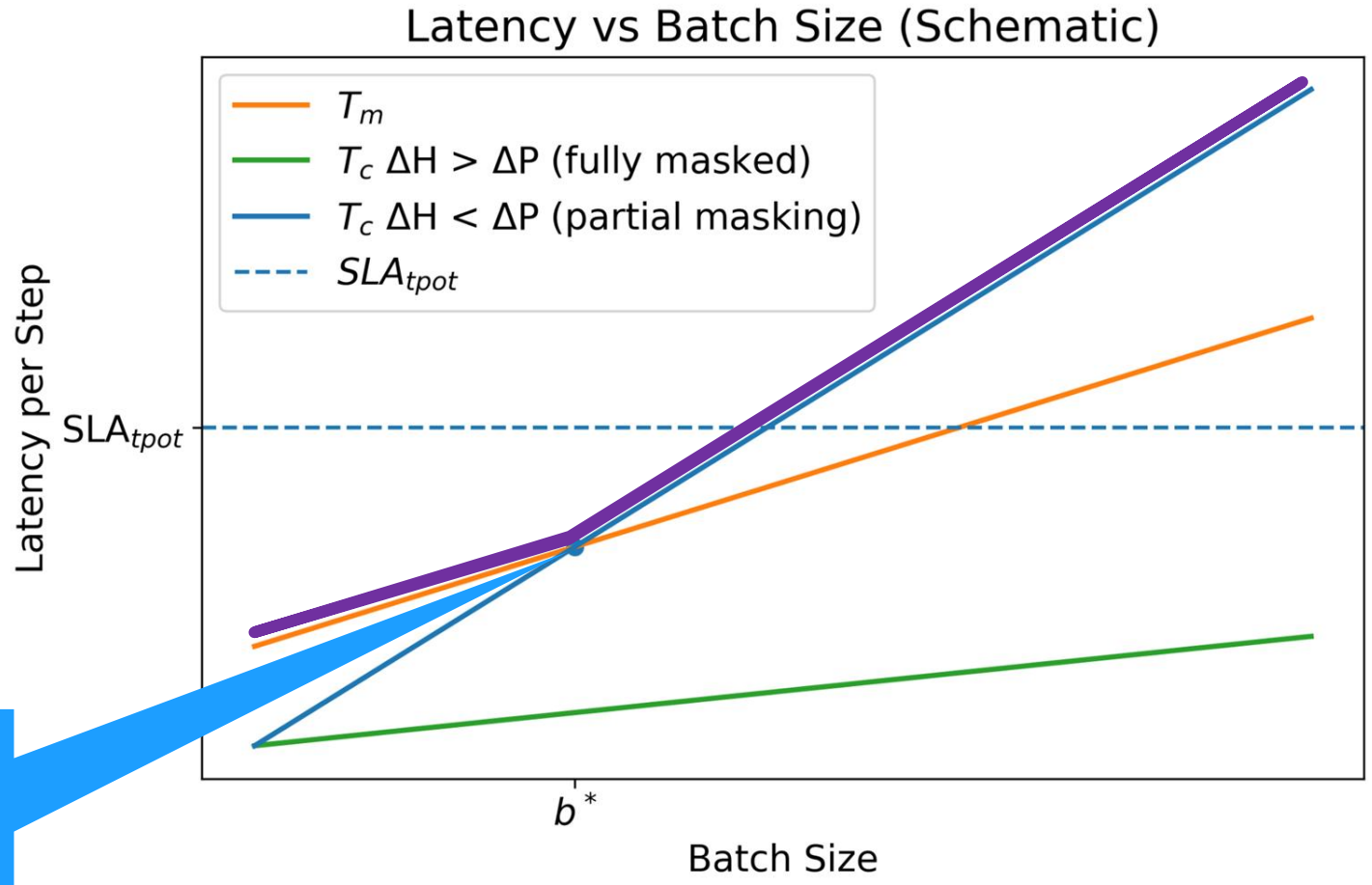
Projected Normalized Throughput Gain

Optimal chunking

- $N_c = b * N_{in} / N_{out}$

Efficiency gain

- $1+\Delta P/\Delta H$ when fully masked
- $1+\Delta H/\Delta P$ partial masking

