SC26 · Chicago · November 2026

TIDE: Temporal Incremental Draft Engine for Self-Improving LLM Inference

Jiyoung Park, Hankyu Jang, Changseok Song, and Wookeun Jung

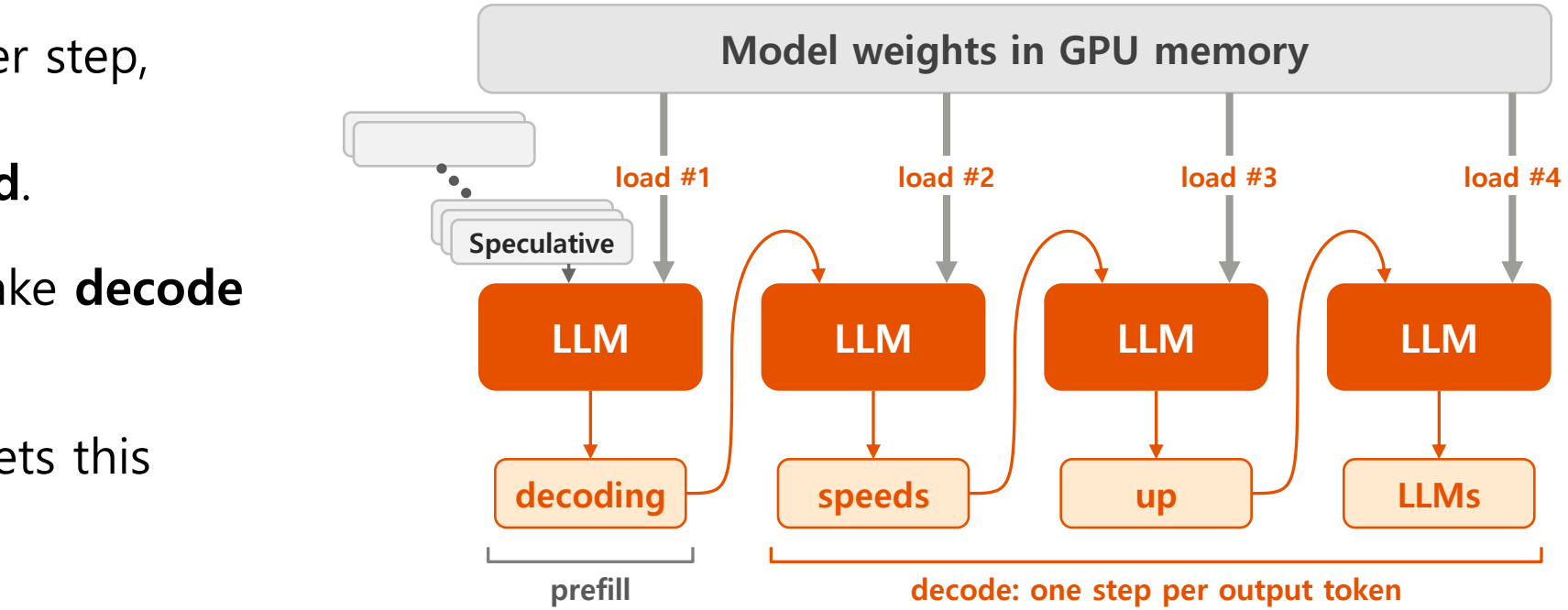
Moreh, Inc.

{jiyoung.park, hankyu.jang, changseok.song, wookeun.jung}@moreh.io

MOREH

Background: LLM Inference

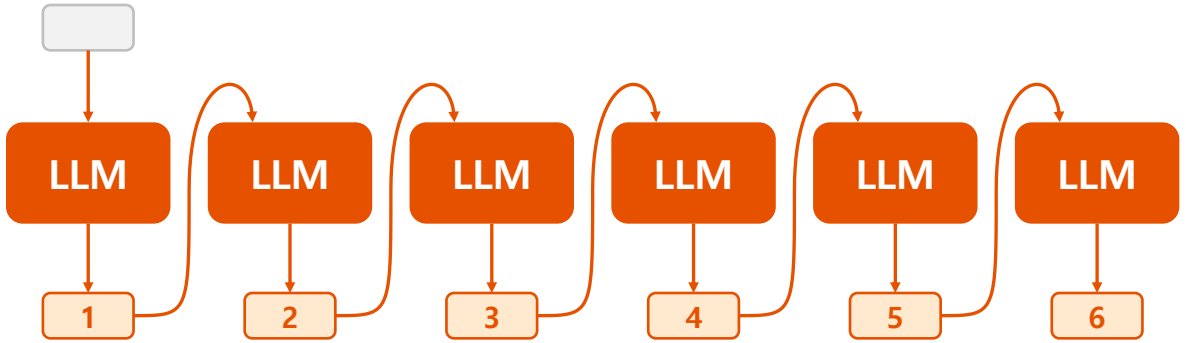
- Two phases: **prefill** (whole prompt at once), then **decode** (output).
- Decode emits one token per step, reloading all weights: **memory-bandwidth bound**.
- Long reasoning outputs make **decode the main bottleneck**.
- **Speculative decoding** targets this bottleneck.



