EuroSys 2025

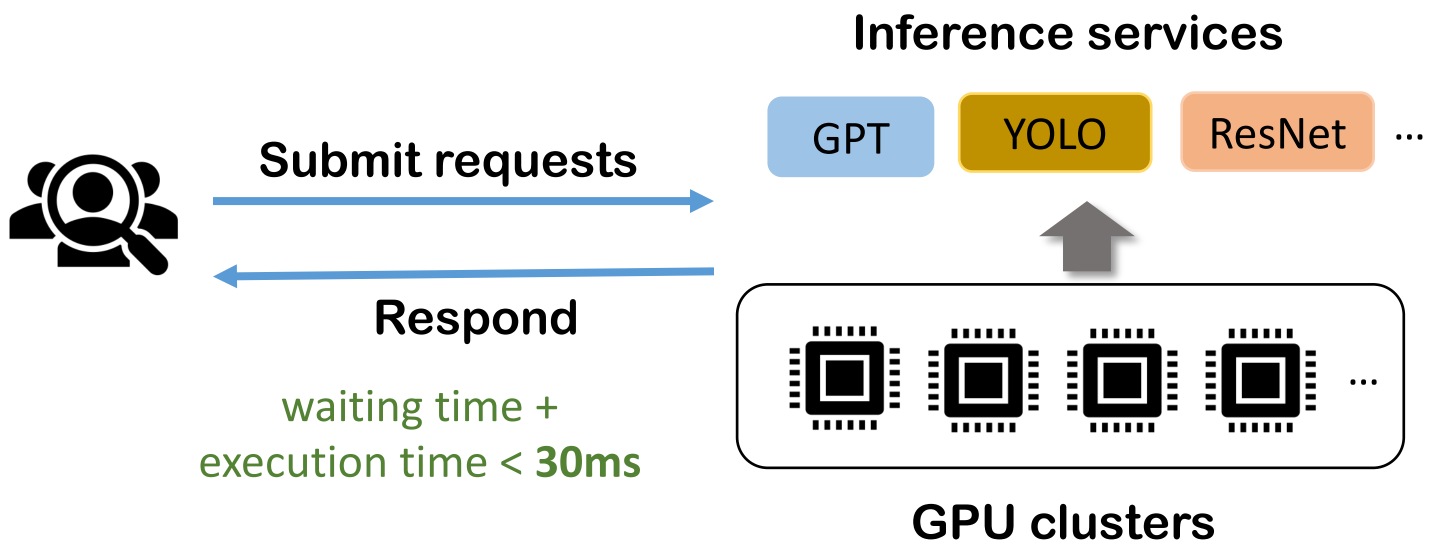
Multiplexing Dynamic Deep Learning Workloads with SLO-awareness in GPU Clusters

Wenyan Chen[1,2], Chengzhi Lu[1,2,3], Huanle Xu[1], Kejiang Ye[2] and Chengzhong Xu[1]

[1]University of Macau
[2]Shenzhen Institute of Advanced Technology, Chinese Academy of Sciences
[3]University of Chinese Academy of Sciences

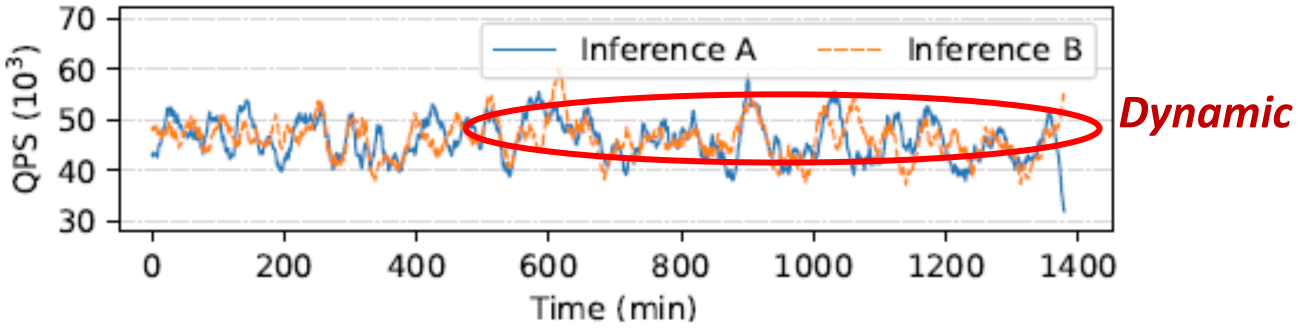
Deep learning (DL) inference with GPUs

- **GPUs** are widely used as inference accelerators
- Service-level objective (**SLO**) must be satisfied
- **Batching** is used to handle inference requests

