Interference-aware Multiplexing for Deep Learning in GPU Clusters: A Middleware Approach

Wenyan Chen, Zizhao Mo, Huanle Xu, Kejiang Ye, Chengzhong Xu

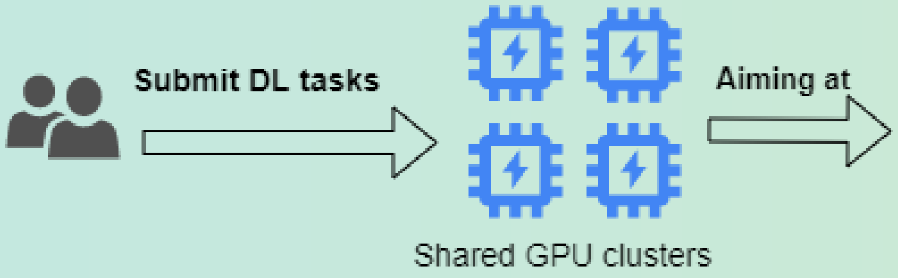
澳門大學
UNIVERSIDADE DE MACAU
UNIVERSITY OF MACAU

中国科学院深圳先进技术研究院
SHENZHEN INSTITUTES OF ADVANCED TECHNOLOGY
CHINESE ACADEMY OF SCIENCES

SC23
Denver, CO | i am hpc.

Deep learning: An important cloud workload

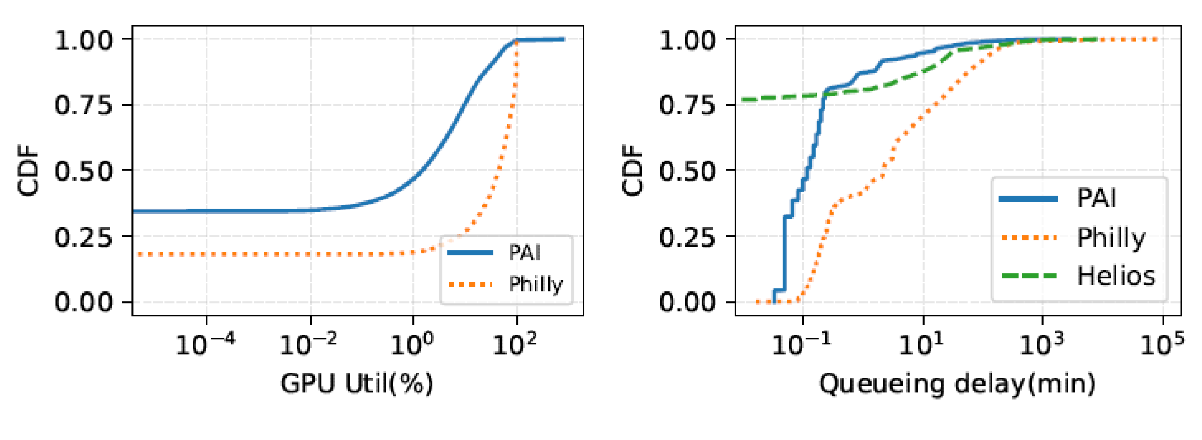
- **Deep learning (DL)** are widely adopted as intelligent applications
  - Computer Vision
  - Natural Language Processing
  - E-commerce Recommendation
  - …
- DL tasks are often trained in GPU clusters to achieve specific validations



DL task	validations
VGG16	85% top1 acc
NCF	69% hit rate
Bert	88% F1 score
YOLOv5	84% mAP

GPU Schedulers for Deep Learning Today

- Underutilization and long queueing delay of deep learning
  - Up to 60% GPUs are below 10% utilization of Philly trace in Microsoft[1] and PAI trace in Alibaba[2]
  - The longest delay spans more than 1,000 minutes in Philly trace and Helios trace in SenseTime[3]

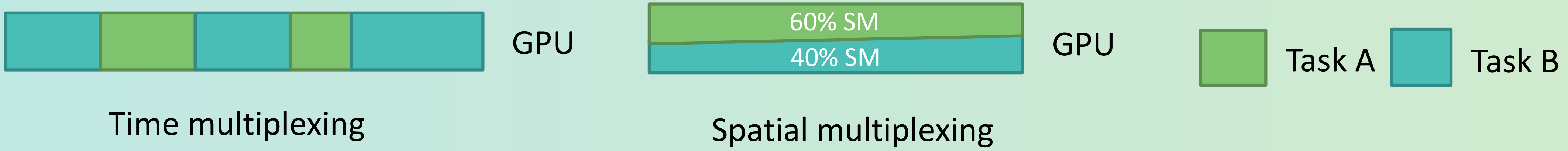


(a) GPU Utilization (b) Queueing delay

[1] Jeon, Myeongjae, et al. "Analysis of Large-Scale Multi-Tenant GPU Clusters for DNN Training Workloads." *In Proceedings of ATC*. 2019.
[2] Weng, Qizhen, et al. "MLaaS in the wild: Workload analysis and scheduling in Large-Scale heterogeneous GPU clusters.“ *In Proceedings of NSDI*, 2022.
[3] Hu, Qinghao, et al. "Characterization and prediction of deep learning workloads in large-scale gpu datacenters." *In Proceedings of SC*, 2021.