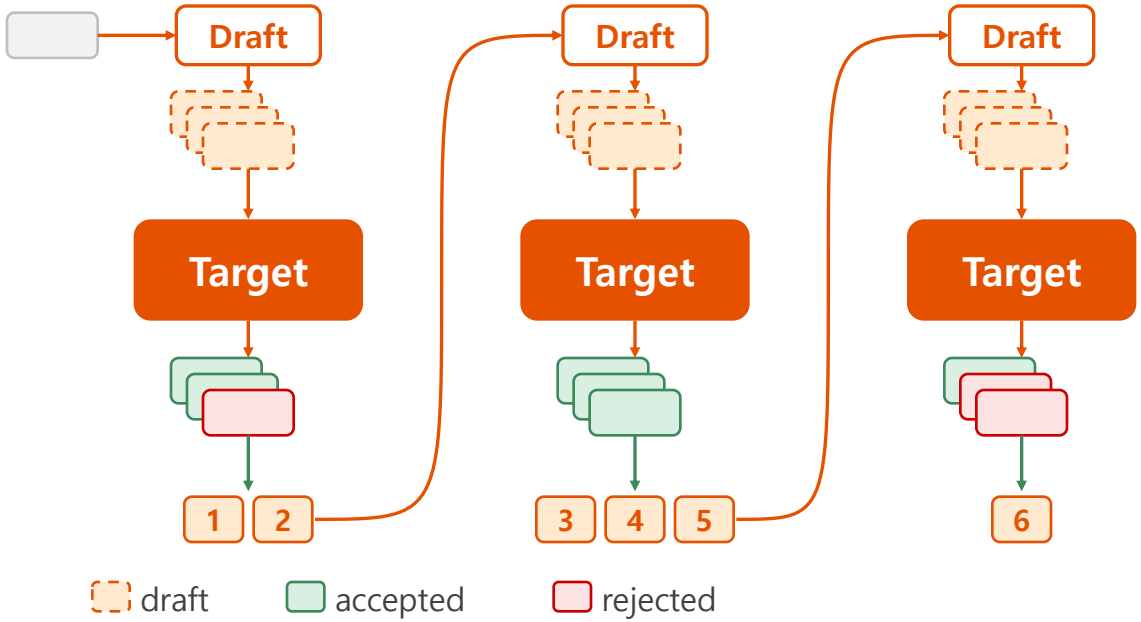
Background: Speculative Decoding

- A **tiny draft model** proposes several candidate tokens.
- In a **single decoding step**, the LLM (target model) can check whether the candidate tokens match what **standard decoding would generate**; this is called **verification**.

Standard decoding: 6 decoding steps → 6 tokens

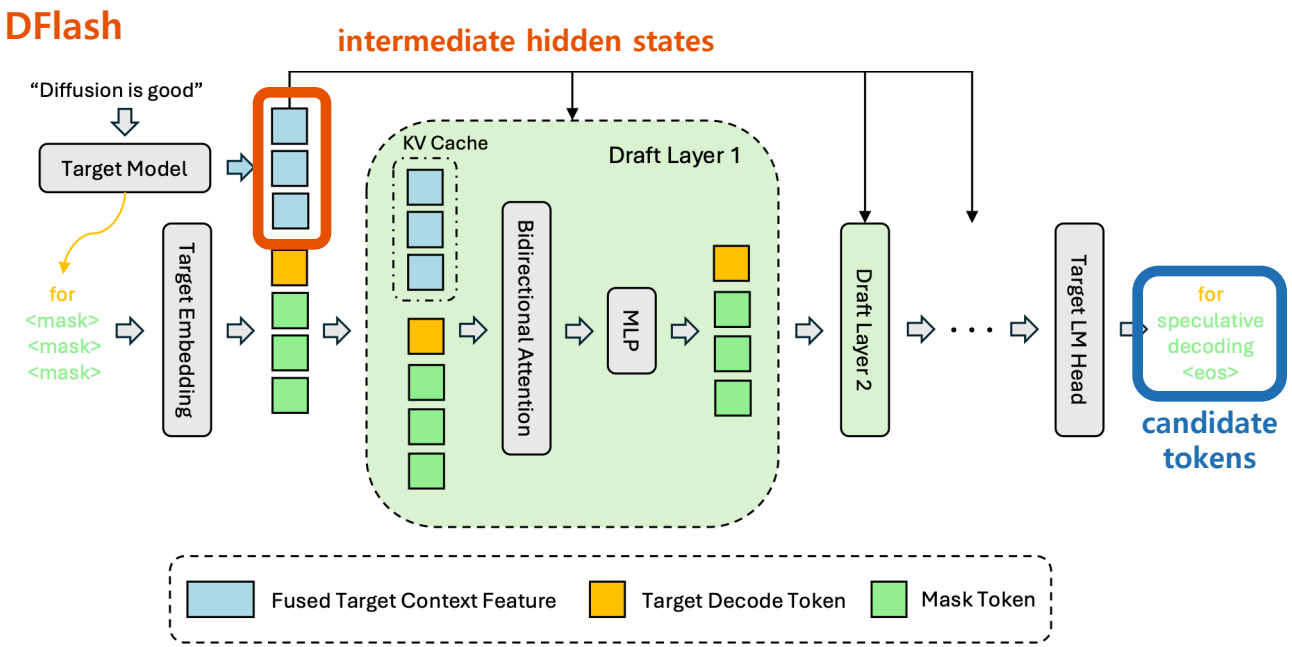


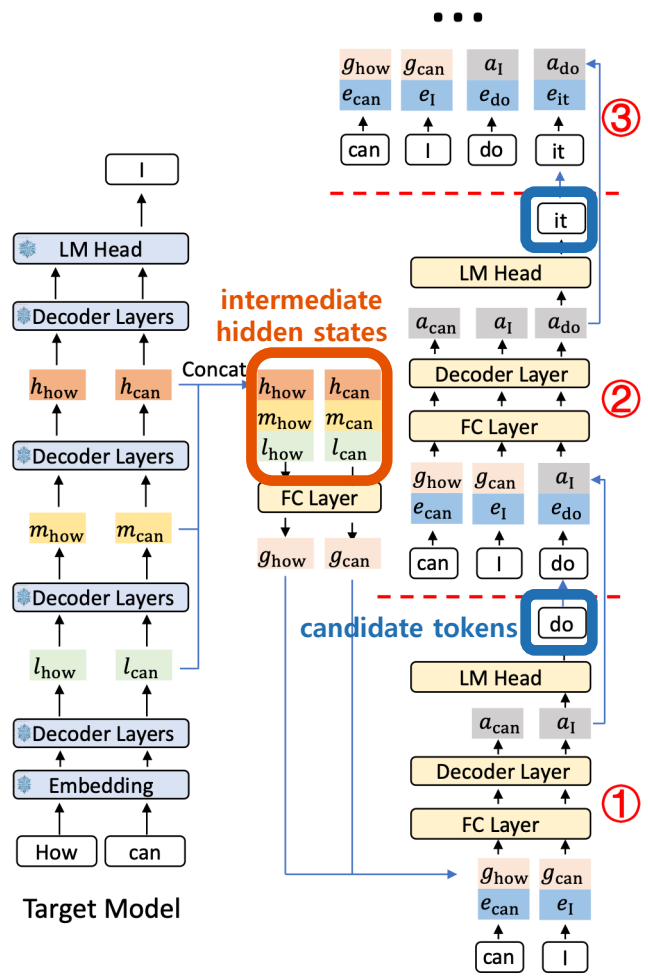
Speculative decoding: 3 decoding steps → same 6 tokens



Background: Hidden-State-Based Draft Models

- Leading draft models take the target model's **intermediate hidden states** of the context (prompt + tokens generated so far) as input.
- They are trained to predict the target model's **next few tokens** from those hidden states.
- DFlash and EAGLE-3 use different architectures, but **both build on the target's hidden states**.

