dLoRA: Dynamically Orchestrating Requests and Adapters for LoRA LLM Serving

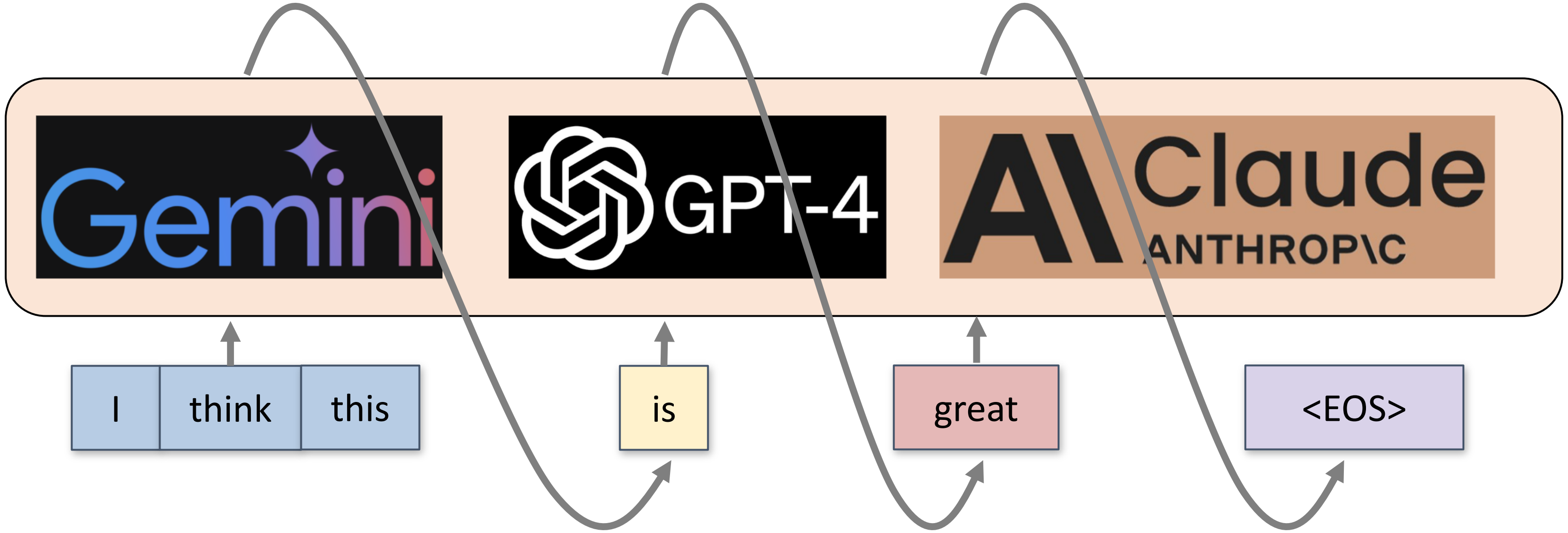
Bingyang Wu **Ruidong Zhu** Zili Zhang
Peng Sun Xuanzhe Liu Xin Jin

PEKING UNIVERSITY
上海人工智能实验室
Shanghai Artificial Intelligence Laboratory

LLMs are changing modern applications

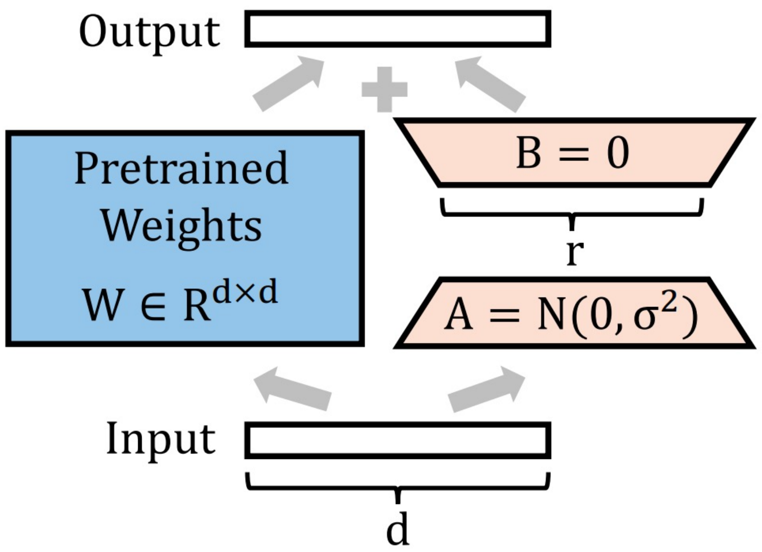
ChatGPT	perplexity	GitHub Copilot
Chat	Search	Program