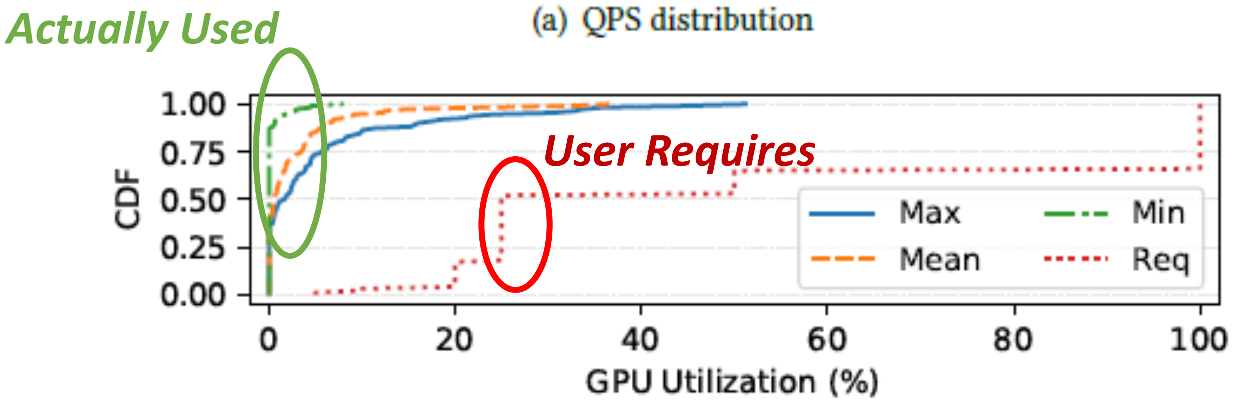


DL inference in GPU clusters

- **Inference requests -** Fluctuating and unpredictable
- **GPU resource –** Underutilization and over-provisioning



(a) QPS distribution

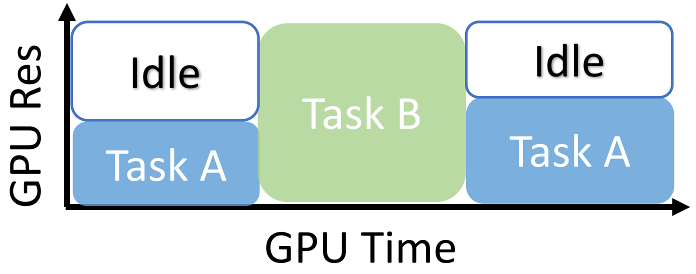


(b) GPU utilization

Approaches to Improve GPU Utilization

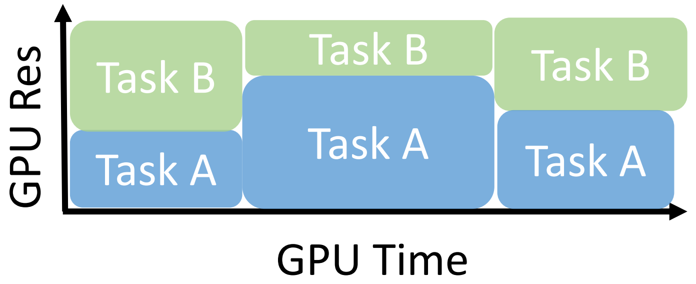
Packing multiple tasks on the same GPU via **time sharing** or **spatial sharing**

Time sharing



Still underutilize spatial resources

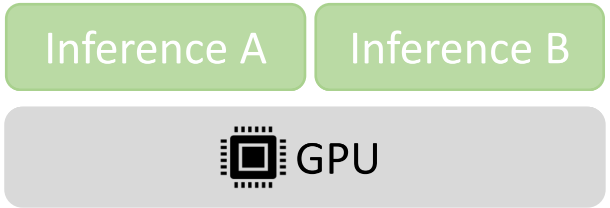
Spatial sharing

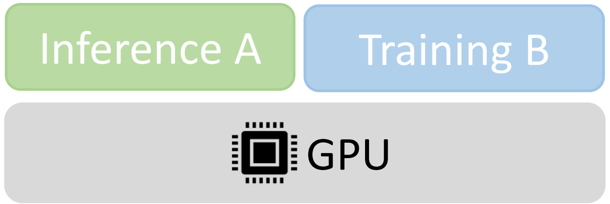


- **MPS** *More flexible* **Good!**
  - Small granularity of resource allocation (1%~100%)
- **MIG** *Less flexible; Higher Cost*
  - Limited resource allocation strategies available (18 cases)
  - Large allocation strategy change overhead (restart all instances)

Spatial Multiplexing of GPUs

Inference with inference or **Inference with training**





Interference of Multiplexing DL tasks

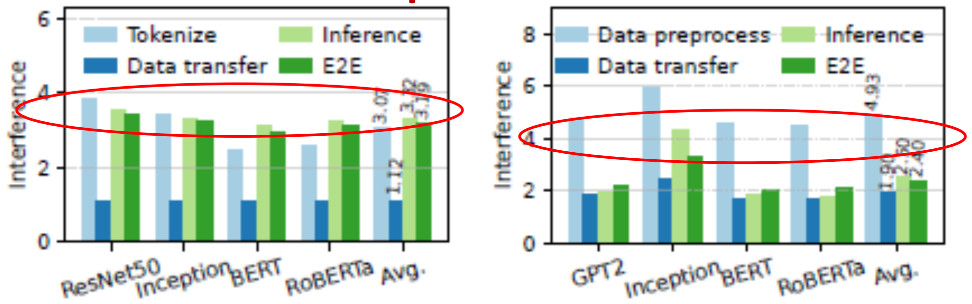
- **Breakdown** the executing process of inference