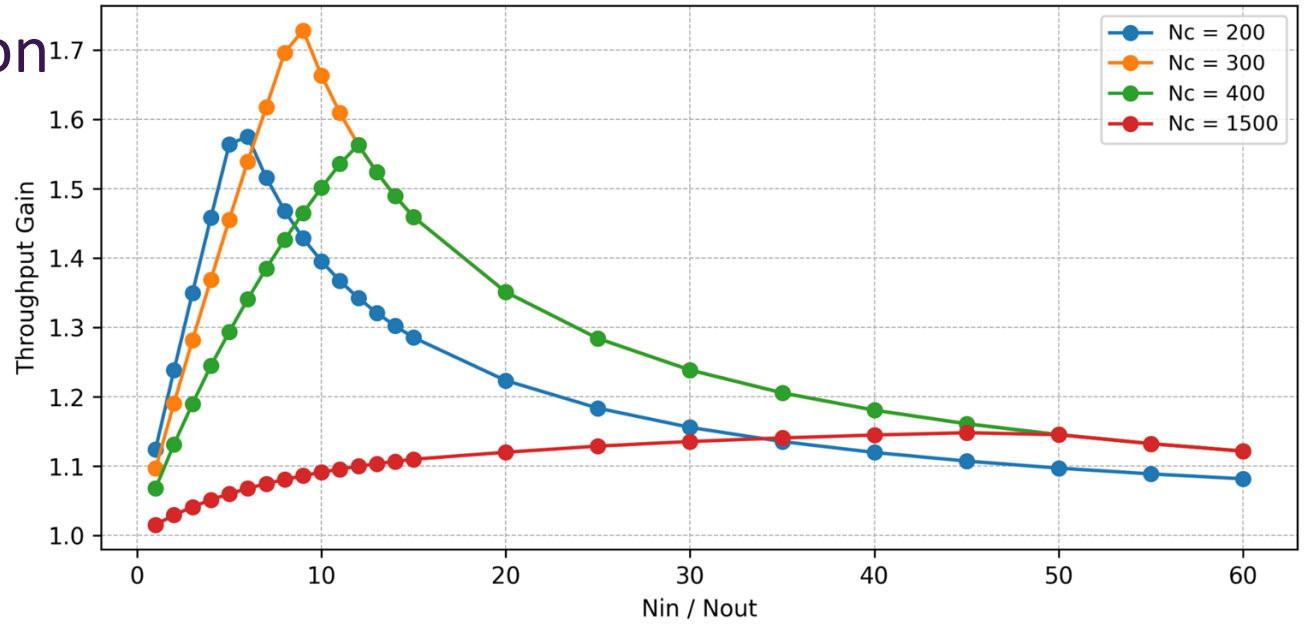


Insight
At most 2x efficiency gain

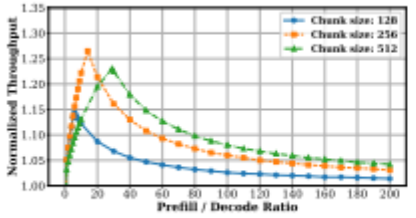
Projected Efficiency Gain

Non-optimal chunking gain projection

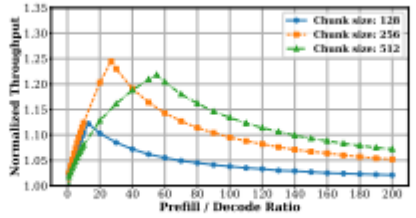
- Llama3-70B (H100, tp2)
- Batch size = 32
- Sequence length 3K
- Vary input-to-output ratio
- Different chunk sizes



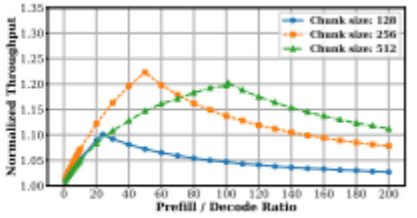
- vs original Sarathi paper
  - Same trends but
  - Lower gain ~25% - why?
    - TTFT degraded performance ??



(a) Sequence length = 1K, batch size = 18.