Autoregressive Generation



LoRA: A popular approach to fine-tune LLMs

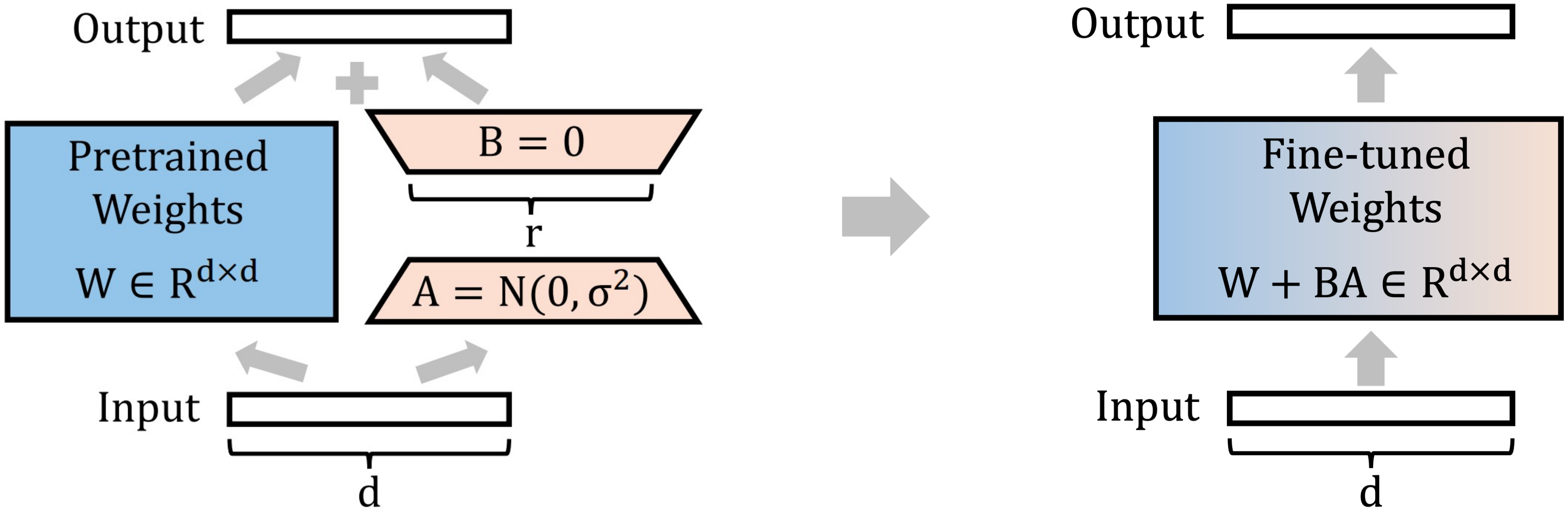
- LoRA: Low-Rank Adaptation
  - $h = Wx + BAx$
- Compared to fully fine-tuning **GPT-3 175B**, LoRA can reduce the number of trainable parameters by **10,000×** and the GPU consumption by **3×**



Hu, Edward J., Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. "**LoRA: Low-Rank Adaptation of Large Language Models**." *arXiv* (2021).

LoRA: A popular approach to fine-tune LLMs

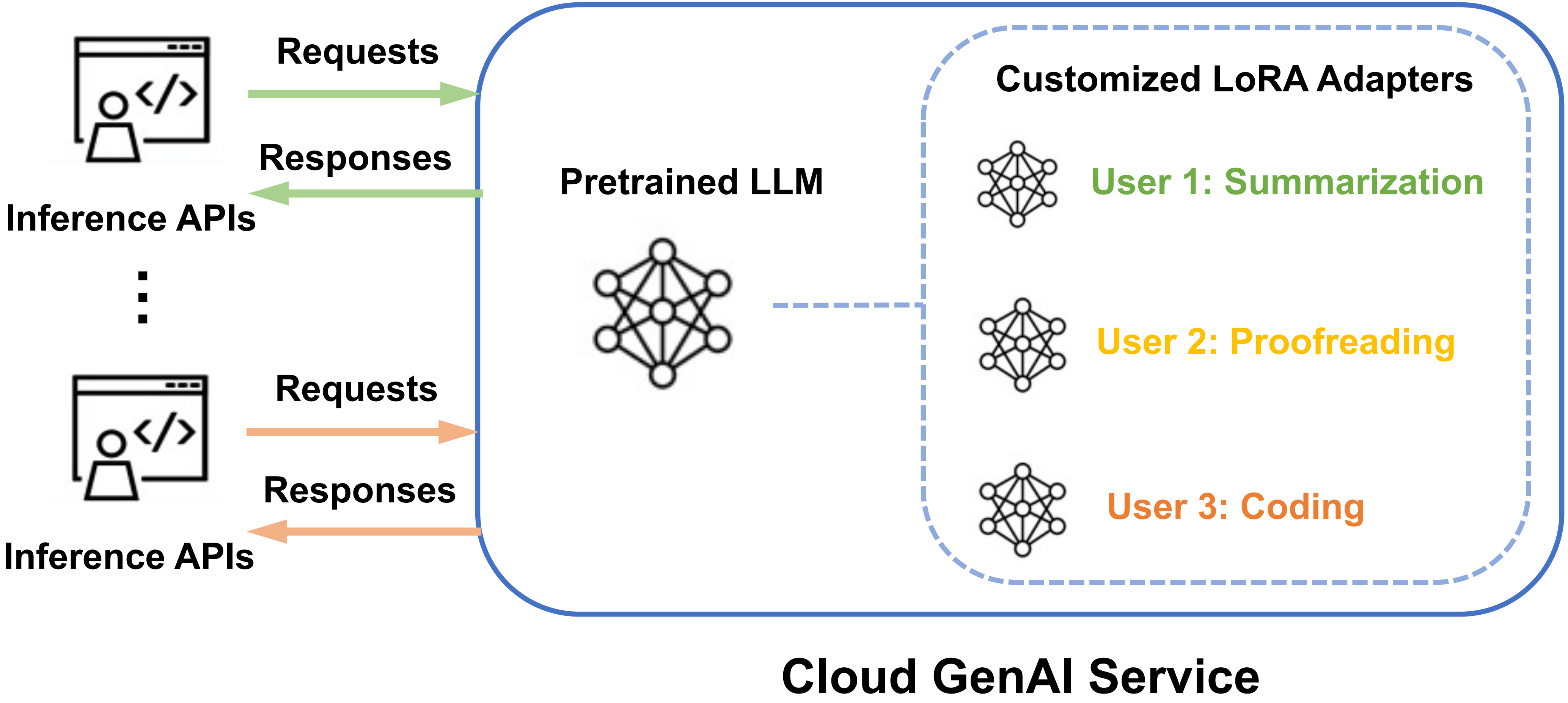
- LoRA introduces **no inference overhead** when serving a single LoRA LLM
  - Merge adapter: $W' = W + BA$
  - Inference with fine-tuned weights: $h = W'x$



