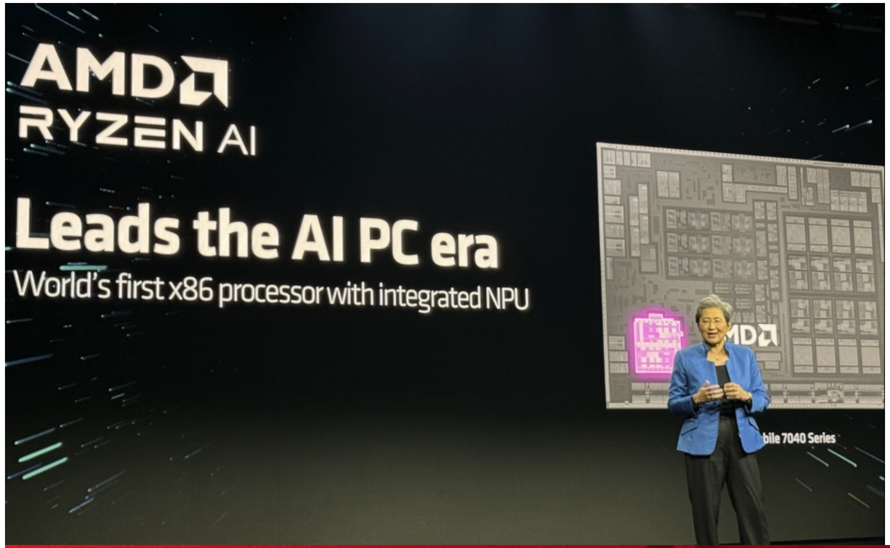
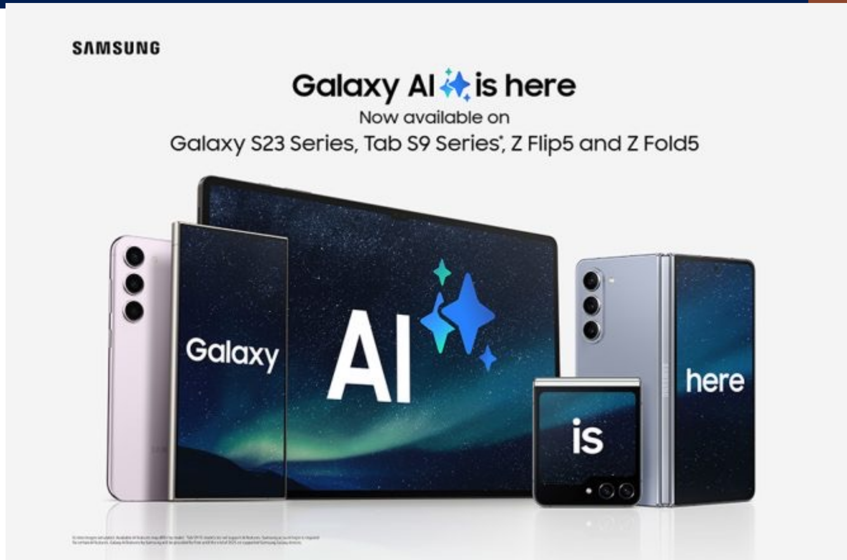
Deferred Continuous Batching in Resource-Efficient Large Language Model Serving