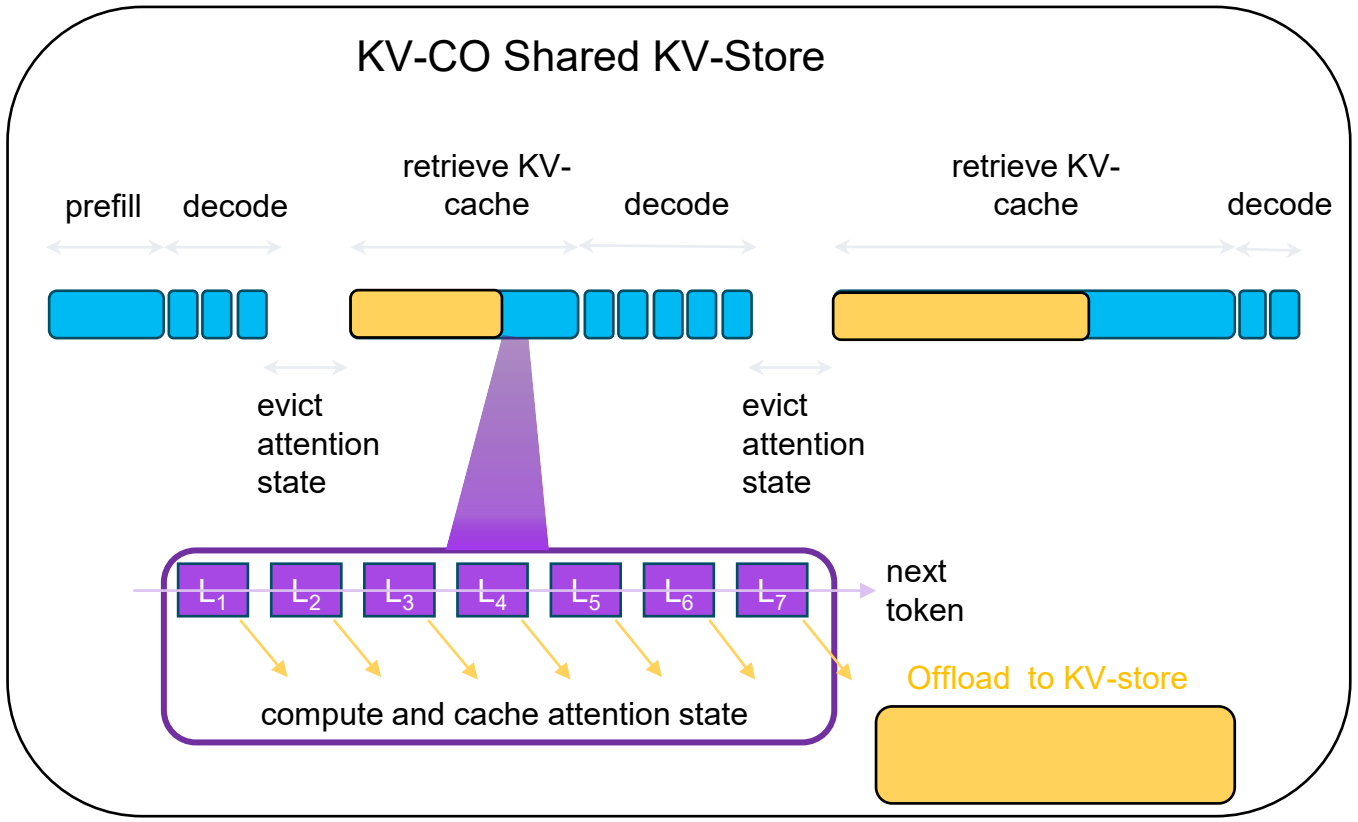


(b) Sequence length = 2K, batch size = 10.



(c) Sequence length = 3K, batch size = 6.