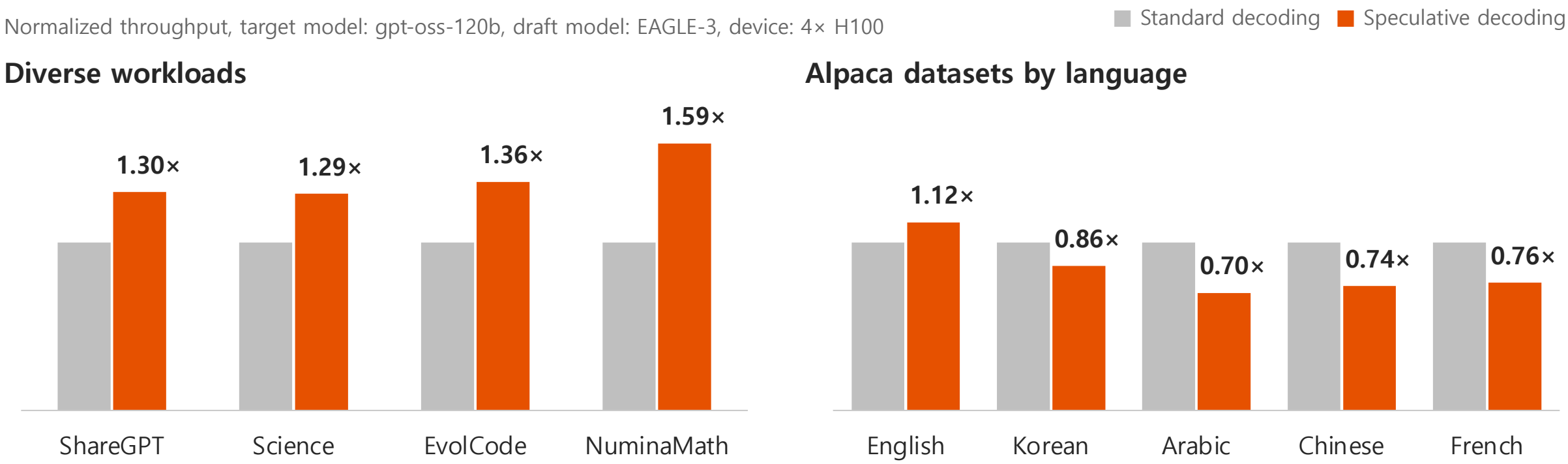




Figures: Chen et al., DFlash, arxiv.org/abs/2602.06036 (2026); Li et al., EAGLE-3, arxiv.org/abs/2503.01840 (2025).

Motivation: Throughput Gain Is Workload-Dependent

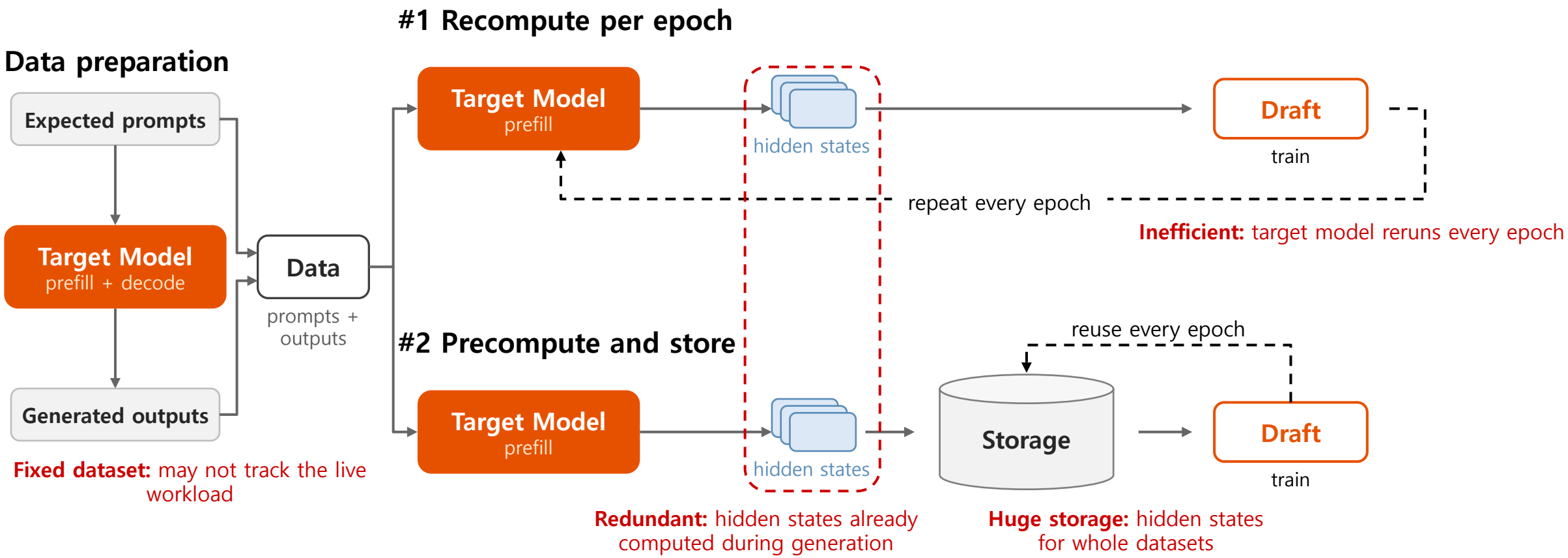
Normalized throughput, target model: gpt-oss-120b, draft model: EAGLE-3, device: 4× H100



- The speedup of pre-trained draft model **varies widely across workloads** (1.29–1.59×).
- On workloads it was not trained for, it runs **below standard decoding** (0.70–0.86×).
- Production workloads keep shifting, so the draft must **keep training online**.
- Speculative decoding should run **only when it helps**, switched on and off adaptively.