GPU Schedulers for Deep Learning Today

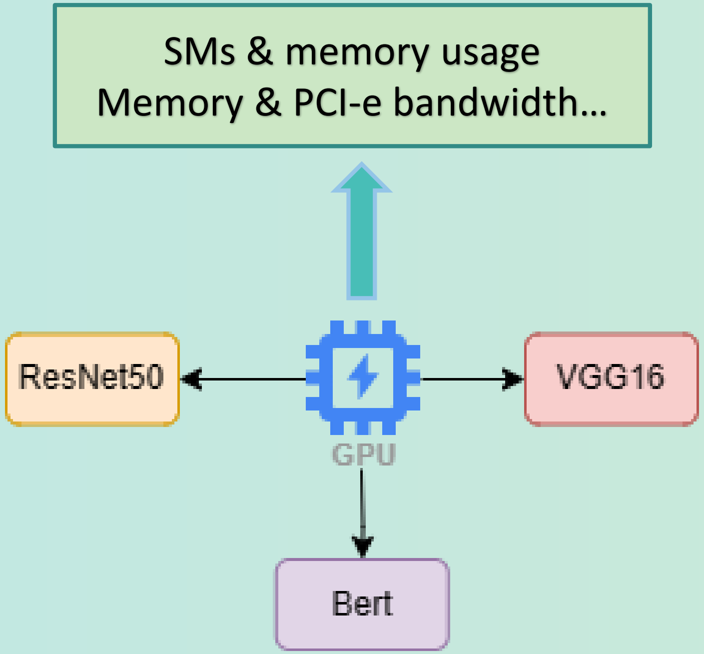
- Packing tasks via time or spatial multiplexing to improve GPU utilization

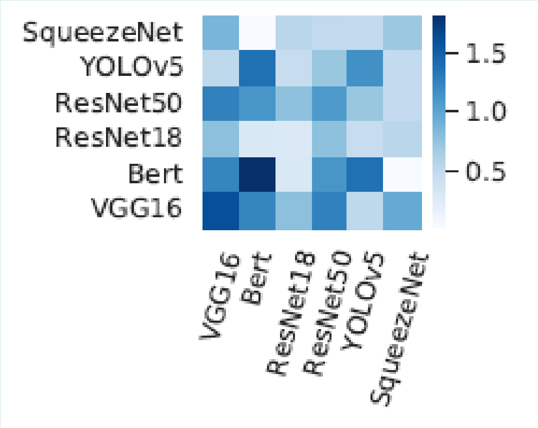


- Time multiplexing: AntMan (OSDI’20), PilotFish (ATC’22), PipeSwitch (OSDI’20)…
- Spatial multiplexing: MPS and MIG in NVIDIA

Is Always Good for Multiplexing?

- Multiple tasks may compete for the same required resources
- Severe interference among multiplexing tasks

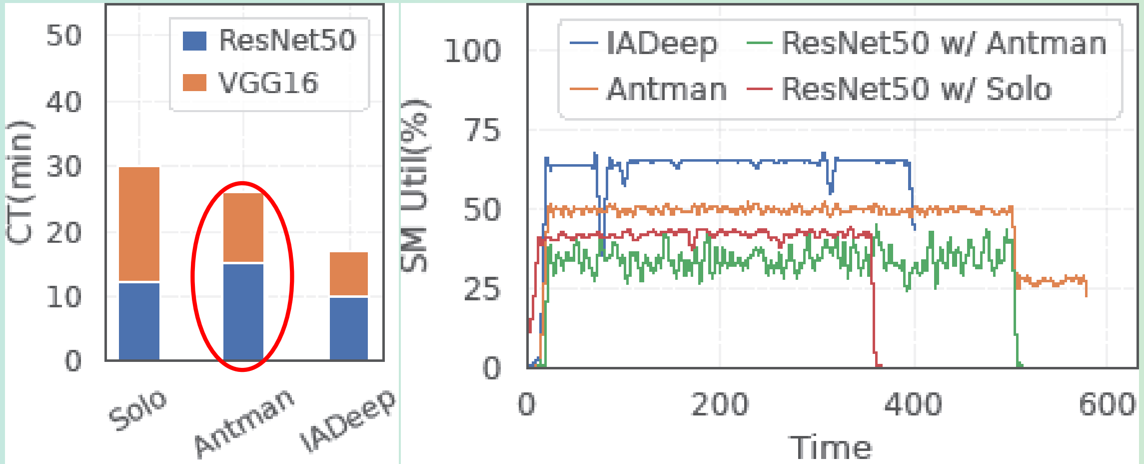




Slowdown of multiplexing tasks

Inefficiencies in Low-level Multiplexing Solutions

- **Kernel-level** solutions: AntMan[1] (OSDI’20)
  - Too fine-grained
  - Can not effectively overlap the kernel computation with copy operations among different tasks
  - Need tailored modifications for different DL frameworks such as Tensorflow and Pytorch



[1] Xiao, Wencong, et al. "AntMan: Dynamic Scaling on GPU Clusters for Deep Learning." *In Proceedings of OSDI*. 2020.