Tokenize/Data Preprocess | Data Transfer | Inference

- **Observation**: e2e interference on inference is smaller when multiplexing inference with training

  - **Inference with inference**

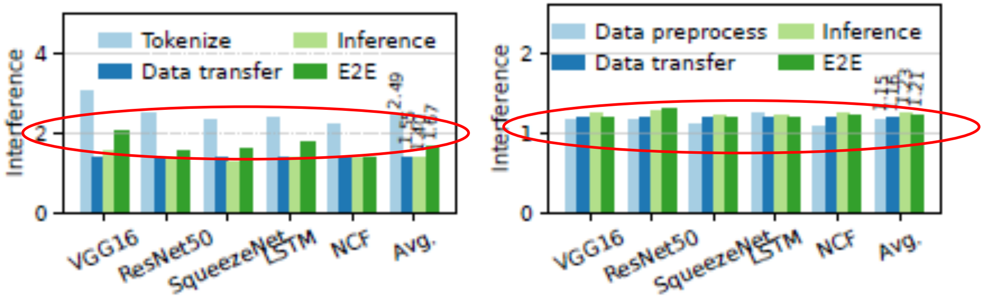
Up to 3.19x



(a) GPT2 (b) ResNet50

  - **Inference with training**

Below 1.67x



(a) GPT2 (b) ResNet50

Interference of Multiplexing DL tasks

In-depth analysis of the reasons

- ***Tokenize/Data Preprocess***: Parallel, requiring substantial CPUs for execution, leading to CPU contention
- ***Data Transfer***: The frequency of data transfers required by training is less than that of inference
- ***Inference***: The control flow accounts for up to 72%[1] of the total execution time in the inference stage
- GPU and memory utilization is high when inference is multiplexed with training

Multiplexing inference service with training tasks is more beneficial

[1] Chen Zhang, Lingxiao Ma, Jilong Xue, et al. Cocktailer: Analyzing and Optimizing Dynamic Control Flow in Deep Learning. In Proceedings of OSDI. 2023.

Can we maintain low latency for inference and high throughput for training? 😕

Low latency

Inference services

High throughput

Training models

Challenges of Multiplexing

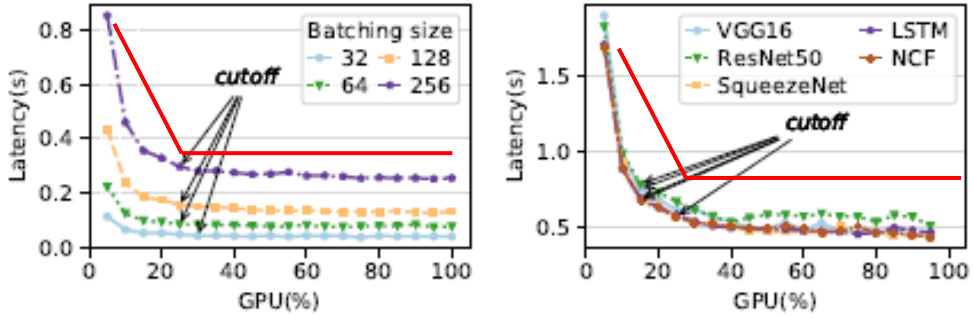
- C_1: Dynamic workload arrivals
  - Unpredictable QPS for inference and unobserved training tasks
- C_2: Intricate Coupling
  - Cluster-wide task colocation and device-level configuration
- C_3: Large optimization space
  - Packing patterns, SM% and batching sizes

Mudi – New Multiplexing system for highly dynamic DL workloads that prioritizes SLO-awareness for inference

Key Ideas

I_1: Explicit modeling of inference latency

- Use piece-wise linear function to fit the relationship between latency and resource partitions
- Address large-optimization space
- Seamless coordination of cluster-wide co-location and device-level interference control