KV-Cache Offloading



e: token acceleration from offloading

TTFT speedup

- Compute bound: $1/(1-\alpha)$
- IO bound: e/α

Efficiency gain

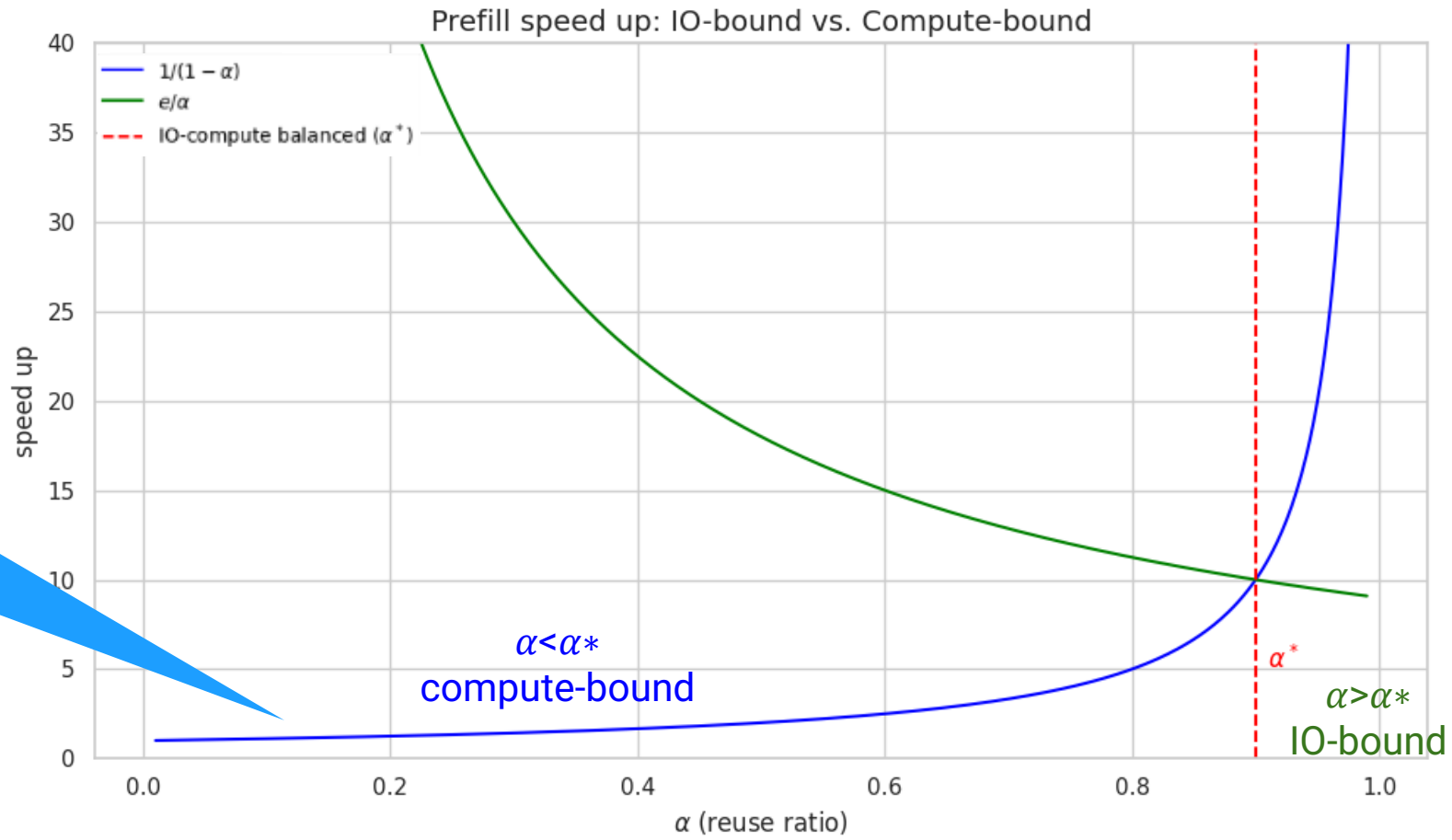
- $1+\Delta P/\Delta H$
- Also when combined with PDD or chunked prefill