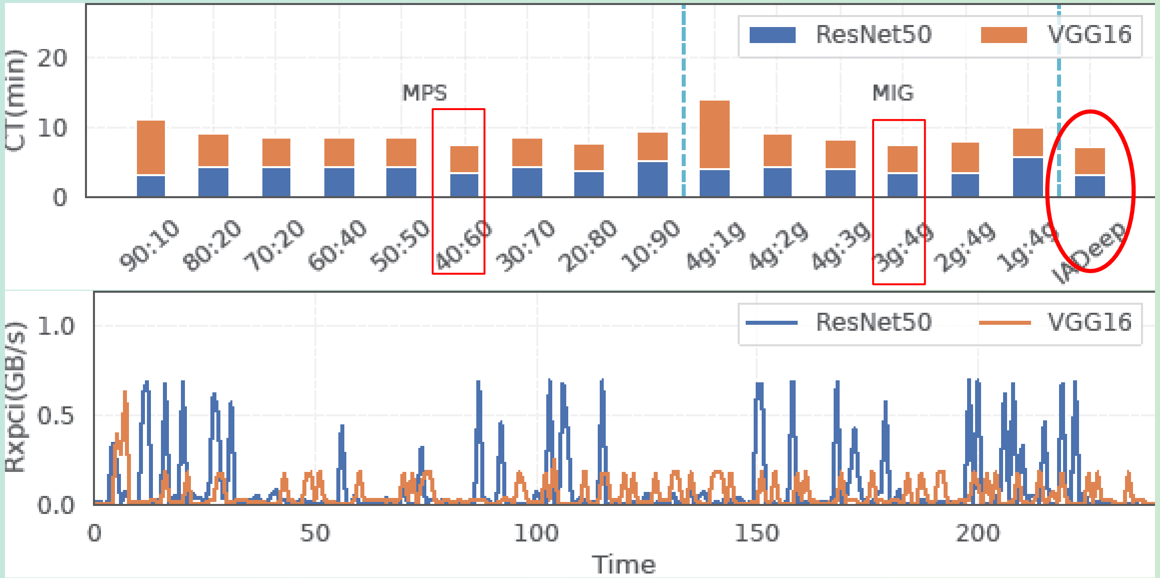
Inefficiencies in Low-level Multiplexing Solutions

- **Hardware-level** isolation solutions: MPS[1] and MIG[2]
  - MPS: A soft isolation solution for GPU multiplexing provided by NVIDIA
    - Require application knowledge to properly set resource partitions
    - Weak fault isolation: when a task fails, other co-located tasks may be affected
  - MIG: A hard isolation solution for GPU multiplexing provided by NVIDIA
    - Only supported by high-end GPU models
    - Inflexible for dynamically allocating resources to tasks

[1] NVIDIA Multi-Process Service. https://docs.nvidia.com/deploy/mps/index.html
[2] NVIDIA Multi-Instance GPU. https://www.nvidia.com/en-us/technologies/multi-instance-gpu

Inefficiencies in Low-level Multiplexing Solutions

- **Hardware-level** isolation solutions: MPS[1] and MIG[2]
  - Too course-grained
  - MPS and MIG can not separate PCI-e bandwidth



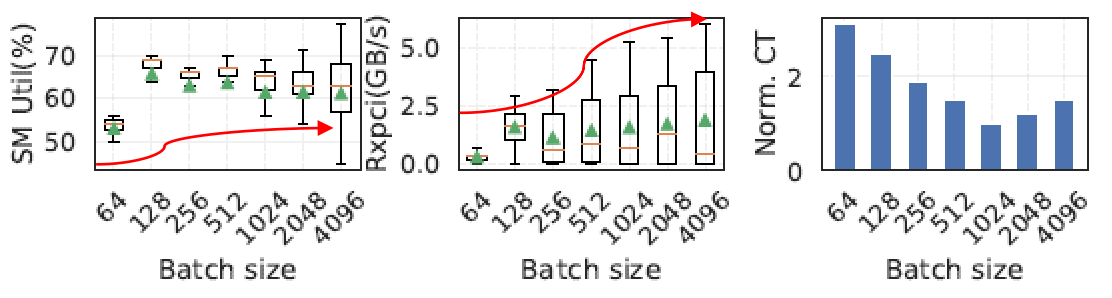
[1] NVIDIA Multi-Process Service. https://docs.nvidia.com/deploy/mps/index.html
[2] NVIDIA Multi-Instance GPU. https://www.nvidia.com/en-us/technologies/multi-instance-gpu