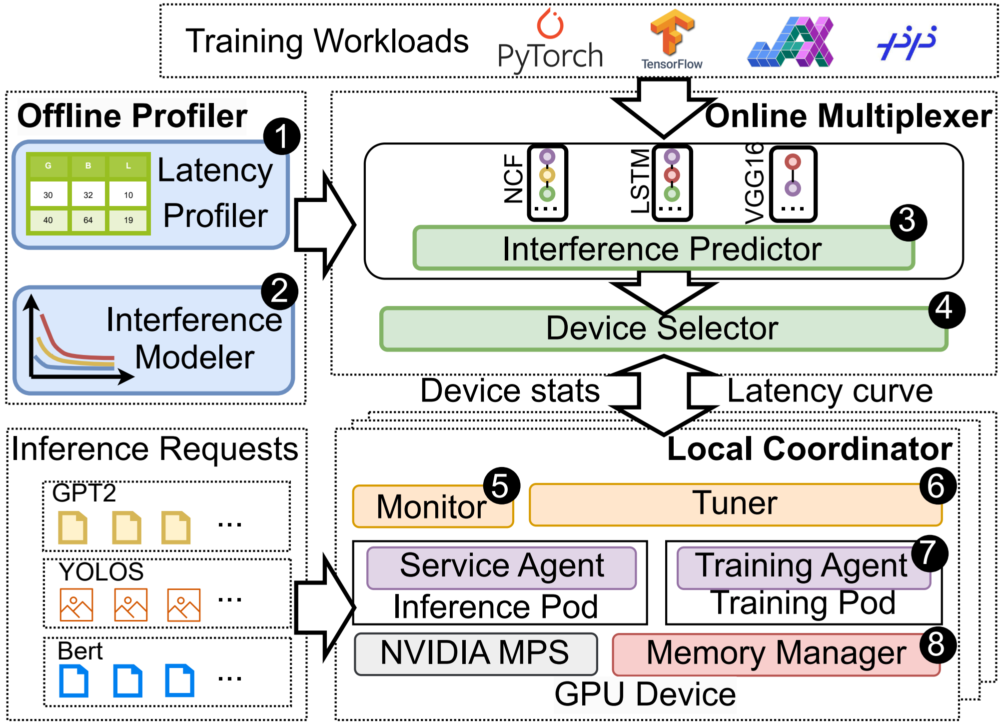
(a) GPT2 with solo-run (b) GPT2 with colo-run

I_2: Predicting interference using underlying network architectures

- Use network architecture to estimate the **slope** of piece-wise linear function
- Adapt to dynamic training workloads

System Architecture: Mudi

- **Offline Profiler**: Profile the interference latency curves of co-located tasks
- **Online Multiplexer**: Record the requests and make packing decisions
- **Local Coordinator**: Monitor QPS of each inference and update configurations

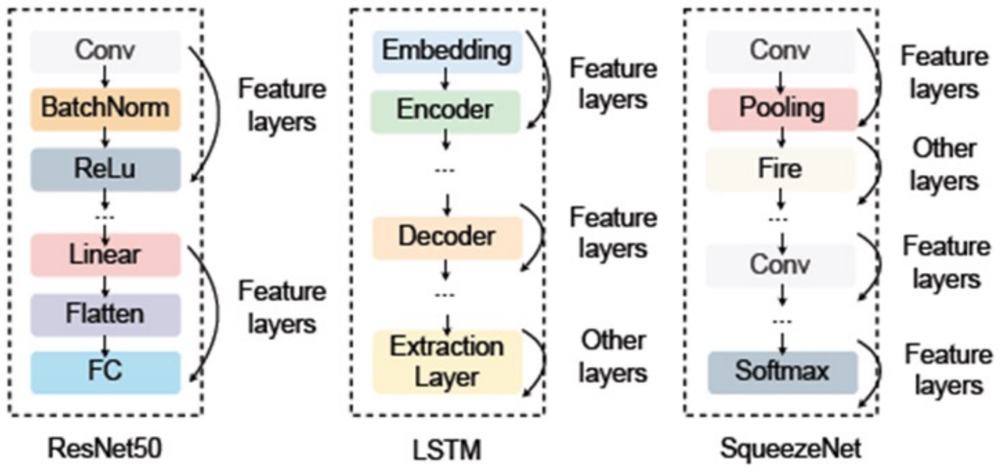


Inference Latency Quantification

- **Inference Latency Profiling**
  - Fit piece-wise linear functions for each inference with various training tasks

$$L_{b,\Psi}^{i} = \begin{cases} k_{\Psi,1}^{i} \cdot (\Delta_i - \Delta_0) + l_0, & \Delta_i \leq \Delta_0, \\ k_{\Psi,2}^{i} \cdot (\Delta_i - \Delta_0) + l_0, & otherwise. \end{cases}$$

- **Online Prediction**
  - Utilize the network layers (which and how many) and configs (bs, GPU%) as Interference Predictor's input
  - Online Multiplexer forecasts the interfered latency of inference based on offline profiles



Online Multiplexing Approach

- **Optimization Model**
  - Objective
    - Minimize the overall training time of all co-located training tasks on each device
  - Constraint
    - The inference latency should meet the SLO of each inference request

Placement — GPU% — Batching size

$$\min_{\{x_j^i, b_i, \Delta_i\}} \sum_{j \in A(t)} \sum_{i=1}^{N} x_j^i \cdot \mathrm{Iteration}_j(b_i, \Delta_i)$$