Prefill Acceleration Analysis

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

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

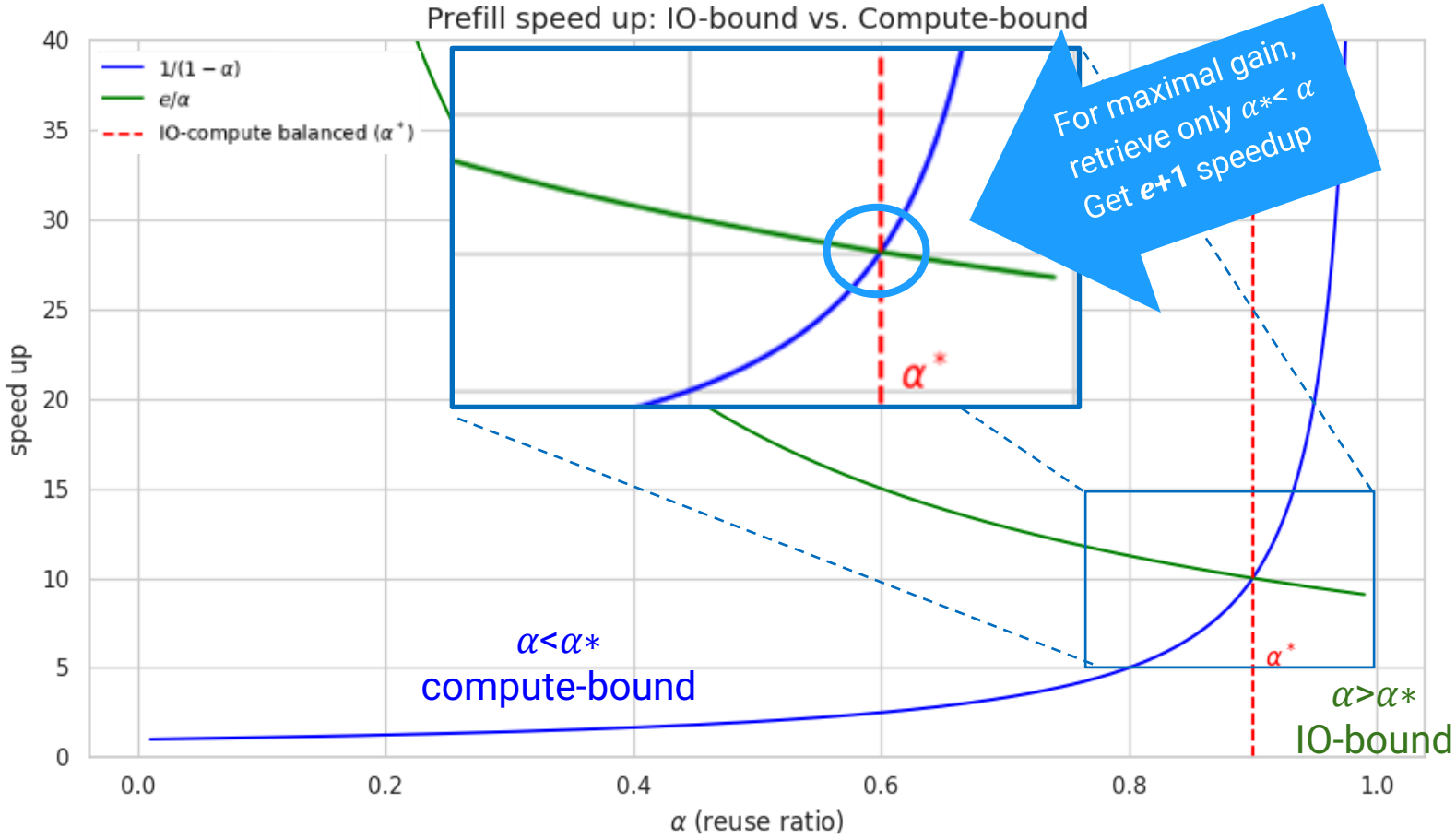
Insight
No advantage for higher IO speed! (HBM, DRAM)



Prefill Acceleration Analysis (Cont.)