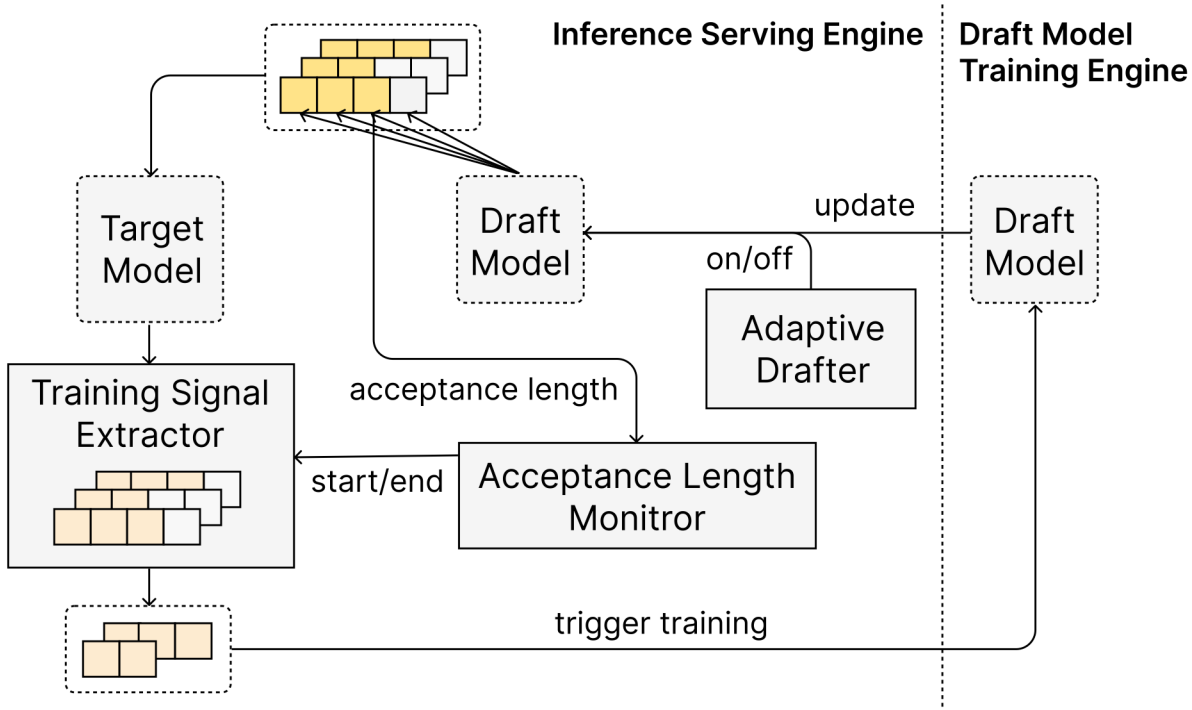
Motivation: Existing Draft Training Is Static and Costly

Existing draft training is **offline**: done before deployment on a **fixed dataset**, in one of two ways



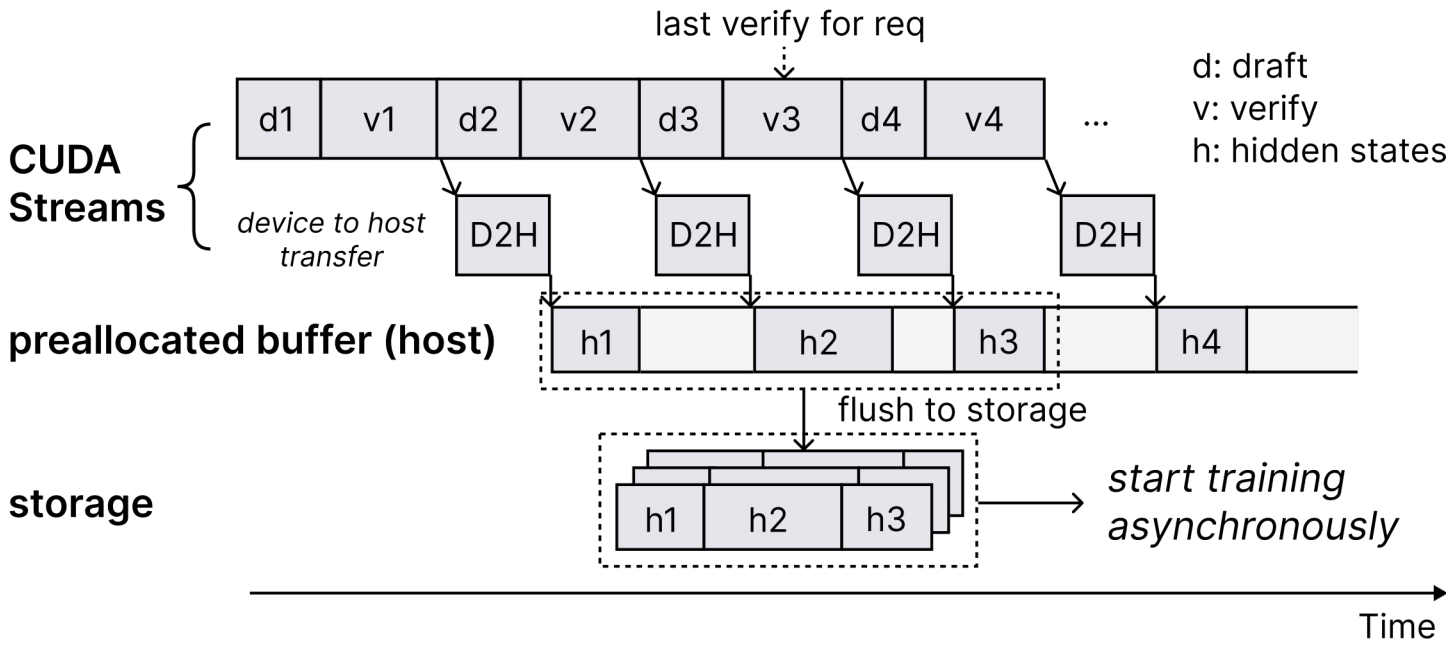
Design: TIDE (Temporal Incremental Draft Engine)

- **Serving-time hidden-state extraction**: reuse hidden states the target computes while serving
  - No redundant prefill; learns from the live workload
- **Chunked draft training**: train once enough hidden states accumulate, then discard them
  - Storage stays bounded
- **Adaptive speculative decoding**: turn off when it hurts, on when the draft improves
  - Mitigates slowdowns during draft adaptation.



Design: Serving-time Hidden States Extraction

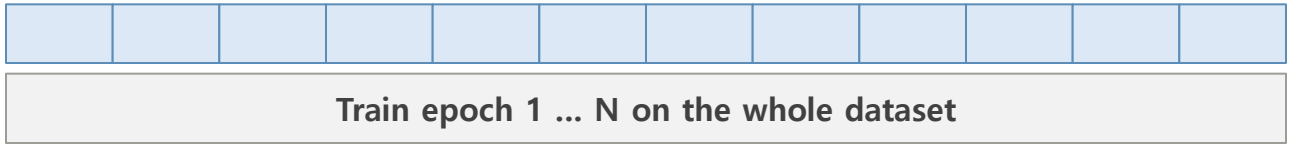
- After each verify step, hidden states of **accepted tokens** are copied to a preallocated host buffer (D2H).
- Copies run on a separate CUDA stream, **overlapping the next draft/verify step**.
- A full buffer is flushed to shared storage; training starts **asynchronously.**
- No extra prefill, no serving slowdown.