Fine-tuning **Merged Inference**

Cloud providers may host many adapters for an LLM

- Different users may use different adapters for different scenarios



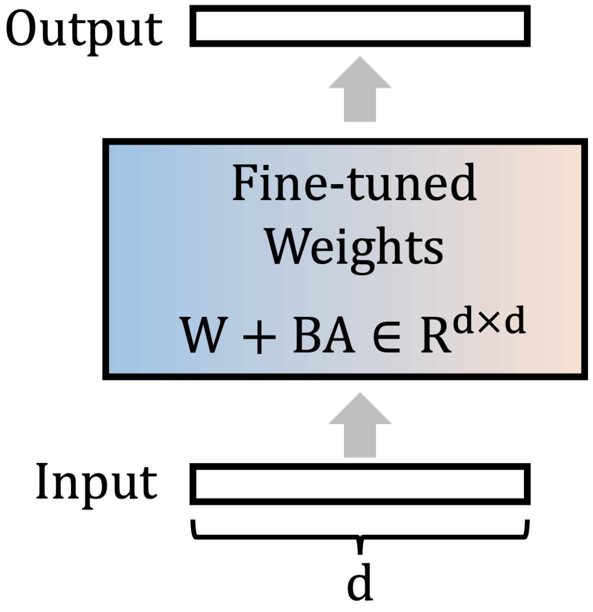
How to serve LoRA LLMs?

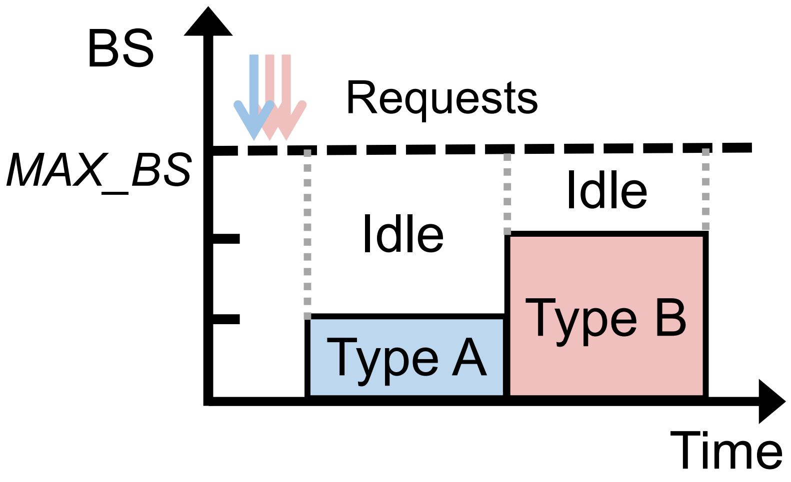
- **LLM serving solutions**
  - vLLM [SOSP'23], Orca [OSDI'22], FastServe
  - Focus on serving a single LLM
- **Traditional model serving orchestrators**
  - AlpaServe [OSDI'23], SHEPHERD [NSDI'23], PetS [ATC'22]
  - Ignore characteristics of LoRA and LLM

- Simply combining these solutions leads to fundamental *inefficiency* at both the **replica** level and **cluster** level

Challenge within replicas: Low GPU utilization

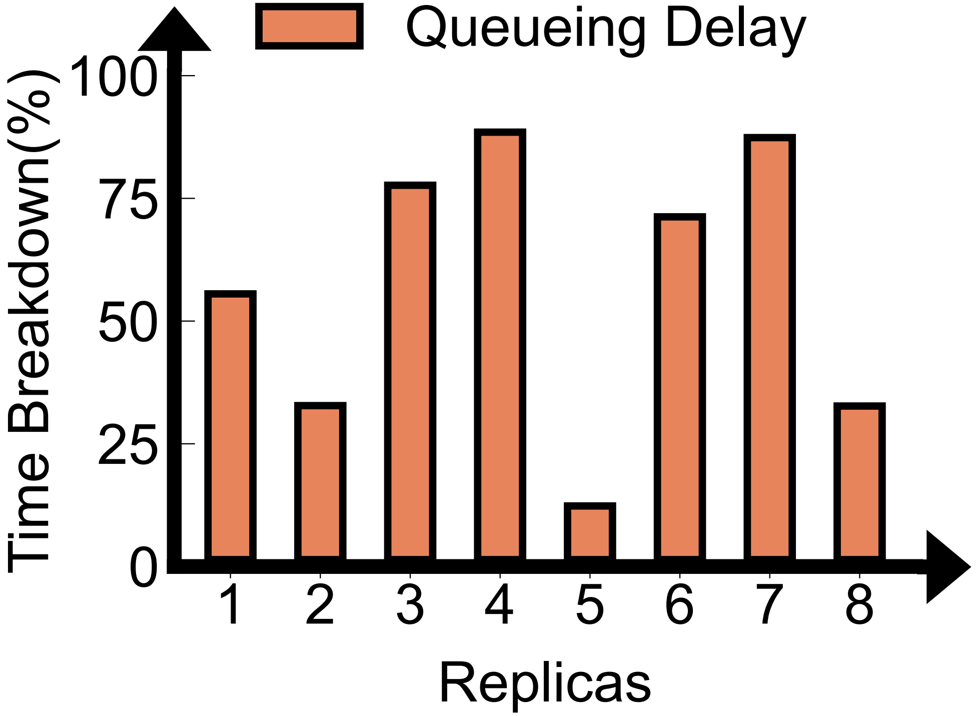
- **Merged inference**: Low utilization when diverse requests arrive
- Example: only 50% GPU utilization