A middleware Solution: IADeep

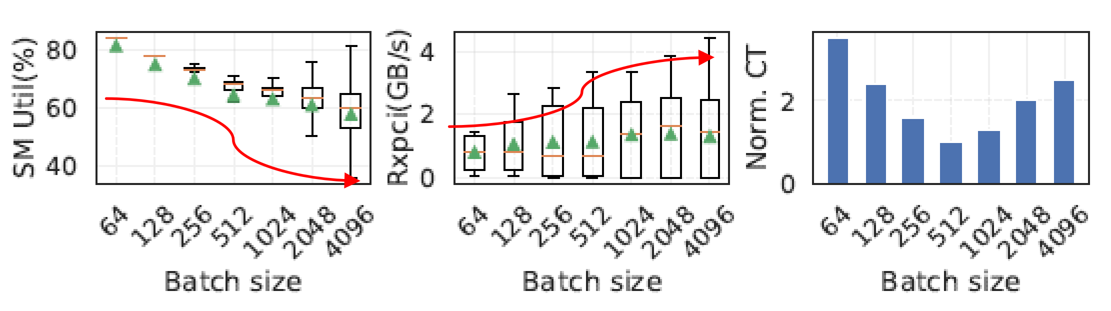
- ✓ Dynamic
- ✓ Less fine-grained
- ✓ Better control all shared resources to mitigate the interference

A middleware Solution: IADeep

- Opportunities and Challenges
  - Training-related configurations, such as *batch size* exhibit strong correlations with various resource metrics



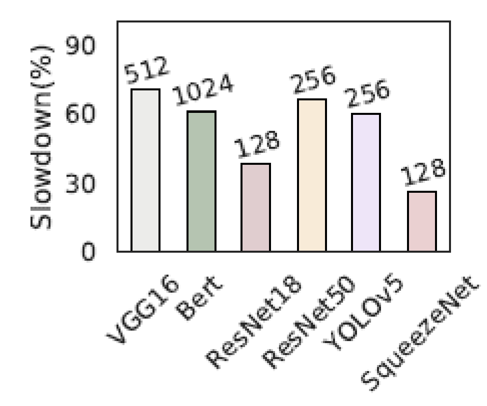
(a) ResNet50



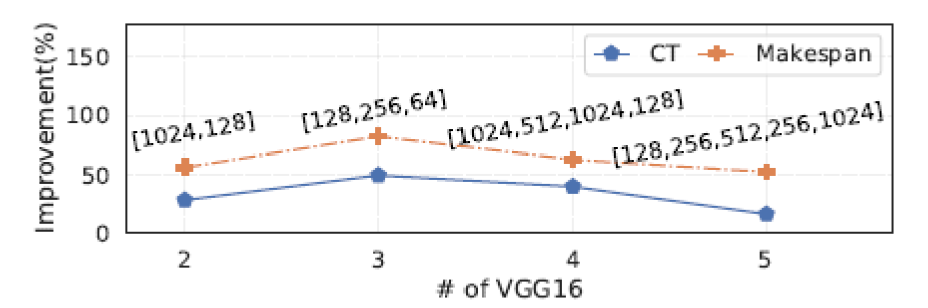
(b) VGG16

A middleware Solution: IADeep

- **Opportunities**
  - Q1: Which tasks to be co-located?
    - Choose appropriate tasks to multiplex on a GPU can mitigate interference
  - Q2: How many tasks should be co-located?
    - Co-locate a suitable number of tasks can balance the waiting time and training time