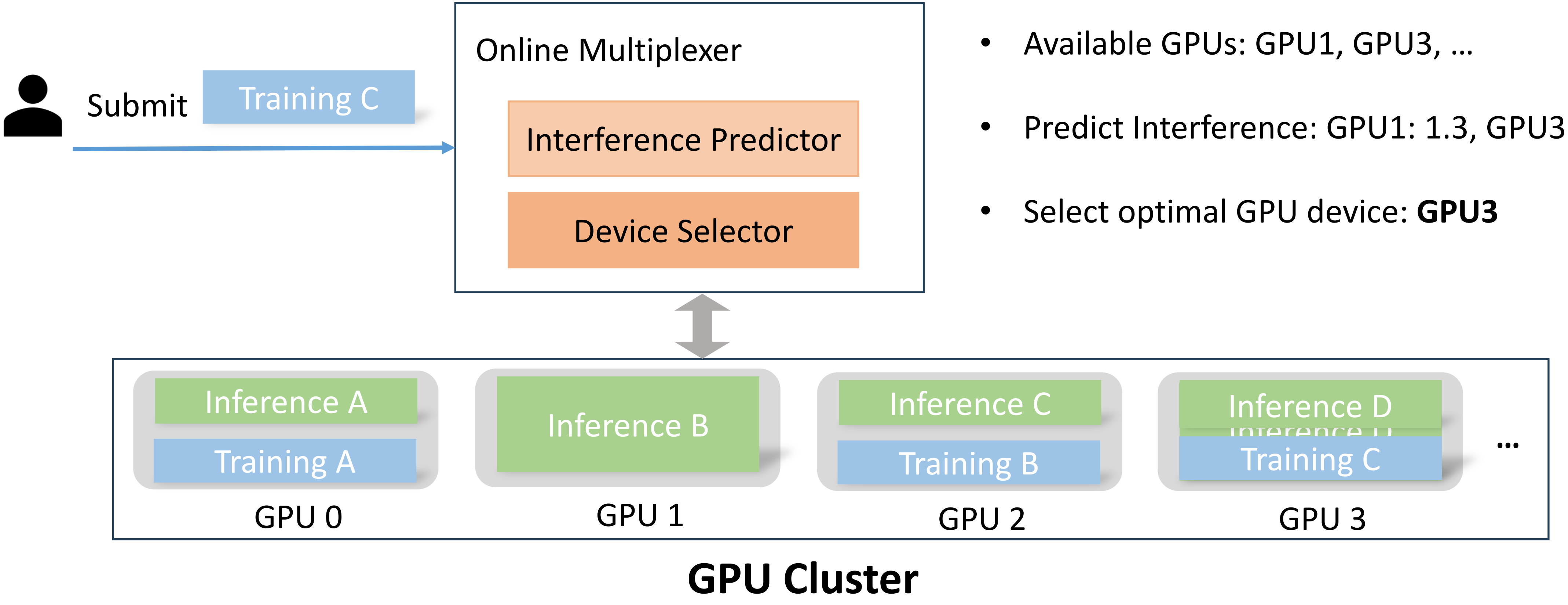
$$s.t., \quad \frac{W_i}{b_i} \cdot P_i(b_i, \Delta_i, \Psi_j) \le \mathrm{SLO}_i, \ \forall 1 \le i \le N,$$

$$\Delta_i \le 1, \ \forall 1 \le i \le N,$$

$$\sum_{i=1}^{N} x_j^i = 1, and\ x_j^i \in \{0, 1\}.$$

Cluster-wide workload co-location

- **Cluster-wide co-location** *Find the best Placement*
  - The Device Selector assigns training task to the device that yields the smallest slope



- Available GPUs: GPU1, GPU3, …
- Predict Interference: GPU1: 1.3, GPU3: 1.5, …
- Select optimal GPU device: **GPU3**

Device-level Multiplexing

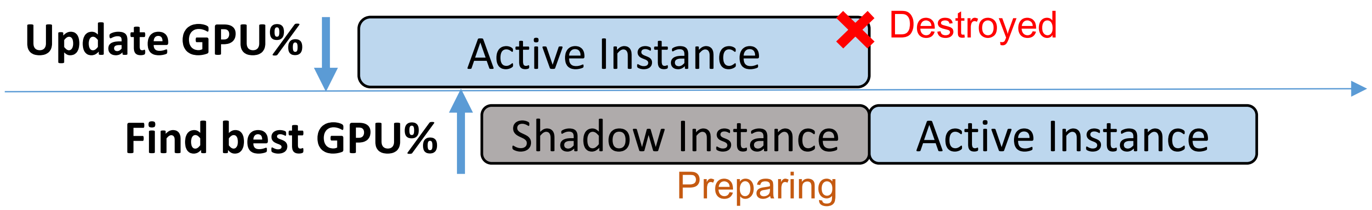
- **Adaptive Batching** *Find the optimal Batching size*
  - Use **Gaussian Process (GP)** as surrogate model and acquisition function based on the **lower confidence bound (LCB)** to guide the exploration process

$$\min_{b_i \in \mathcal{R}} \mathcal{A}(b_i) = \mu(b_i, \Delta_i) - \beta_n^{1/2} \sqrt{\sigma(b_i, \Delta_i)}.$$