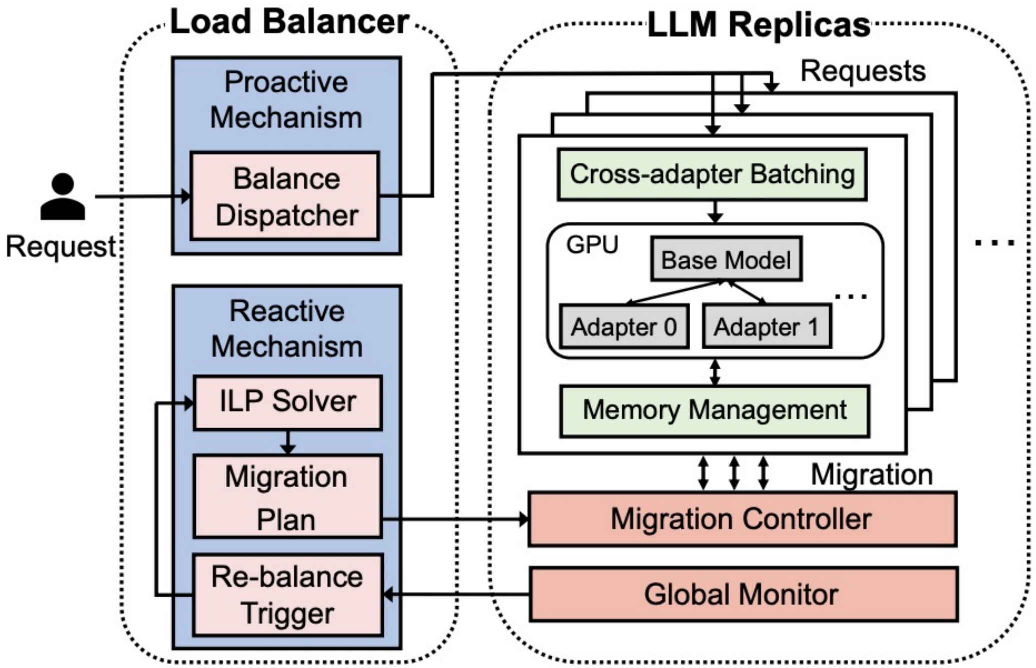
Challenge across replicas: Severe load imbalance

- The burst of variable requests leads to severe load imbalance under static LoRA placement
- Input and output lengths of requests are highly variable



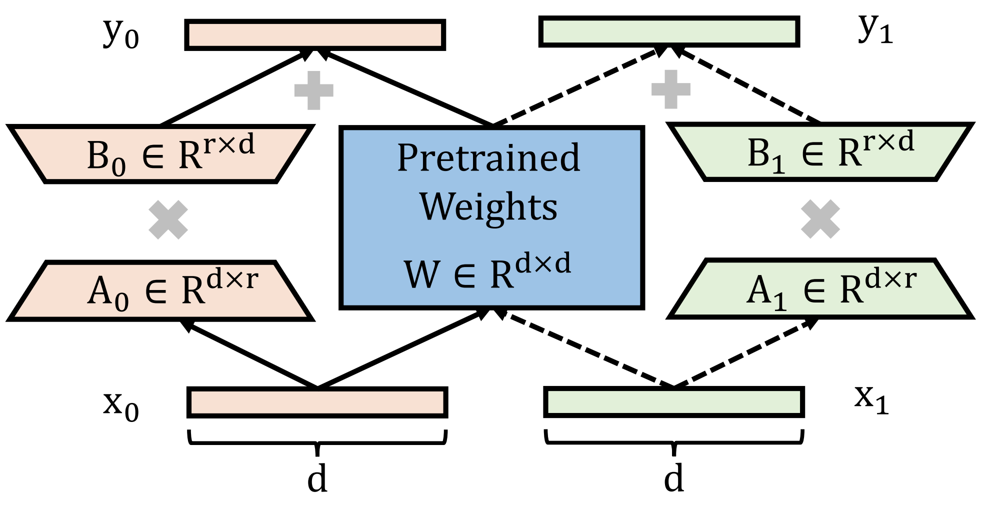
Our design: dLoRA

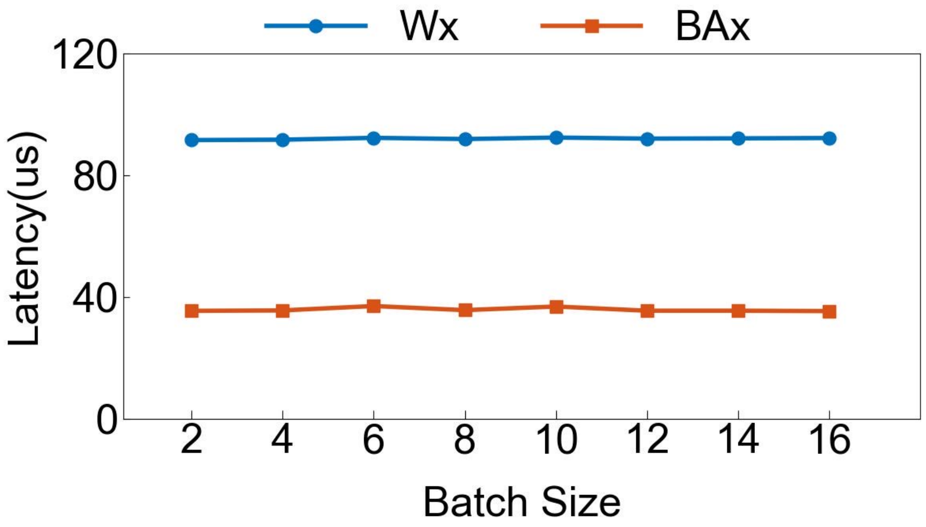
- **Insight:** dynamically orchestrate requests and LoRA adapters
- **Intra-replica**: dynamic batching + memory management
- **Inter-replica**: proactive dispatching + reactive migration