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

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

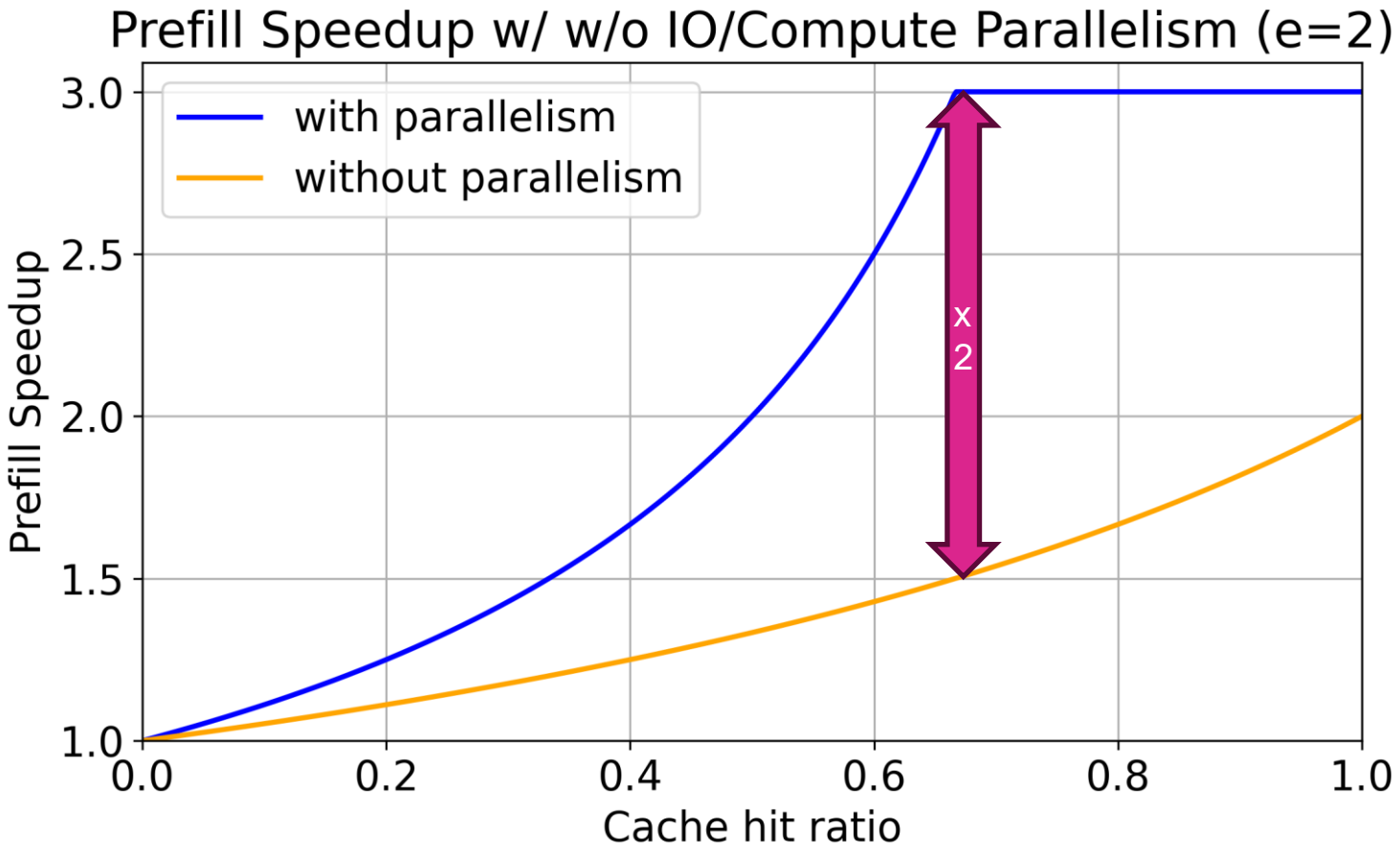


SDC | StorageAI™

Prefill Acceleration Analysis (Cont.)

Load entire KV-cache vs layer-by-layer

- Can result in 2x diff in prefill acceleration
- Depends on e



Summary and Conclusions

- Predictive framework for understanding end-to-end LLM inference performance
  - Shifts system design from empirical tuning to principled, analytically driven optimization
- Closing the prefill interference gap results in $1+\Delta P/\Delta H$ gain
- Model limited to deployments of an inference unit with negligible comm overhead
- More details, and empirical validation in the paper
  - submitted, under review


SDC StorageAI™
A SNIA Event
Thank You