- **Dynamic Resource Scaling** *Find the optimal GPU%*
  - Find the optimal GPU% while meeting SLOs using CVXPY

$$\Delta_i = \operatorname{argmin} \Delta,\ s.t.,\ \mathrm{W}_i / b_i \cdot \mathrm{P}_i(b_i, \Delta, \Psi_j) \leq \mathrm{SLO}_i$$

  - Use **shadow instance** to overlap the restarting cost of updating GPU%



Optimality Analysis

- Identify the optimal co-location 92.67%
- **Iteration time** bounded by 1.10x
- **SLO violation** bounded by 1.08x (Optimal as 1.0)

All training jobs

Prediction Accuracy of optimal co-location

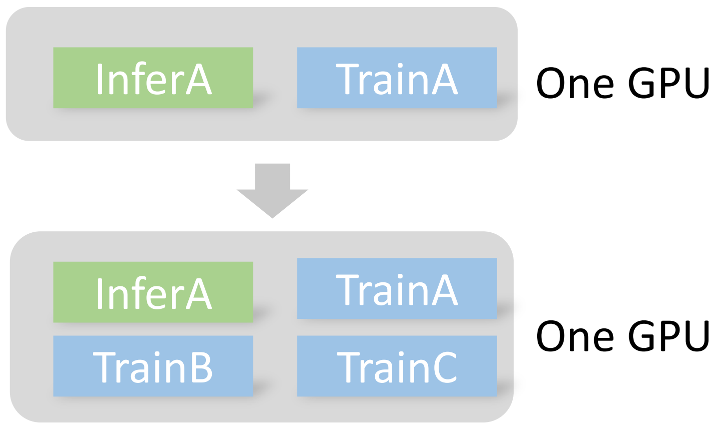
$$\mathcal{E} \leq \frac{1}{M}\sum_{j=1}^{M}\left(\mathcal{P} \cdot \boxed{\text{Iteration}_j^*} + (1-\mathcal{P}) \cdot \boxed{\text{Iteration}_j^\dagger}\right)$$

Optimal cases

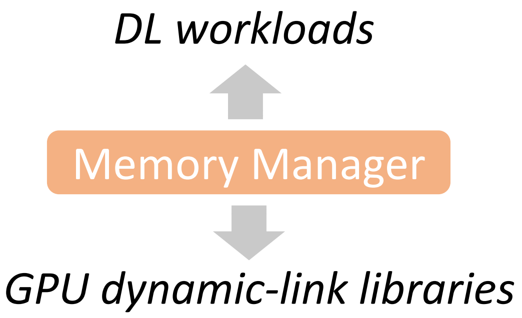
Worst cases

System Optimization

- **Extension to Multiplexing more tasks**
  - No more than three training (IADeep SC'23)
  - Profile more samples (one inference with two/three training)
  - Designate cumulative feature layers as ψ
  - Evenly distribute the unassigned resource partitions



- **Memory Management**
  - Prevent out-of-memory errors
  - Dynamically swap memory between GPU and host for training tasks
  - A middleware between DL tasks and dynamic-link libraries



Experimental Setup

- **Physical cluster –** 3 physical servers with each equipes 4 A100 GPUs
- **Large-scale cluster** – use 1000 processes to simulate a 1000-GPU large-scale cluster based on more profiles
- **Baselines** – GSLICE (SoCC'20), MuxFlow (ByteDance), gpulets (ATC'22)
- **DL workloads** – arrival rates follow Microsoft trace

Table 1. Inference services with SLOs from various domains

Field	Model	Dataset	Param (M)	SLO (ms)
♦	ResNet50 [25]	ImageNet [13]	25.6	150
♦	Inception [65]	ImageNet	23.8	120
★	GPT2 [52]	SQuAD [53]	335	100
♡	BERT [14]	SQuAD	110	330
♣	RoBERTa [40]	SQuAD	125	110
♠	YOLOS [16]	COCO [39]	30.7	2200

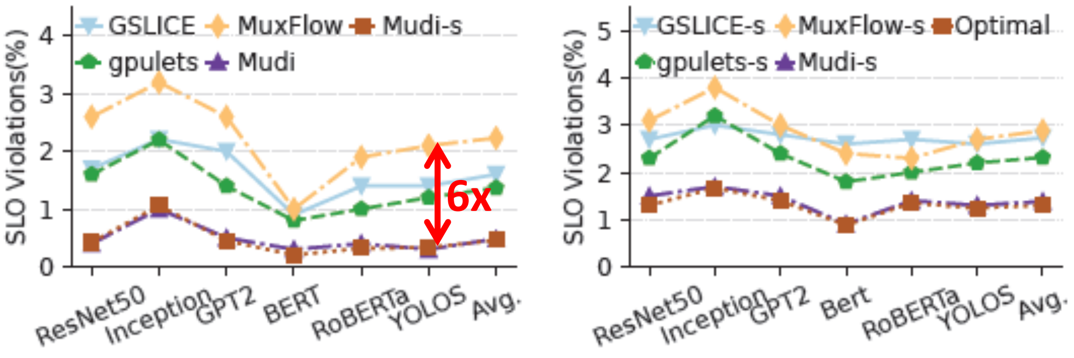
♦ Image Classification ★ Text Generation ♡ Language Modeling ♣ Question Answering ♠ Object Detection.