(a) Slowdown of VGG16



A middleware Solution: IADeep

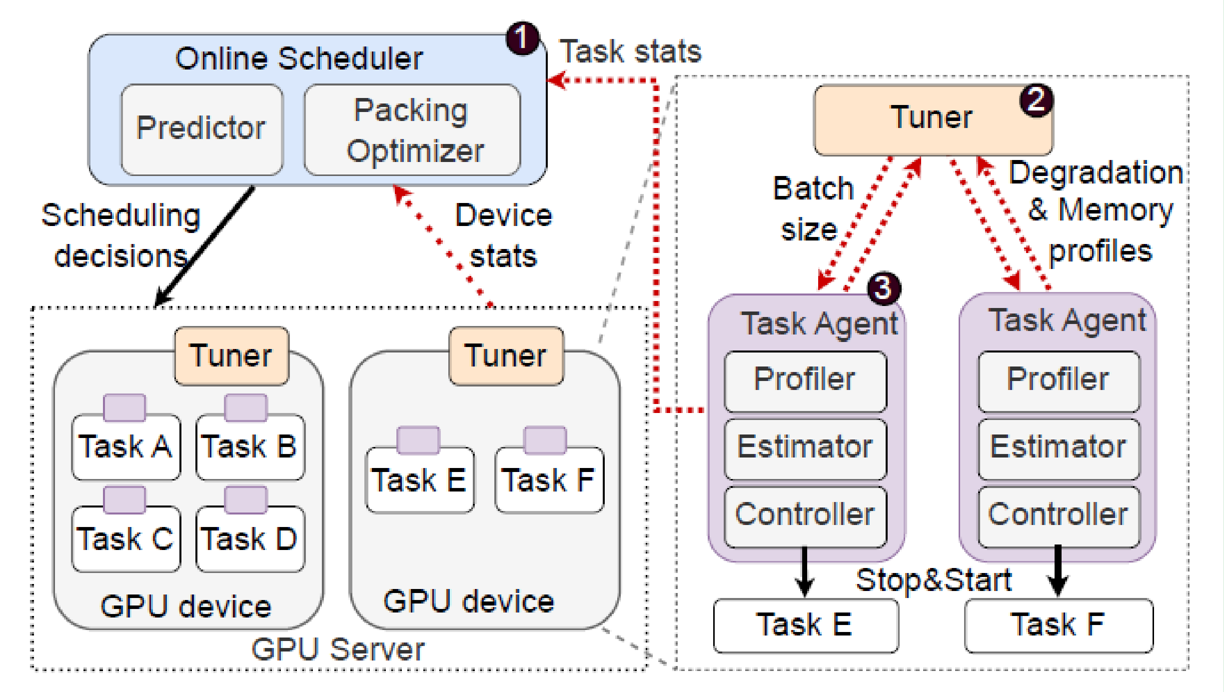
- **Challenges**
  - **C1:** Task configurations heavily influence both interference and training progress
  - **C2:** Vast search space of task configurations
  - **C3:** Intricate coupling between adjusting task configurations and designing task placement policies

IADeep: System Design

① **Online Scheduler**: Find the optimal device to place the new arrival task

② **Tuner**: Tune configurations (batch sizes) to mitigate the interference

③ **Task Agent**: Update the configurations for each co-located task



IADeep: Basic Optimization Problem

- Performance degradation (PD)

Drop in training throughput

$$PD_d^k(*^k) = \boxed{IF_d^k(*^k)} \times \boxed{DT^k(*^k)} \times \boxed{\frac{1}{EFFICIENCT_d^k(*^k)}}$$

Co-location interference

Statistical efficiency

This expression takes interference (IF) and training progress (EFFICIENCY) into PD to address Challenge 1.

IADeep: Basic Optimization Problem

- Co-location interference

$$IF(*) = T^{co}(m) \div T(m)$$

- Drop in throughput

$$DT(*) = m_0/T(m_0) \div \mathrm{m}/\mathrm{T}(\mathrm{m})$$

- Statistical efficiency

Gradient noise scale[1]

$$EFFICIENCY(*) = \frac{\varphi_t + m_0}{\varphi_t + m}$$

$$\boxed{PD(*) = \frac{T^{co}(m)}{T(m_0)} \cdot \frac{m_0}{m} \cdot \frac{\varphi_t + m}{\varphi_t + m_0}}$$

[1] Qiao, Aurick, et al. "Pollux: Co-adaptive Cluster Scheduling for Goodput-Optimized Deep Learning." *In Proceedings of OSDI*, 2021.

IADeep: Basic Optimization Problem

- **Objective**
  - Minimize the overall performance degradation of all co-located tasks on each device
- **Constraint**
  - The memory capacity limitation of each device

x_d^k is a binary variable indicating whether task k is placed on device d

$$\min \sum_{d=1}^{D} \sum_{k=1}^{N} \boxed{x_d^k} \cdot \mathrm{PD}_d^k(\star^{\mathrm{k}})$$

$$\text{s.t.,} \sum_{k=1}^{N} x_d^k \cdot \mathrm{PM}^k(\star^{\mathrm{k}}) \leq \mathrm{C}_d, \forall d,$$

$$\sum_{d=1}^{D} x_d^k = 1, \forall k, \text{ and } x_d^k \in \{0, 1\}.$$

IADeep: Online Task Placement

- **Online Prediction**
  - Use an alternative solution that predicts co-location performance based only on co-location patterns
  - Collect training samples online and train the prediction model incrementally
- **Task Placement**
  - Find the device with minimal performance degradation to place the task (**Q1**)

IADeep: Task Configuration Selection

- Finding task configurations
  - Bayesian Optimization (GP-LCB)

$$\min_{\Delta \in \Re} \mu(\Delta) - \beta_n^{1/2} \sqrt{k(\Delta, \Delta)}, \text{ s.t., } \sum_{k \in \mathcal{T}} \mathrm{PM}^k(\star^k) \leq \mathrm{C}_d$$