Intra-replica: Strawman solution

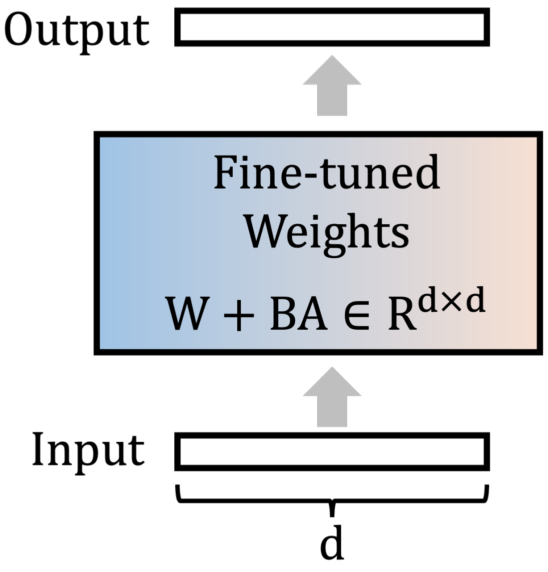
- **Unmerged Inference**: share the same computation among different requests
- **Challenge**: extra computational overhead



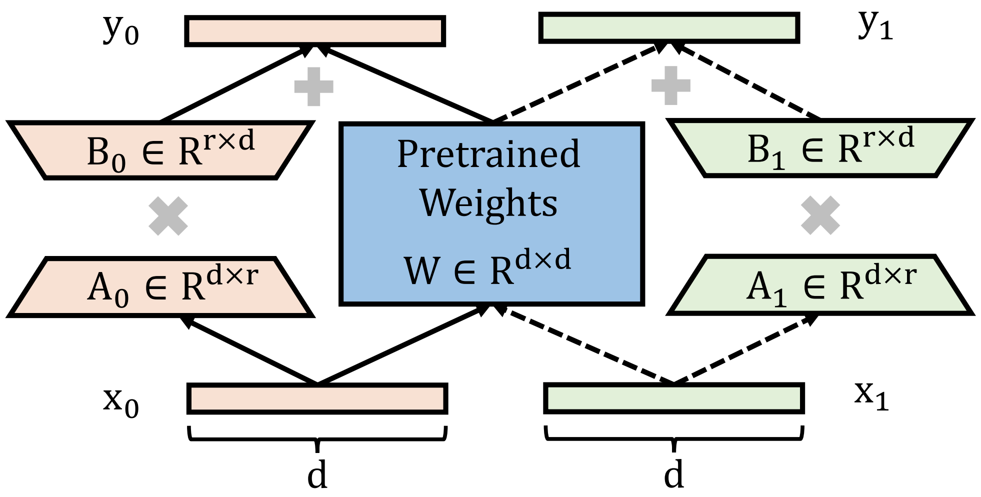


Intra-replica: Dynamic batching

- **Insight**: why not use both of merged and unmerged inference?



No additional inference overhead,
but longer queuing delay



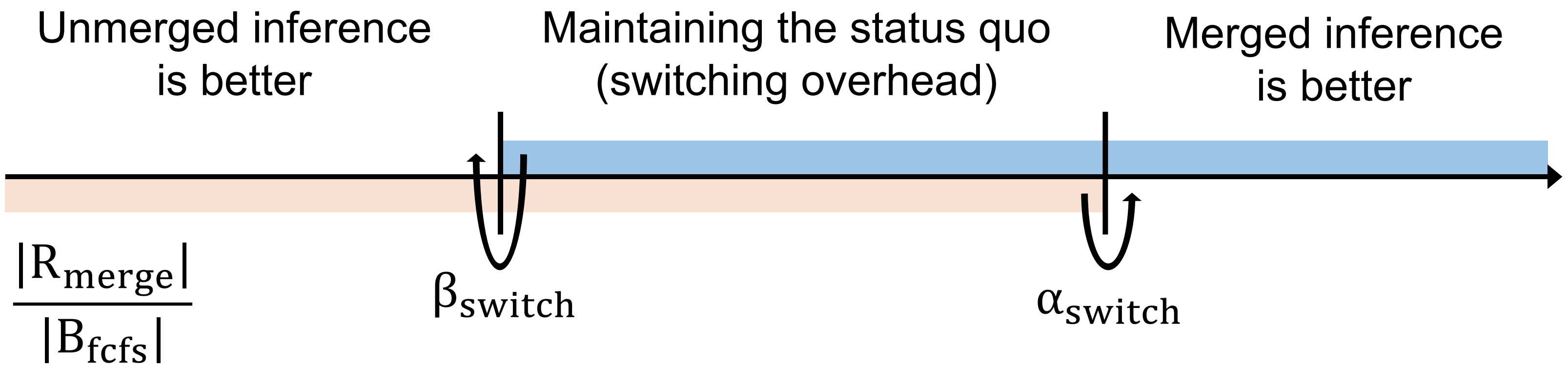
Shorter queuing delay,
but extra computational cost

Intra-replica: Dynamic batching

- **Challenge**: switching overhead + scheduling overhead
  - Switching overhead >= decoding iteration time
  - Complex scheduling at the granularity of the iteration incurs overhead
- **Solution**: dynamic batching algorithm