Table 3. DL training tasks from various domains

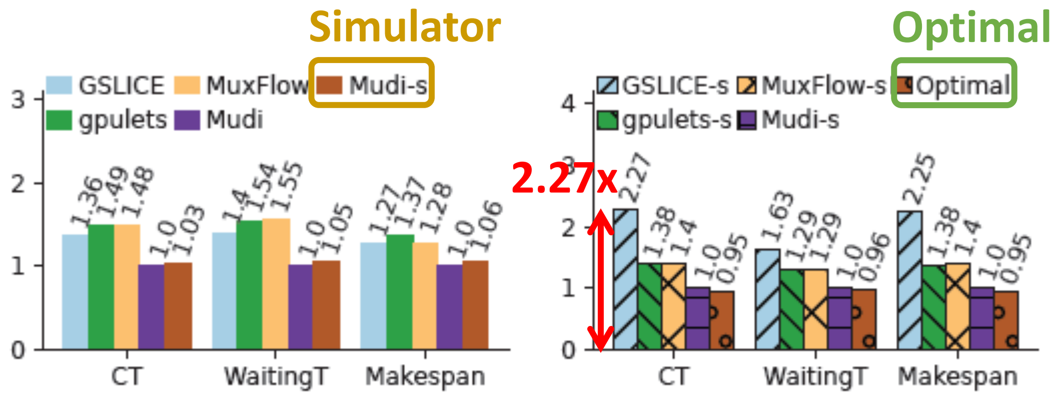
Field	Task Name	Dataset	Optimizer	batchsize	Size	Frac.
♦	VGG16 [60]	CIFAR10 [36]	Adam	512	S	14%
♦	SqueezeNet [31]	CIFAR10	Adam	512	S	14%
♦	ResNet50 [25]	CIFAR100 [36]	Adam	1024	S	14%
▷	NCF [26]	MovieLens [23]	SGD	1024	M	12%
♣	LSTM [49]	Wikitext-2 [42]	Adadelta	256	M	12%
□	AD-GCL [64]	Reddit[3]	Adam	64	M	12%
♣	Bert [14]	SQuAD [53]	AdamW	32	L	12%
♠	YOLOv5 [33]	COCO [39]	SGD	64	L	10%
♦	ResNet18 [25]	ImageNet [13]	SGD	128	XL	2%

♦ Image Classification ▷ Recommendation System □ Social Network ♡ Language Modeling ♠ Object Detection ♣ Question Answering.