Yongjun He (yongjun.he@inf.ethz.ch)
Yao Lu
Gustavo Alonso

ETH Zürich, National University of Singapore

The Rise of AI PCs



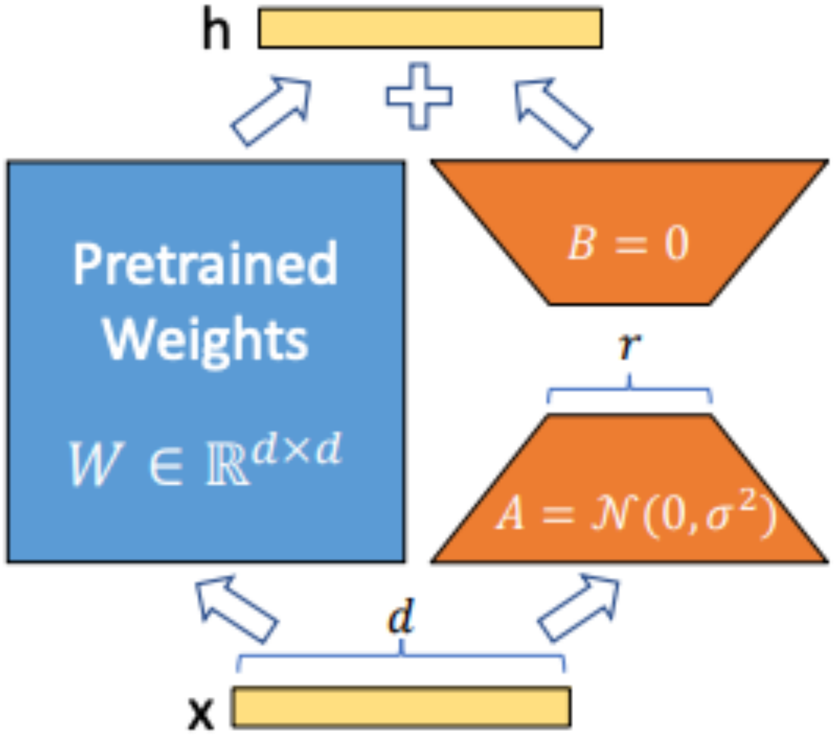


Resource-Efficient Fine-Tuning

Previous methods: Retrain all model parameters.

Low-Rank Adaptation (LoRA)

- Freeze the pre-trained model and update a small number of (additional) parameters
$$h = (W + \Delta W)x = Wx + BAx$$
where $W \in \mathbb{R}^{d\times k}$, $B \in \mathbb{R}^{d\times r}$, $A \in \mathbb{R}^{r\times k}$, and the rank $r \ll (d, k)$.
- Reduce memory usage during fine-tuning by 60 - 70%.
- Reduce memory usage during inference by 100 - 10000x for each new fine-tuned model.

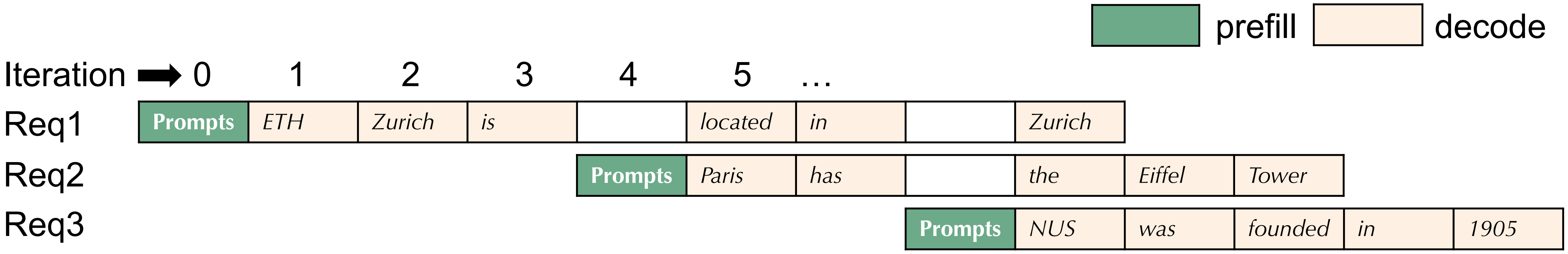


Resource-Efficient Inference

Previous methods: Newly arrived requests have to wait for the current batch to complete.

Continuous Batching

- Newly arrived requests only need to wait for the current token to complete.
- Enable 10 – 20x throughout since the decode stage is memory-bound



Heterogeneous Fine-Tuning and Inference

Example

A student wants to fine-tune an LLM to help improve her thesis for a herbology class. She would choose the biggest LLM that fits in her AI PC for better results.

Challenge

All other local LLM-based applications will not work until the completion of fine-tuning.