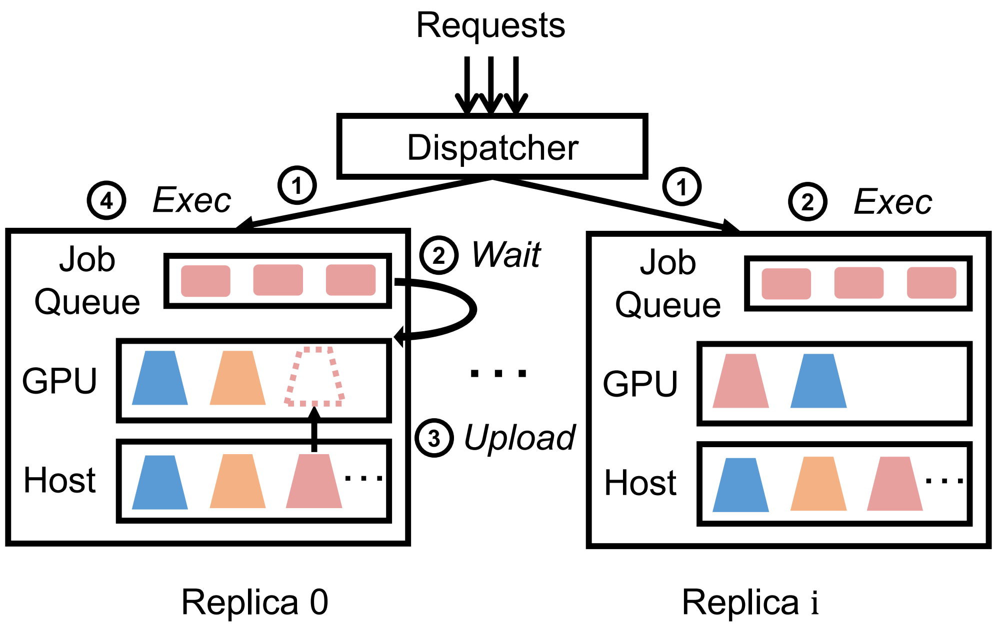
Intra-replica: Dynamic batching

- **Solution**: dynamic batching algorithm
  - B_{fcfs}: set of FCFS requests
  - R_{merge}:
    - When merged: $R_{merge} = \{r_i \in R \mid r_i.type == S.type\}$
    - When unmerged: $R_{merge} = \arg\max_{l_i \in L} |\{r_i \in R \mid r_i.type == l_i\}|$



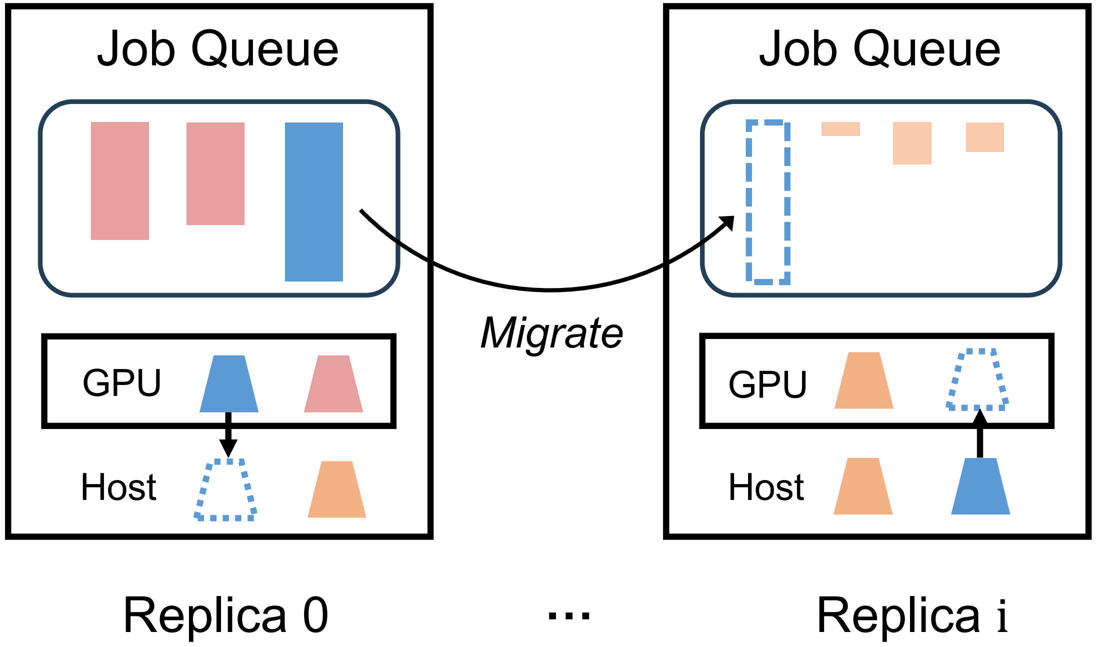
Inter-replica: Proactive dispatching

- Considering both adapter loading time and queuing delay to balance the load
- **Challenge**: requests' unpredictable long output length also causes load imbalance