End-to-End Performance



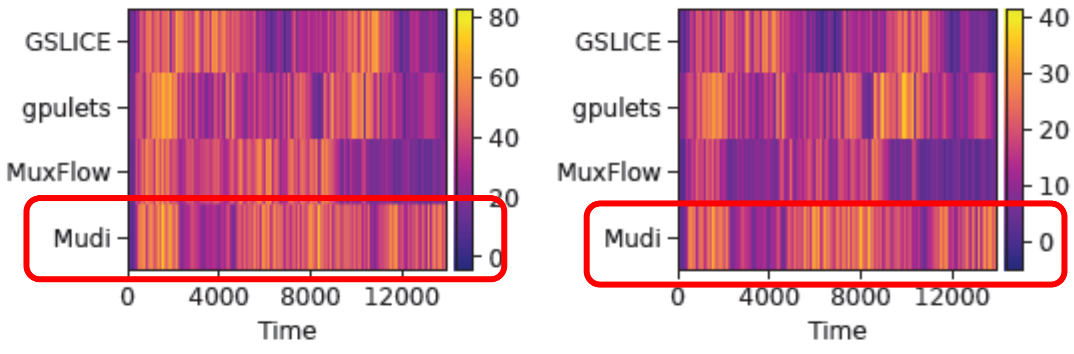
(a) Small-scale (b) Large-scale



(a) Small-scale (b) Large-scale

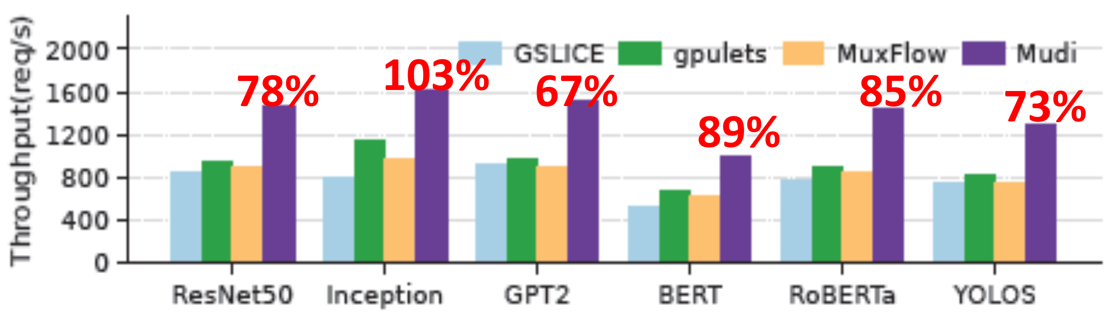
- **SLO violations for inference**
  - As low as 0.5% (1.2%) in small / large cluster
  - Achieve up to 6x SLO violation reduction
- **CT for Training**
  - Achieve up to 2.27x CT reduction
- **Simulator Fidelity**
  - Minor discrepancies of < 4.7% in SLO violation and CT
- **Optimality Analysis**
  - Discrepancy of SLO violation is only 5.86%
  - Training performance deviates from Optimal by no more than 5%

End-to-End Performance



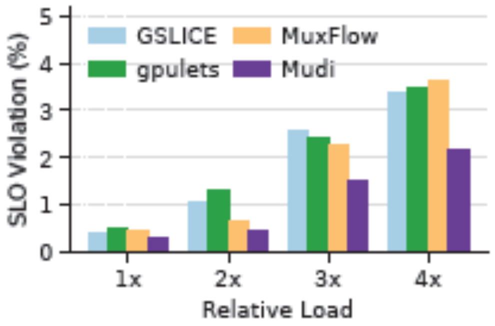
(a) SM utilization (b) Memory utilization

- **GPU Utilization in physical cluster**
  - SM utilization improvement 37%
  - Memory utilization improvement 19%
  - Effective co-design of cluster-level and device-level multiplexing

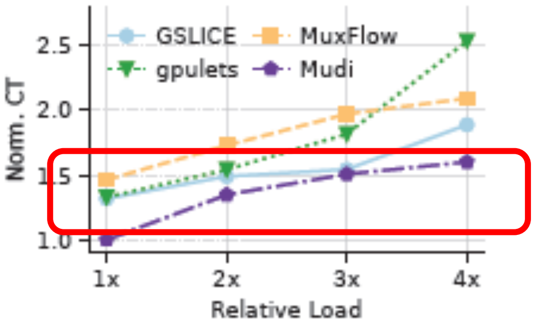


- **System Throughput**
  - Increase requests loads until the SLO is not satisfied
  - Achieve up to 103% throughput for all inferences

End-to-End Performance

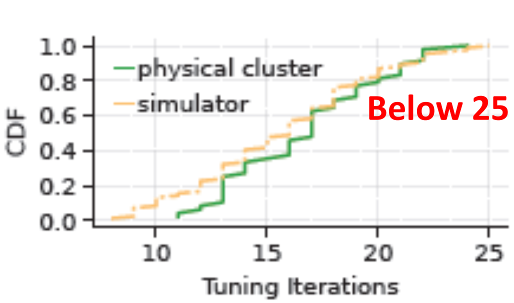


(a) SLO violation

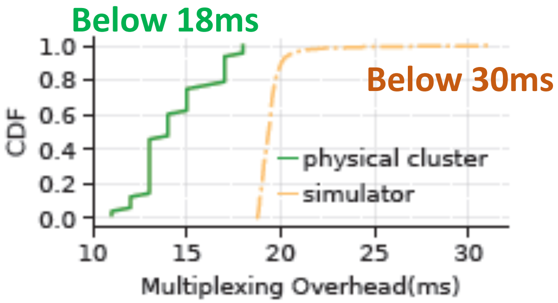


(b) Norm. CT

- **Sensitivity to Heavy loads**
  - Higher SLO violations / longer CTs with increasing loads
  - Mudi exhibits a nonlinear increase and surpasses the baselines for all cases



(a) Tuning iterations



(b) Multiplexing overhead

- **System Overhead**
  - Tuning iteration <25
  - Multiplexing overhead <18ms in physical cluster and <30ms in simulated cluster

More evaluations

- **Accuracy of interference modeling**
- **Effectiveness of cluster-level co-location**
- **Effectiveness of per-device control**
- **Capability to handle more training tasks**
- **…**

Summary

😭 **Problem:** How to maintain low latency of inference and high throughput for training?

🤔 **Key Insight** - Multiplexing training with inference has **much lower** interference on inference services

💡 **Key Ideas**

- Explicit modeling of inference latency using piece-wise linear functions
- Predicting interference using underlying network architectures

🎉 **Results** - Mudi reduces CT of training by **2.27x** with **SLO compliance** for inference requests

✉ yc17498@um.edu.mo

Thanks & QA