Design: Chunked Draft Training

- Once hidden states in storage reach a **threshold**, the training engine starts a round.
- Each round fine-tunes the current draft on that chunk, then **discards it**.
- Chunk-wise training achieves **final draft accuracy comparable** to that of multi-epoch training on the full dataset.
- Storage holds **at most two chunks**, not the whole dataset.

Whole dataset, one round



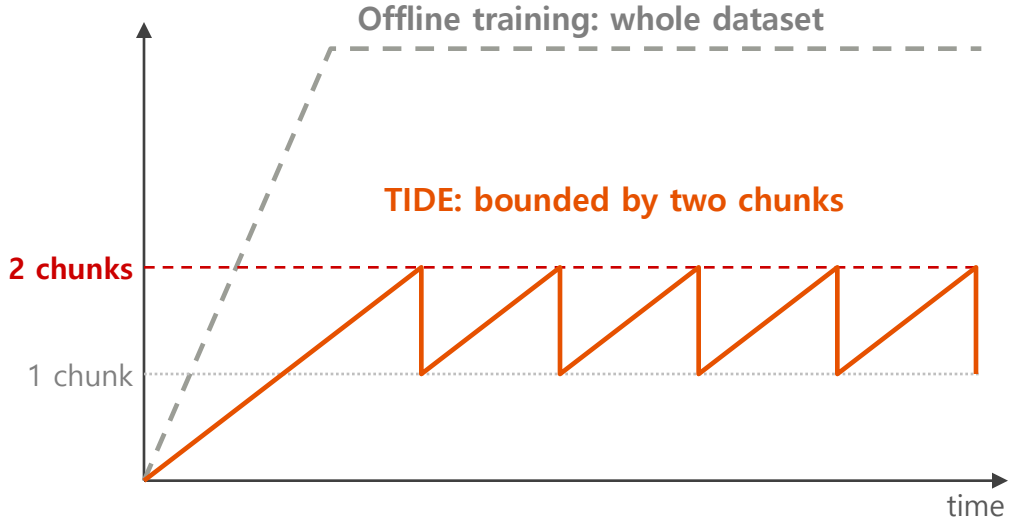
Must store all hidden states; training starts only after collection ends