Inter-replica: +Reactive request-adapter migration

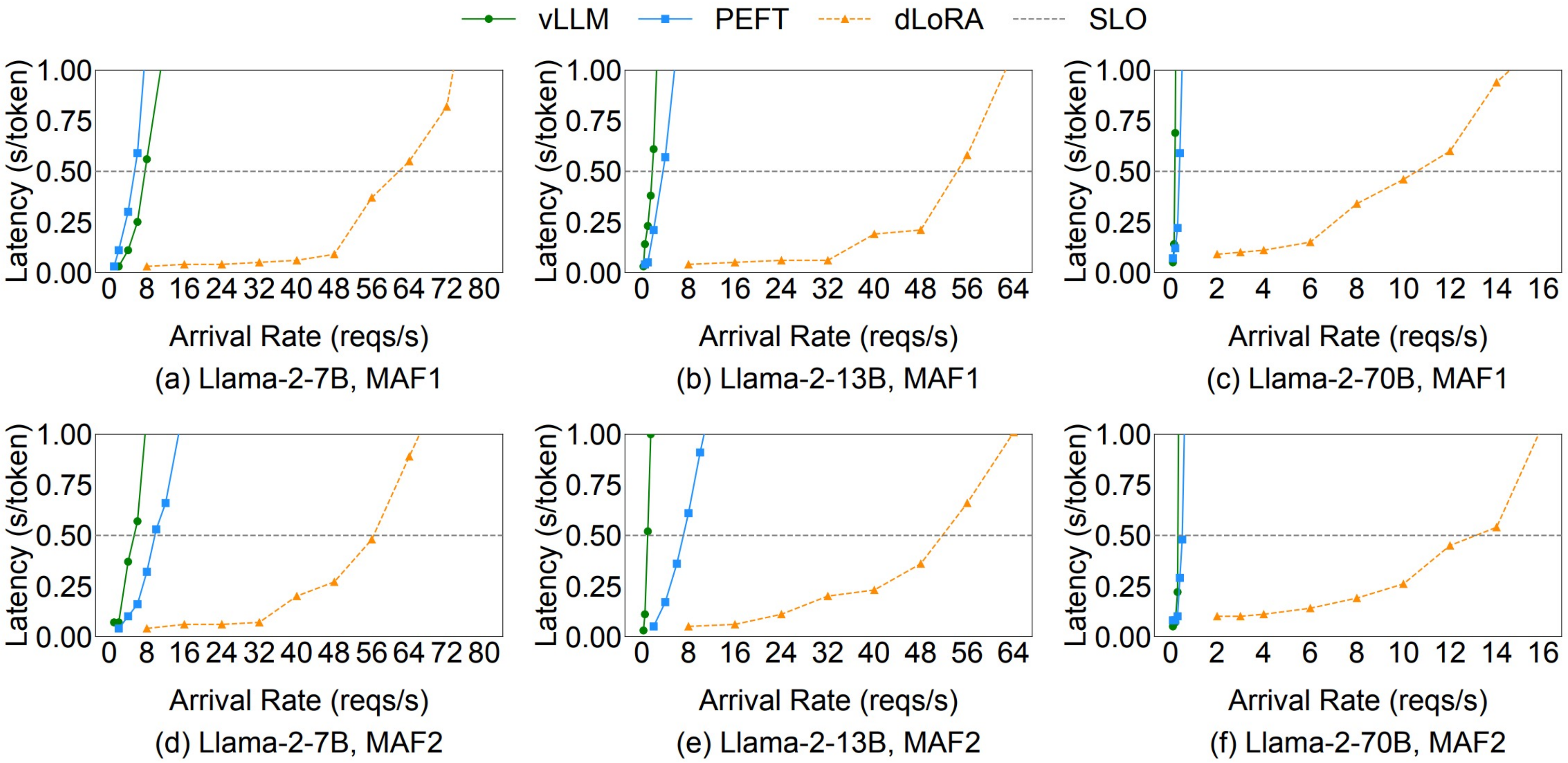
- **Insight**: reactively migrate workload to balance the load
- **Challenge**: dependency between requests and adapters
- **Solution**: reactive request-adapter co-migration algorithm



Experiment setup

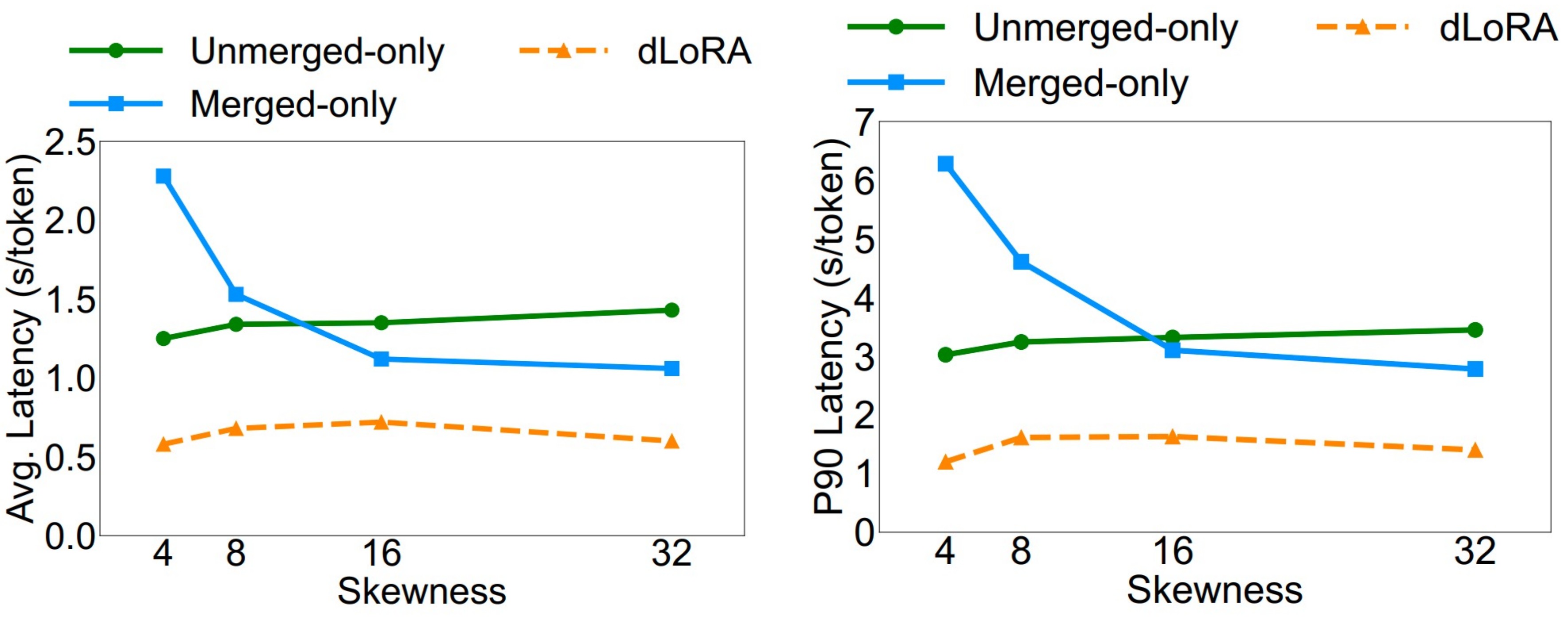
- **Testbed**: 4 nodes * 8 NVIDIA A800-80GB GPUs
- **Models**: Llama-2 (7B, 13B, 70B) + 128 LoRA adapters
- **Workloads**:
  - Dataset: ShareGPT
  - Trace: Microsoft Azure function trace 2019 (MAF1) and 2021 (MAF2)
- **Baselines**:
  - vLLM
  - HuggingFace PEFT