    - Able to handle noise
    - Efficient to find an optimal configuration from a vast search space (Challenge 2)
- Profiling memory usage
  - Fit functions of batch size and memory usage to avoid OOM (memory limitation constraint)
  - Use cubic polynomial regression with an average testing error 0.06

IADeep: Co-location Optimization

- Optimization 1
  - Regulate the average resource utilization of both SM and memory on each device
- Optimization 2
  - Evaluate the performance gain of co-location (**Q2**)
  - Find a tradeoff between *waiting time* and *interference mitigation* of assigning a task

$$\boxed{E_i} = (1 + \boxed{\epsilon_i}) \cdot \sum_{l=1}^{n} \frac{\epsilon_i}{1 + \epsilon_l} - 1, \quad 1 \le i \le n - 1$$

If positive, schedule the task n, the overall CT will be worse.

The change of PD for each co-located task i.

IADeep Implementation

- **Online Scheduler** *(5500 LOC in Go)*
  - On top of *Kubernetes*
  - Develop a *device plugin* to expose device status
  - Use *etcd* to store intermediate results
- **Tuner** *(300 LOC in Python)*
  - On each GPU device
  - Provides well-tuned *batch sizes* to each task agent
- **Task Agent** *(360 LOC in Python)*
  - On each DL task
  - Collect GPU memory and task runtime information and upload them to *ectd*

Evaluation: Experiment setup

- Testbed
  - A GPU cluster of 20 RTX 3090 GPUs managed by Kubernetes 1.18.13
  - CUDA 11.4 & CUDNN 7 & NVIDIA Driver 470.57
- Workloads
  - With job arrival process follows Microsoft trace

Task Name	Dataset	Validation	m_0 (batch size)	Optimizer	Size	Frac. Tasks	Filed
VGG16 [60]	CIFAR10 [42]	85% top1 acc.	512	Adam	S	14%	
SqueezeNet [35]	CIFAR10	50% top1 acc.	512	Adam	S	14%	Image Classification
ResNet50 [32]	CIFAR100 [42]	75% top1 acc.	1024	Adam	S	14%	
NCF [33]	MovieLens [31]	69% hit rate	1024	SGD	M	12%	Recommend System
AD-GCL [62]	REDDIT-MULTI-12K[3]	40% acc.	32	Adam	M	12%	Social Network
LSTM [55]	Wikitext-2 [51]	4.0 PPL	256	Adadelta	M	12%	Text Generation
Bert(finetune) [22]	SQuAD [57]	88% F1 score	8	AdamW	L	10%	Question Answering
YOLOv5 [39]	COCO [46]	84% mAP	32	SGD	L	10%	Object Detection
ResNet18 [32]	ImageNet [21]	75% top1 acc.	128	SGD	XL	2%	Image Classification

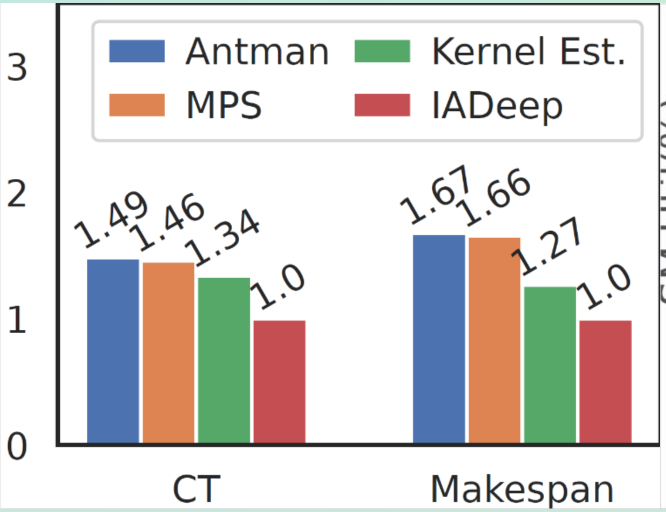
Evaluation: Experiment setup

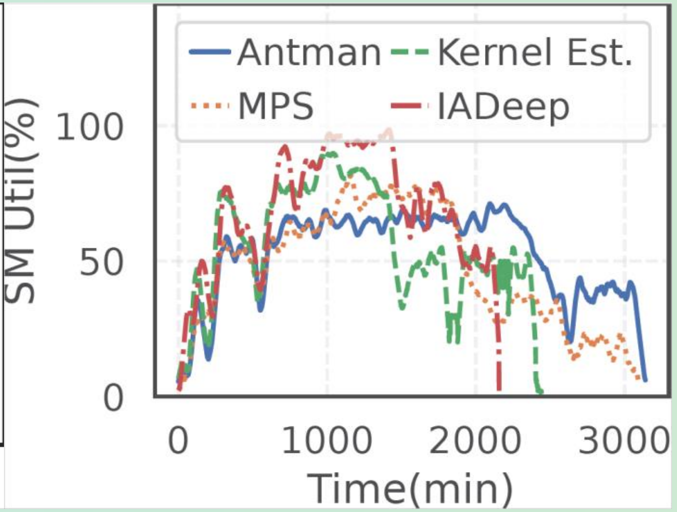
- **Baselines**
  - AntMan[1] , Tencent’s solution based on MPS[2] , Kernel Est.[3]
- **Evaluation metrics**
  - CT: Average Complete Time of the overall tasks
  - Makespan: The total time it takes to complete all tasks
  - GPU resource utilization: SM utilization and memory utilization