Chunked: one round per chunk



Stores one or two chunks; training starts as soon as a chunk fills

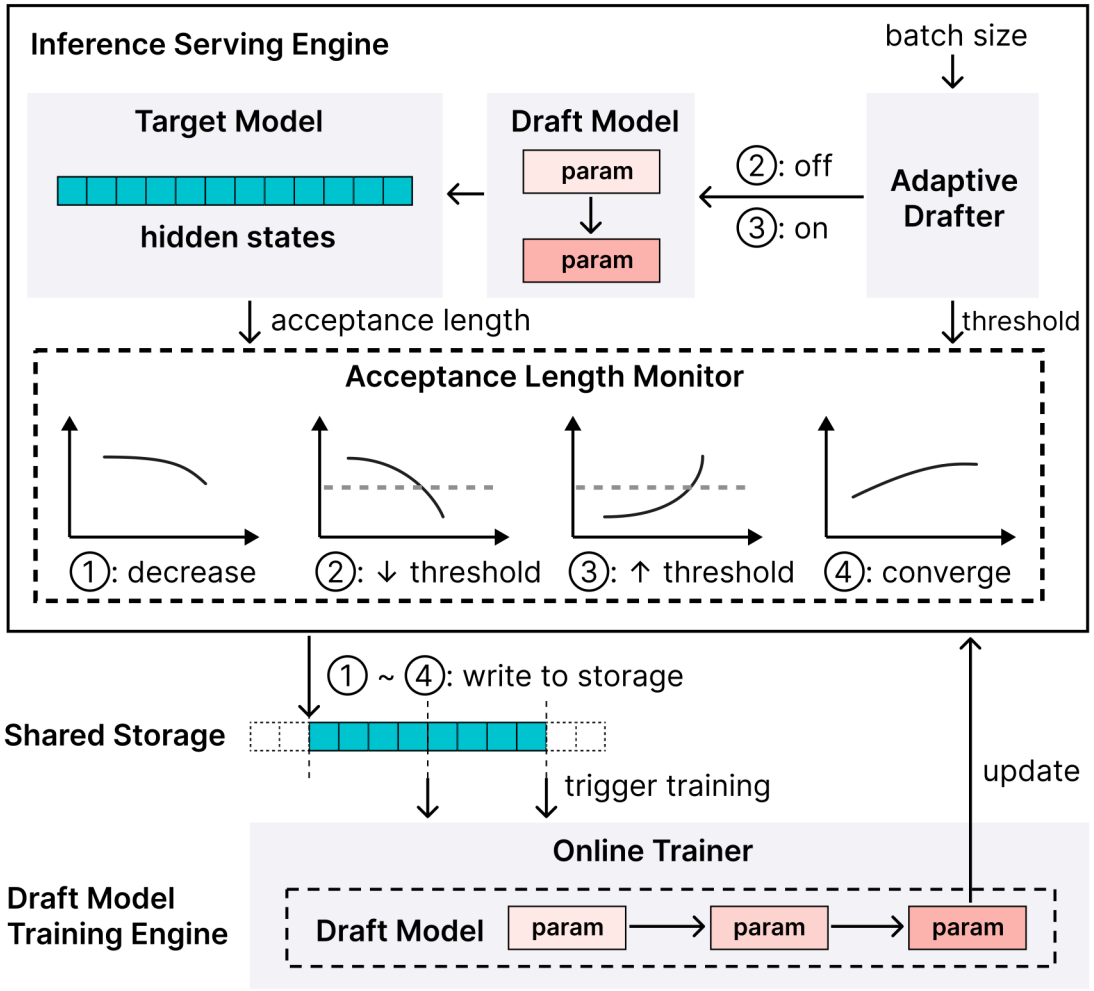
Storage usage over time



next chunk fills while the previous one trains; deleted after training

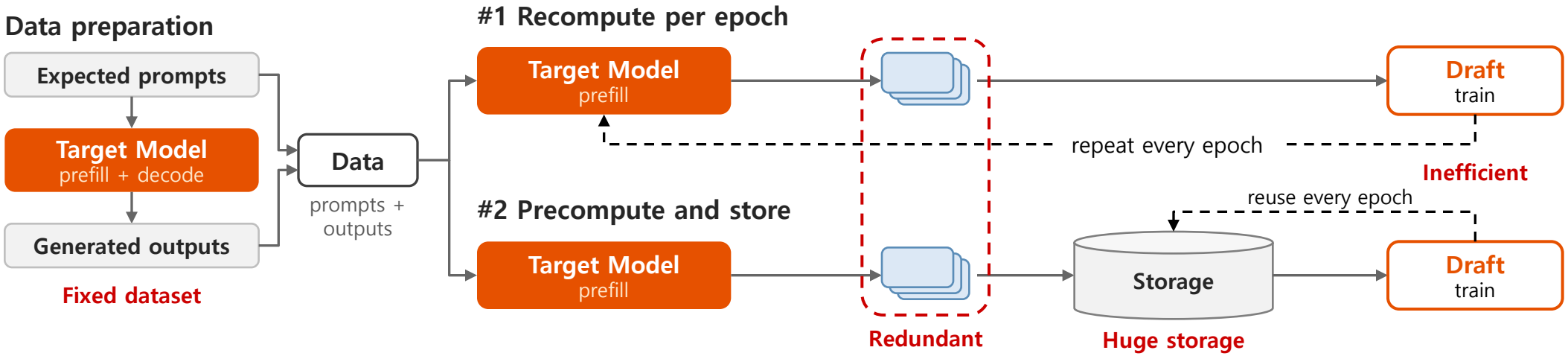
Design: Adaptive Speculative Decoding

- The monitor tracks acceptance length; a drop signals a workload shift and **starts hidden-state extraction** (①).
- Below the break-even threshold for the current batch size, speculative decoding is **turned off** (②).
- Training lifts acceptance length; above the threshold, speculation is **turned back on** (③).
- Once acceptance converges, **extraction stops** until the next shift (④).
- **Mitigates slowdowns** during draft adaptation.