Prior Solutions

Spatial GPU sharing

- Execute fine-tuning and inference in parallel on smaller pre-trained LLMs
- Reduce model's capabilities

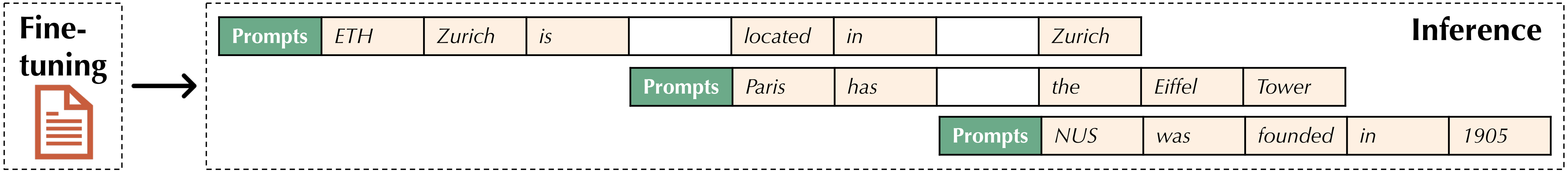
Temporal GPU sharing

- Switch from fine-tuning to inference when new requests arrive
- Leave little time for fine-tuning tasks
- Incur high context switch overhead

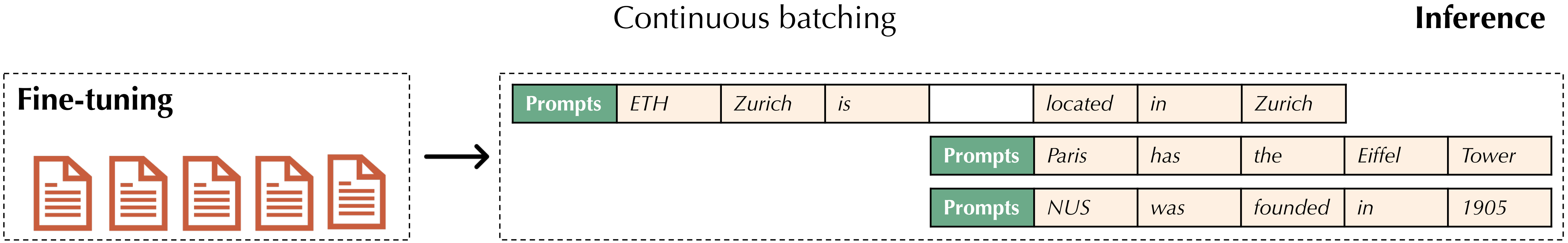
Deferred Continuous Batching

A new task scheduling mechanism

- Schedule at the granularity of a single fine-tuning or inference iteration
- Slightly defer inference requests without violating service level agreements