[1] Xiao, Wencong, et al. "AntMan: Dynamic Scaling on GPU Clusters for Deep Learning." In Proceedings of OSDI. 2020.
[2] NVIDIA Multi-Process Service. https://docs.nvidia.com/deploy/mps/index.html
[3] Xu, Xin, et al. "Characterization and Prediction of Performance Interference on Mediated Passthrough GPUs for Interference-aware Scheduler." *In Proceedings of HotCloud*. 2019.

Evaluation: End-to-end performance

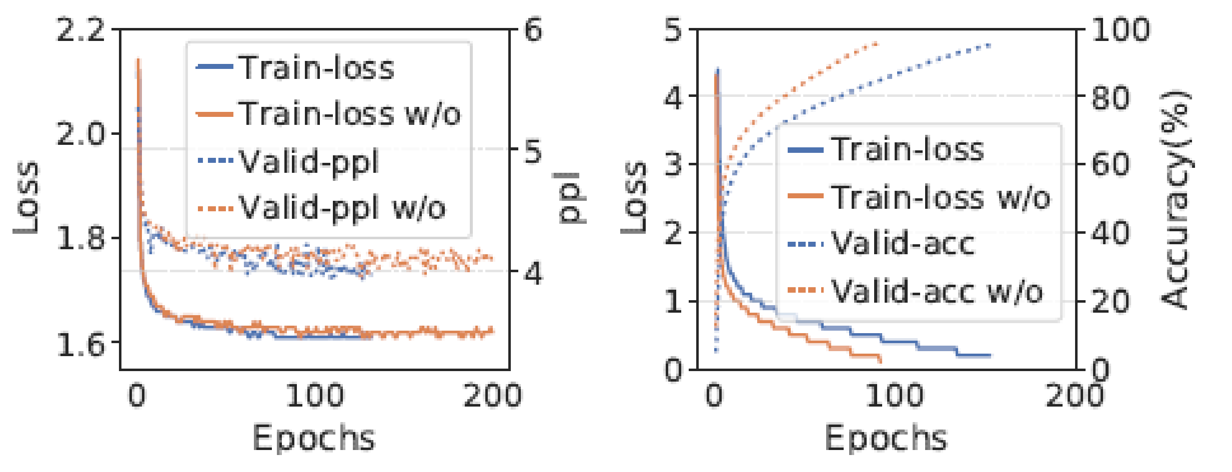
- A task stream contains 300 DL training tasks
  - Up to 49% **CT reduction** compared to baselines
  - Up to 67% **makespan reduction** compared to baselines
  - 31% **GPU utilization improvement**





Evaluation: End-to-end performance

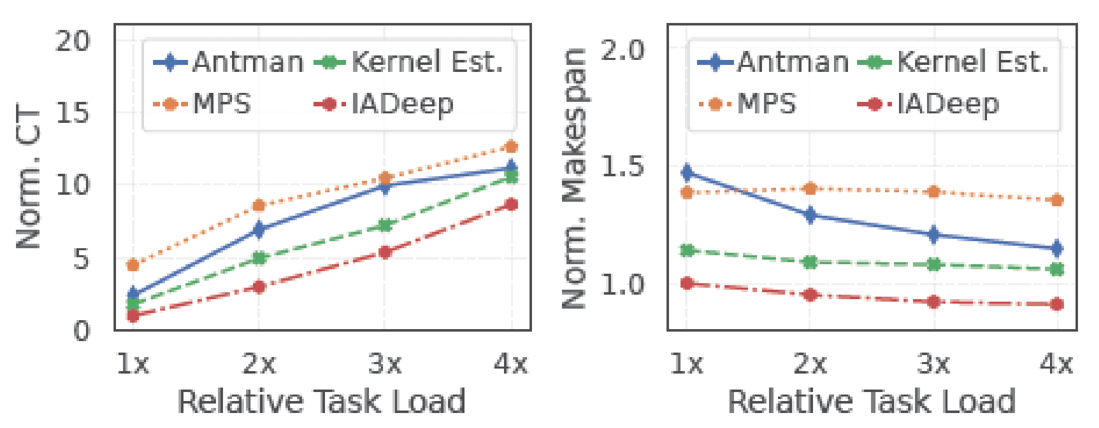
- A task stream contains 300 DL training tasks
  - **Convergence**
    - The training tasks can converge at several epochs to achieve the target validations