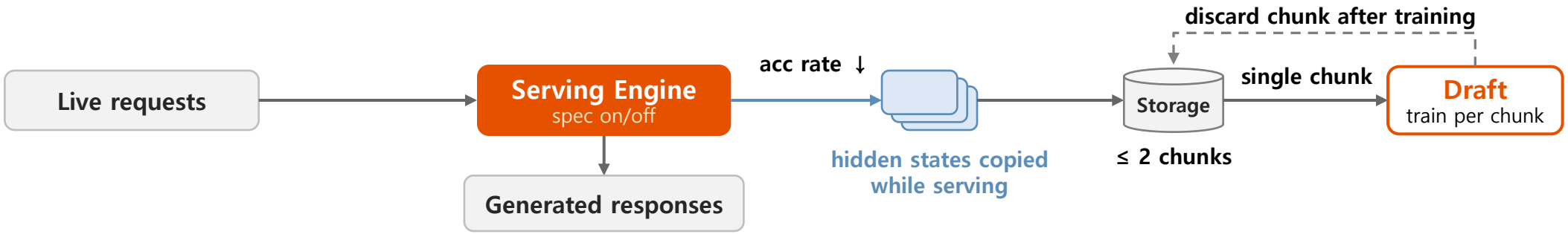
Design: Overall

Existing: offline training



Online training with TIDE

TIDE: online training

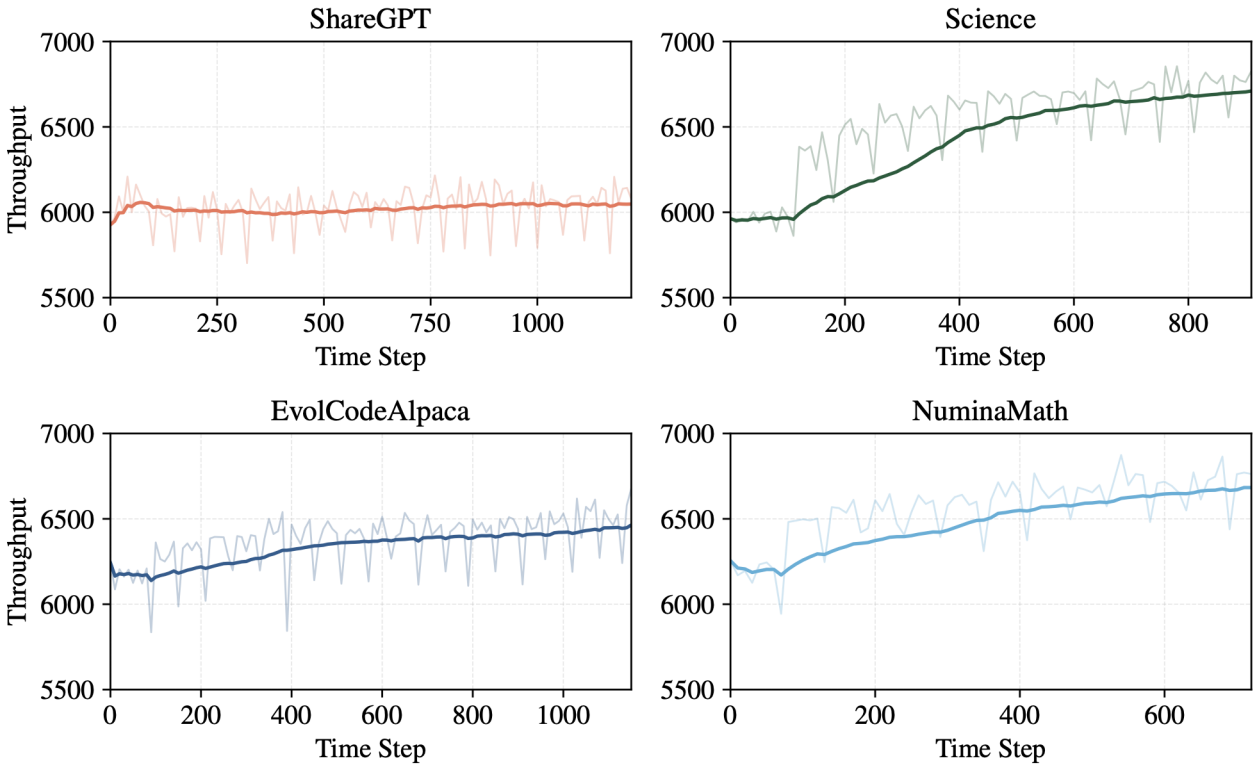


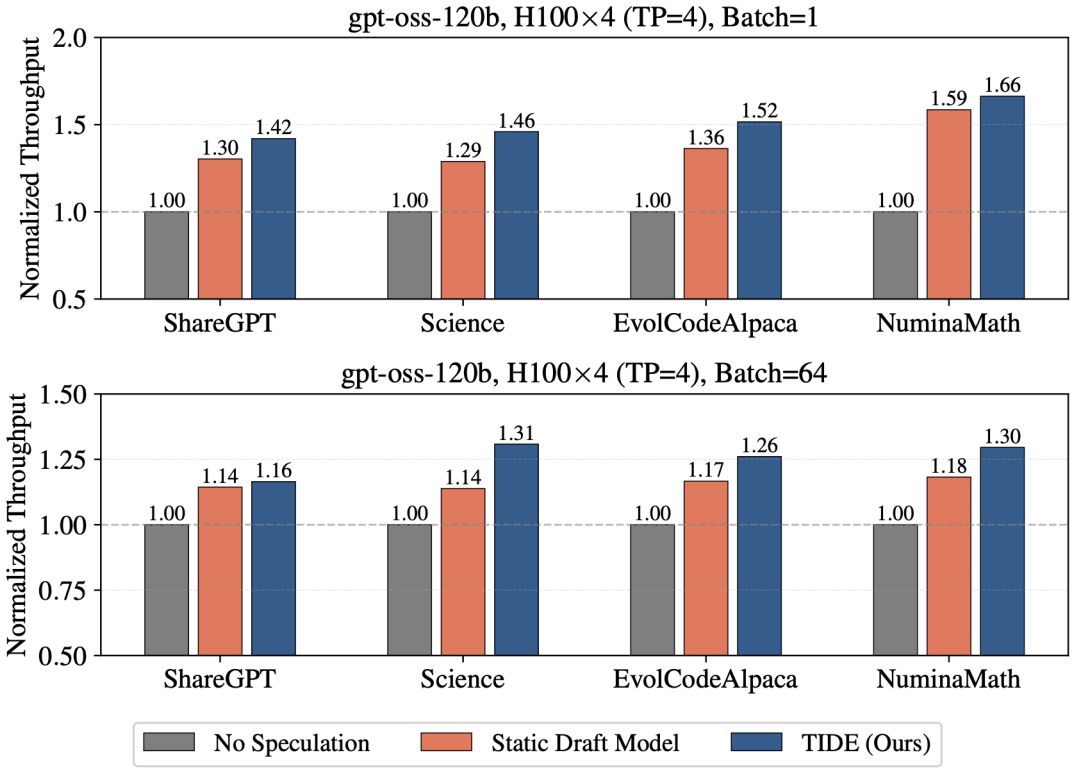
Evaluation: Experimental Setup

Serving	SGLang on 1 node, up to 8× NVIDIA H100
Draft training	SpecForge on 1 node, 4× AMD MI250 (PyTorch FSDP); SpecForge = draft-training framework supporting Recompute and Precompute
Shared storage	VAST, NFSv3 over RDMA, 200 Gb/s InfiniBand
Target models	gpt-oss-120b
Draft model	EAGLE-3, 3 draft tokens per step (empirically best)
Datasets	ShareGPT, Science, EvolCodeAlpaca, NuminaMath; Alpaca in Korean, Arabic, Chinese, French

Evaluation: Throughput Improves as the Draft Adapts Online

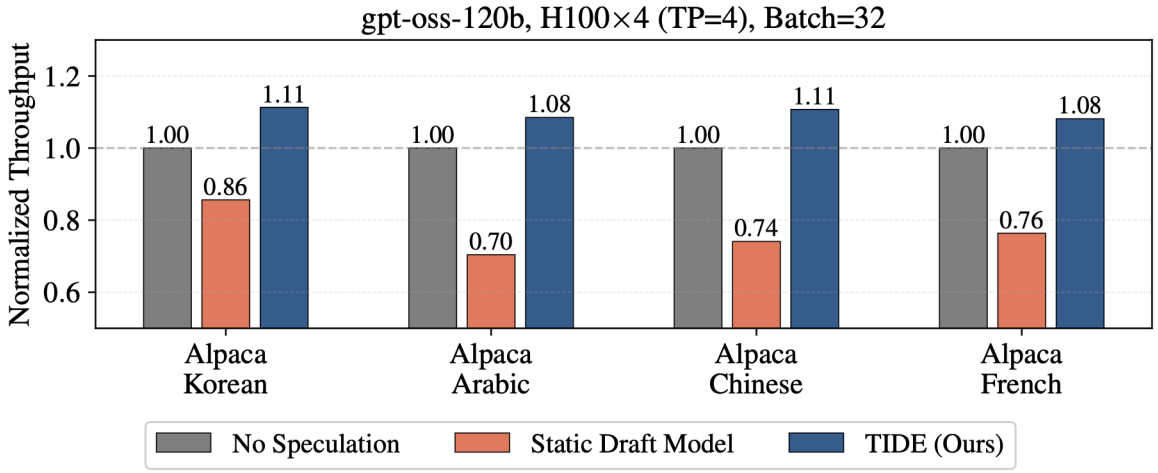
- As training proceeds, throughput **gradually increases** and ends above the static draft.

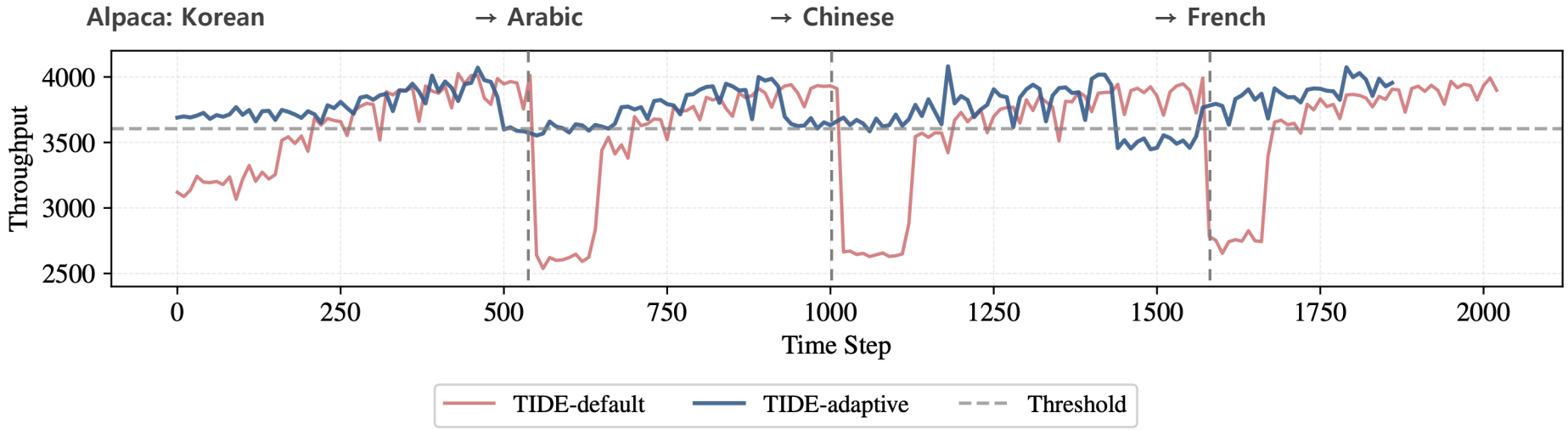




Evaluation: Recovery from Workload Shifts

- Workload shifts across Alpaca datasets: **Korean → Arabic → Chinese → French**.
- TIDE-default: throughput drops right after each shift, then **recovers via online training**.
- TIDE-adaptive: speculation is **turned off** during the drop, avoiding the slowdown.