Continuous batching

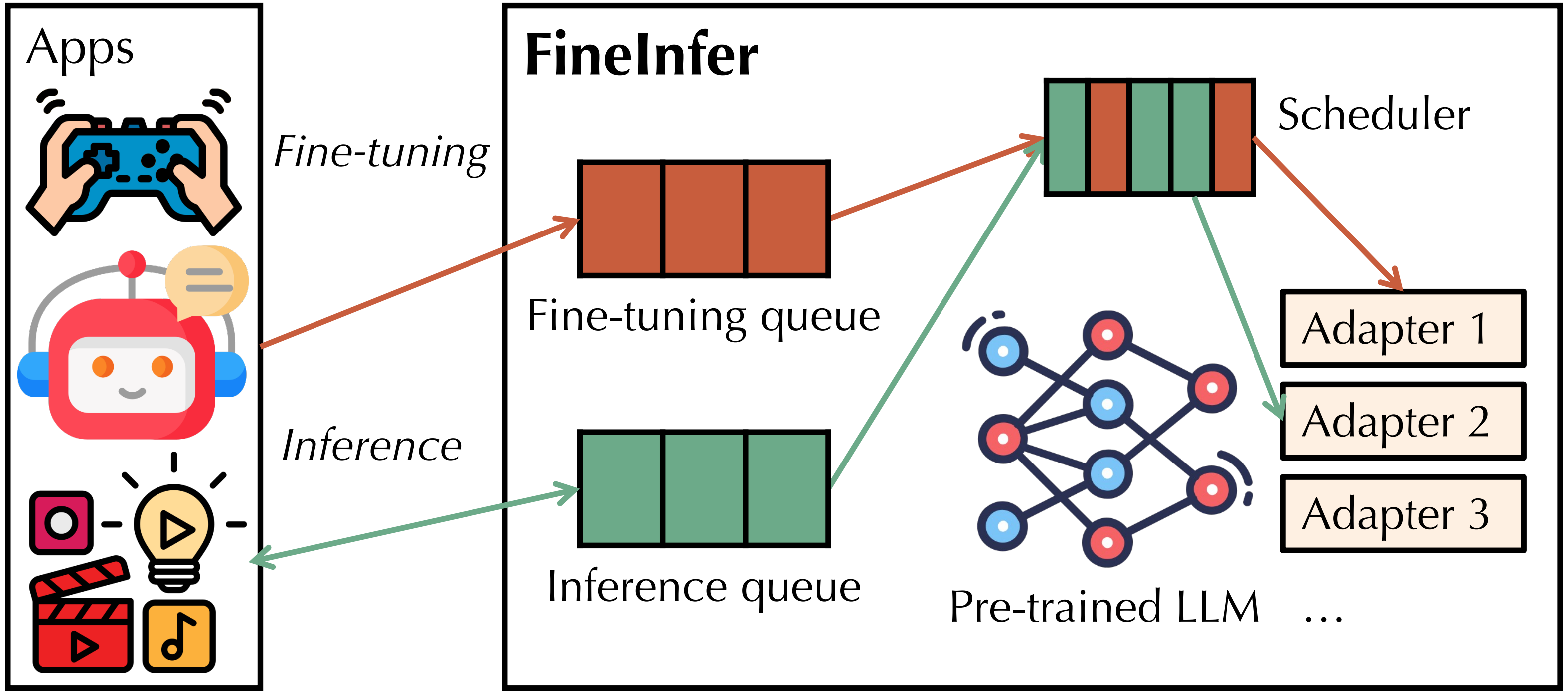


Deferred continuous batching

FineInfer at a Glance

Designed for concurrent parameter-efficient **Fine-tuning** and **Inference**

- Deferred Continuous Batching
- Hybrid system architecture



Hybrid System Architecture

Minimize context switch overhead

- Base model multiplexing
- Iteration-level switching

Stage	DeepSpeed	Colossal-AI	FineInfer
Task initialization	1.153 / 0.015 s	0.28 / 0.045 s	0 / 0 s
Task cleanup	2.330 / 1.252 s	3.456 / 1.376 s	0 / 0 s
Data movement	5.882 s	5.918 s	0 - 0.052 s

The breakdown of switching overhead with Llama2-7B workloads on an Nvidia 4090 GPU.