(a) LSTM (b) ResNet18

Evaluation: End-to-end performance

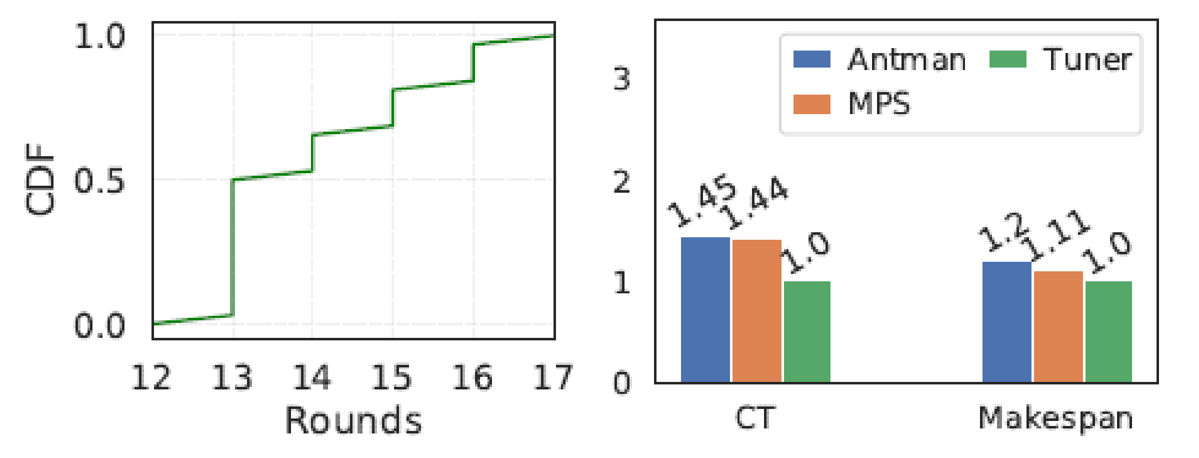
- A task stream contains 300 DL training tasks
  - **Sensitivity to task arrival rate**
    - IADeep always outperforms other baselines concerning the overall CT
    - For makespan, IADeep achieves a linear speedup with the increase of task arrival rate



(a) CT (b) Makespan

Evaluation: Effectiveness of Interference Mitigation

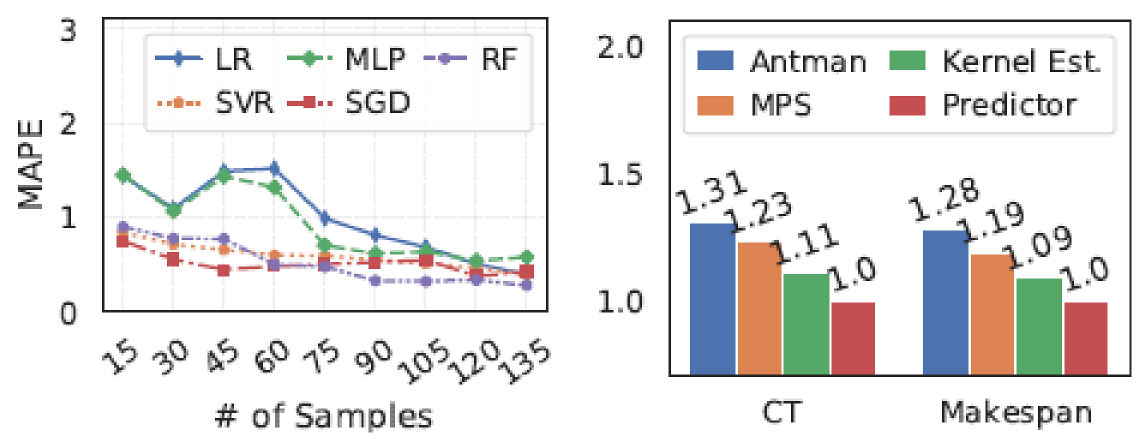
- A task stream contains 300 DL training tasks
  - GP-LCB (in Tuner) converges within 17 rounds and all search cost is within 2s
  - Tuner alone improves CT and makespan by up to 45% and 20%



(a) CDF of search rounds (b) Performance improvement

Evaluation: Effectiveness of Task Assignment

- A task stream contains 300 DL training tasks
  - RF as the Predictor achieves only 27.8% MAPE within 135 samples
  - Online Predictor alone achieves up to 31% CT and 28% makespan reduction



(a) Prediction error (b) Performance improvement

More Evaluations in our Paper

- Performance of Online Optimizer
  - Optimizer alone performance improvement
  - Scheduling overhead

Conclusion

Interference-aware Multiplexing for Deep Learning in GPU Clusters

- Propose a formulation that quantifies co-location performance by combining interference and job training progress
- Co-optimize cluster-level job assignment and per device interference control
- Achieve up to 49% in CT and up to 31% in GPU resource utilization

Thanks & QA

yc17498@um.edu.mo