Evaluation: Training Efficiency

- TIDE trains **1.67× faster** than Precompute with **24× less storage**.
- Chunk-wise training achieves **final draft accuracy comparable** to that of Precompute.

Draft training time

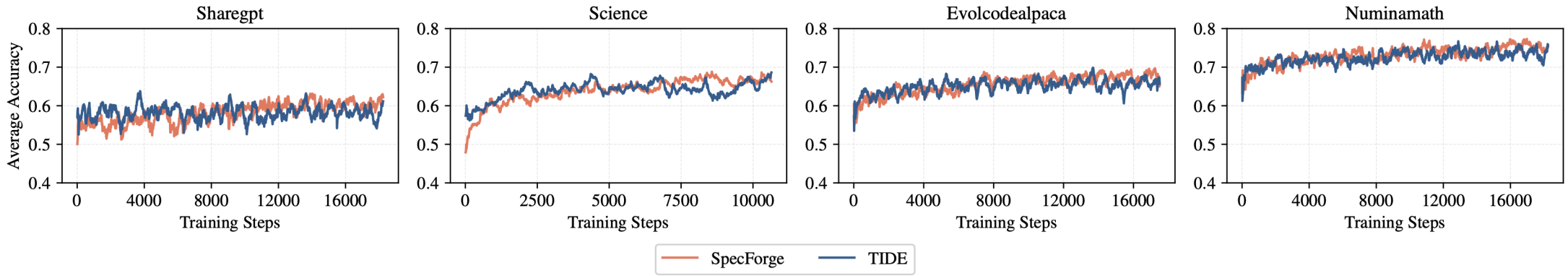
Method	Prefill	Train	Total	Speedup
Precompute	6.16 h	9.16 h	15.32 h	1.00×
Recompute	18.48 h	9.16 h	27.64 h	0.55×
TIDE	–	9.16 h	9.16 h	1.67×

Hidden states storage

Method	Storage
Precompute	4.66 TB
Recompute	0 (none)
TIDE	0.19 TB

Both tables: training on ShareGPT (100k samples); TIDE chunk size = 4k samples

Draft accuracy during training · TIDE vs. SpecForge (Precompute)

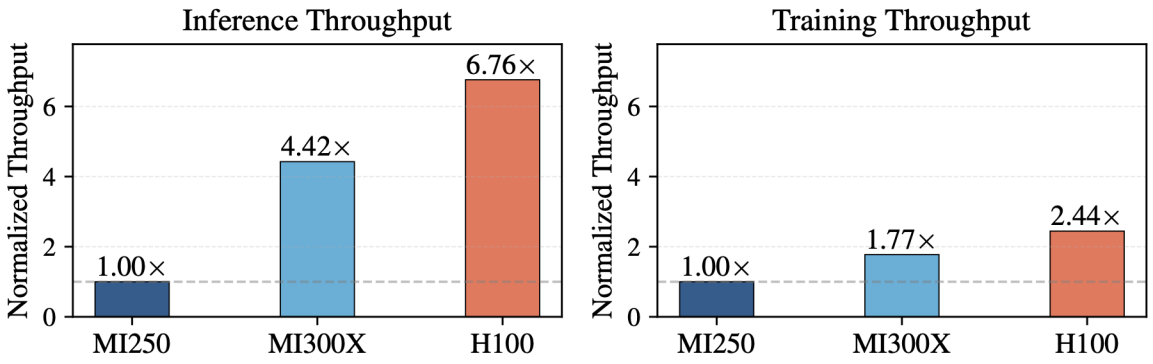


Evaluation: Heterogeneous GPU Allocation

- Old-gen GPUs lag far behind on inference but much less on training, so TIDE **serves on H100 and trains the draft on MI250**.
- TIDE beats running plain inference on both H100 and MI250 by **1.08–1.22×** after draft training converges.

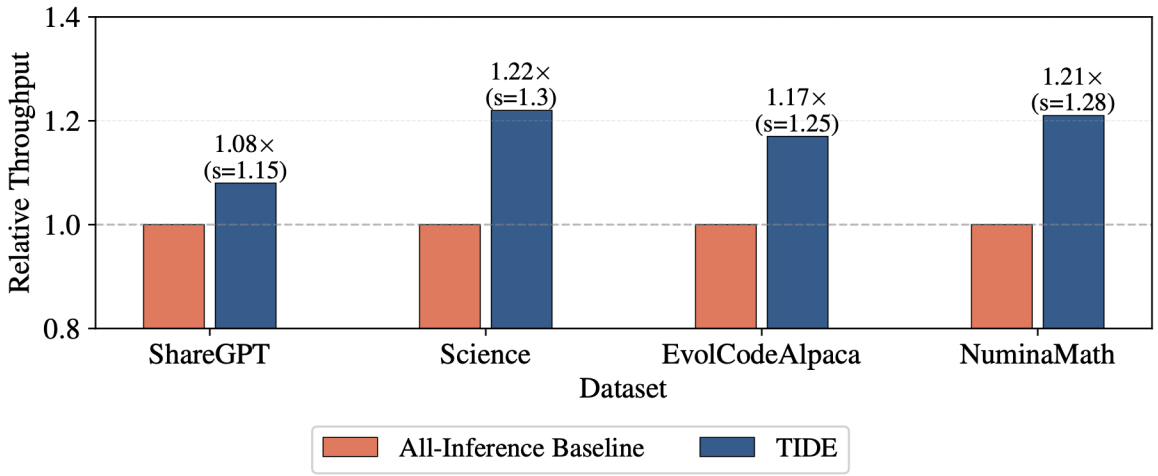
Per-GPU throughput, normalized to MI250

New-gen GPUs gain far more on inference than on training



TIDE vs. all-inference baseline

Same budget: 8× H100 + 4× MI250



Summary

- A static draft's speedup depends on the workload and can fall **below standard decoding**.
- TIDE trains the draft **online on the live workload**, in a separate training engine fed by hidden states copied from serving.
- Chunked training keeps storage small; adaptive speculation avoids slowdowns during shifts.
- Training is **1.67× faster** with **24× less storage** than Precompute.
- Training on old-gen GPUs adds **1.08–1.22×** system throughput.