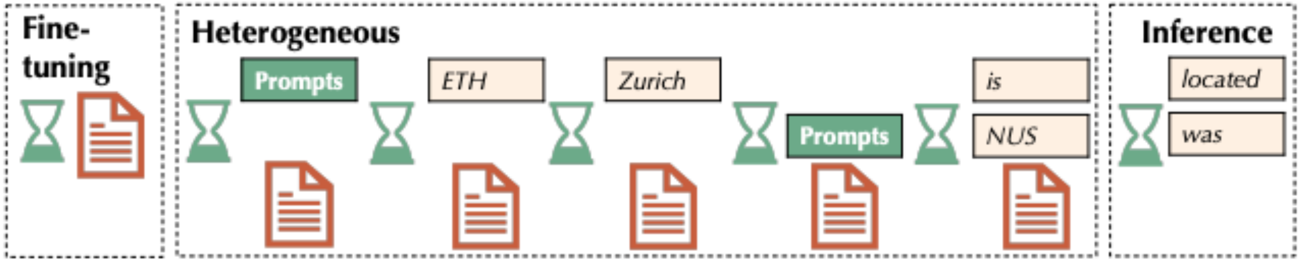
Hybrid System Architecture

Amortize data movement overhead for larger-than-GPU LLMs

- Heterogenous batching

(a) Continuous batching

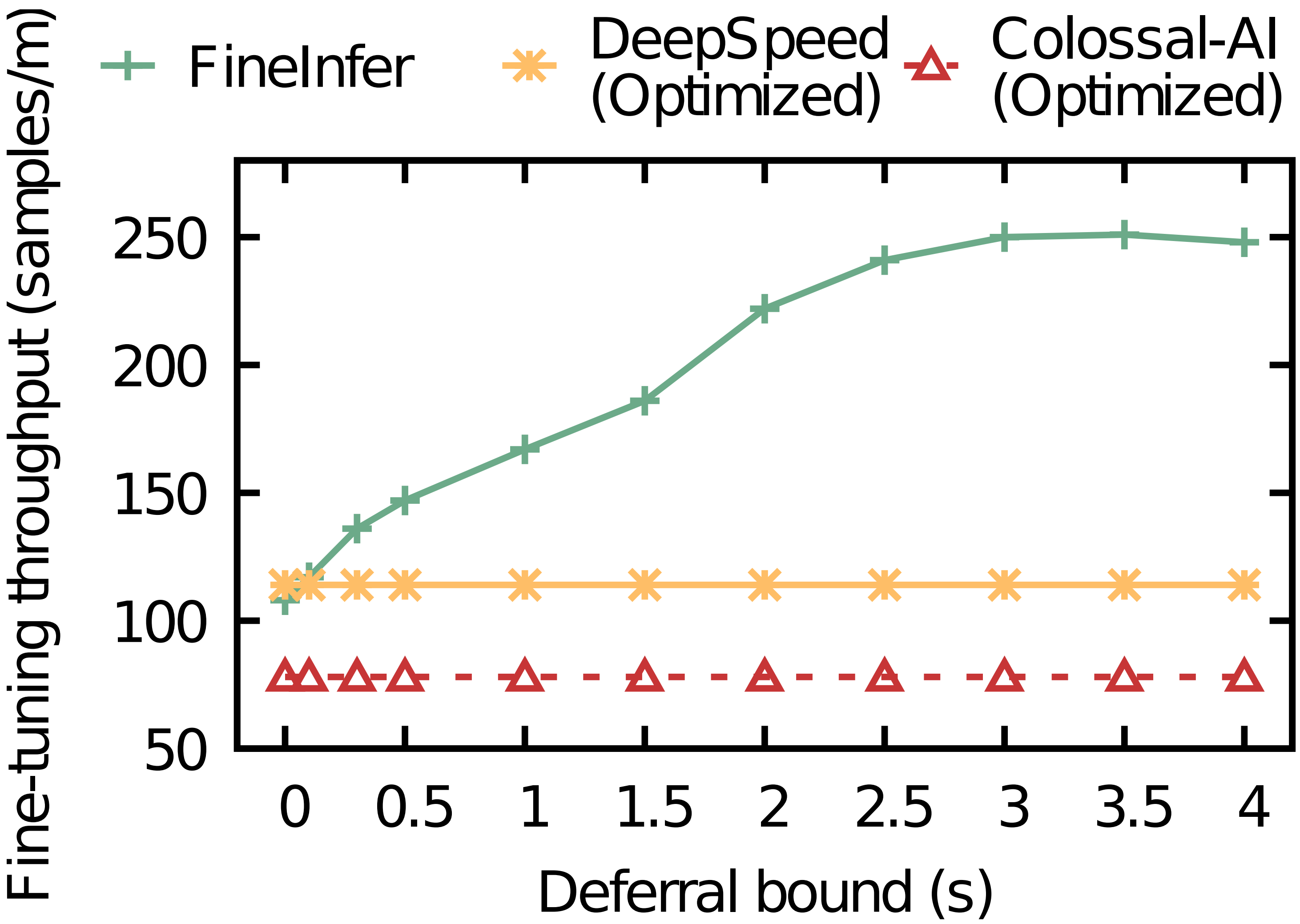
(b) Deferred continuous batching



(c) Deferred continuous batching optimized for larger-than-GPU LLMs

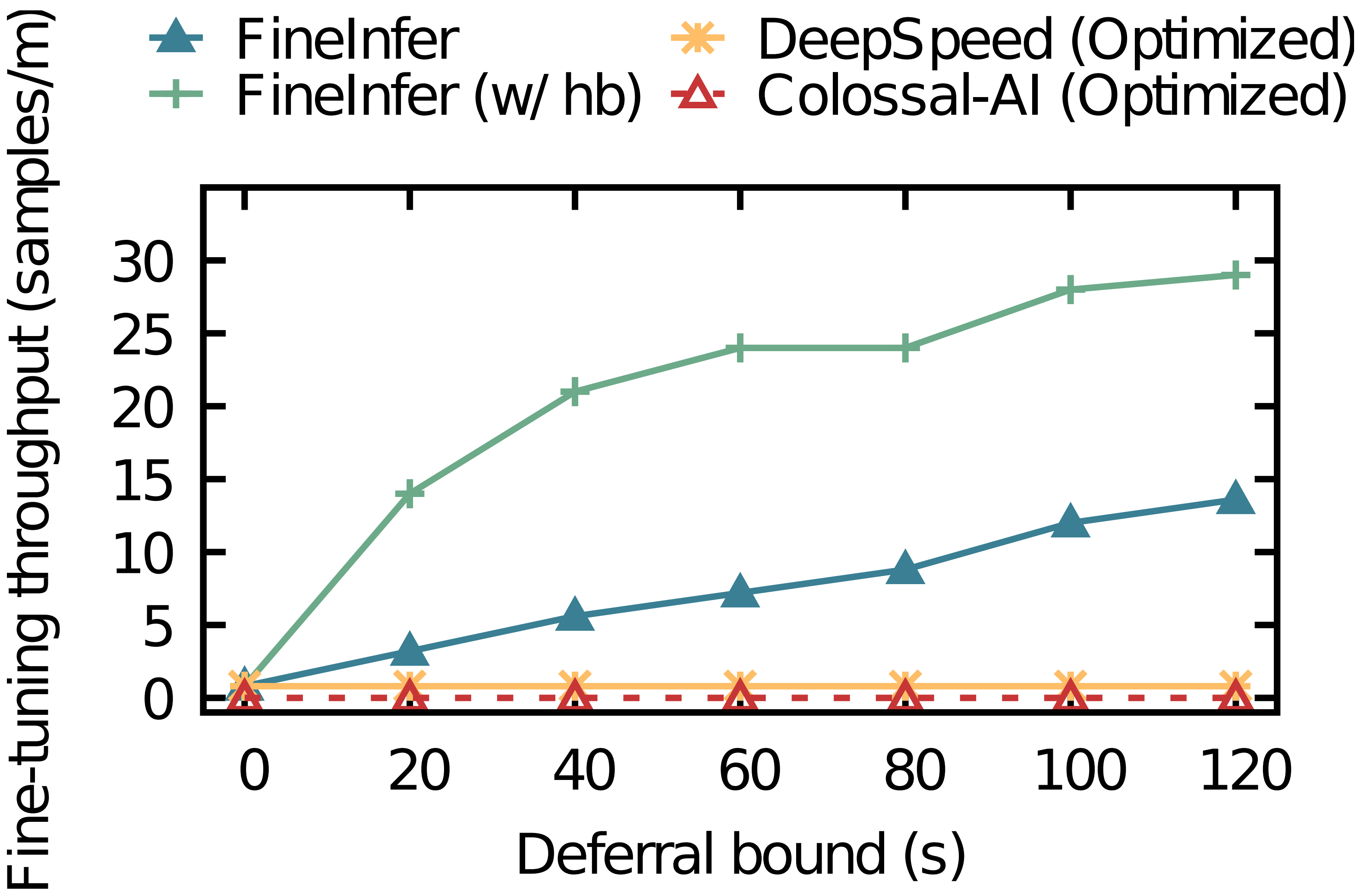
GPU-Resident Performance

Llama2-7B on a 24GB Nvidia 4090 GPU



Larger-than-GPU Performance

Llama2-13B on a 24GB Nvidia 4090 GPU



Summary

We need to evolve systems for LLMs for the new era of the AI PC.

FineInfer = **Fine-tuning** + **Infernece**

- Deferred continuous batching improves fine-tuning throughput by slightly deferring inference requests without violating SLAs
- Hybrid system architecture minimizes context switch and data movement overhead.

Source code: https://github.com/llm-db/FineInfer

Thank you!