End-to-end performance



dLoRA improves the throughput by up to **57.9×** compared to **vLLM** and up to **26.0×** compared to **PEFT** under the SLO requirement

Effectiveness of dynamic batching

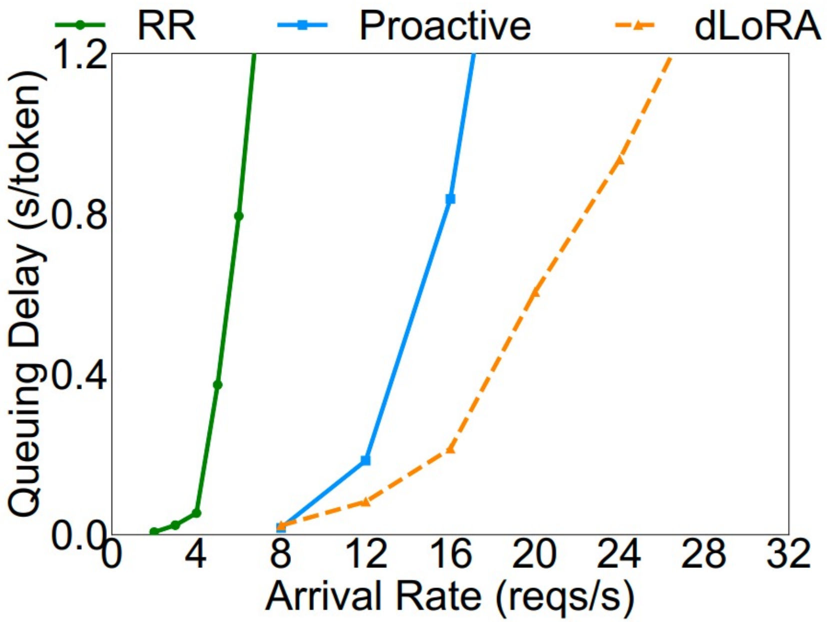


(a) Average Latency.

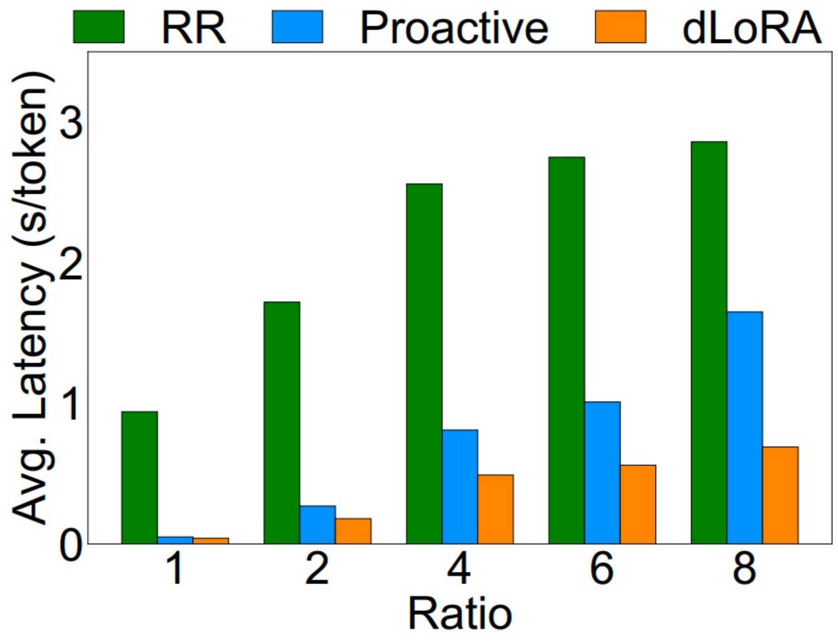
(b) P90 Latency.

dLoRA consistently outperforms both **merged-only** and **unmerged-only** under diverse skewness

Effectiveness of dynamic load balancing



(a) Reduction in Queuing Delay.



(b) Stability under Different Ratios.

dLoRA outperforms **RR** by **3.6x** and **proactive dispatch** by **1.4x** under the SLO requirement

Conclusion

- **dLoRA: an efficient serving system for multi-LoRA LLMs**
  - **Intra-replica:** *dynamically* merges and unmerges adapters
  - **Inter-replica:** *dynamically* migrates both requests and adapters

Thanks!

bingyangwu@pku.edu.cn