KI Service Zentrum
by Hasso-Plattner-Institut

HPI

SPONSORED BY THE

Federal Ministry of Education and Research


Draco: Dynamic Resource Allocation for Concurrent ML Applications

Theo Rudig
Hasso-Plattner-Institute
Internet Technologies and Softwarization
Potsdam, Germany
theo.rudig@hpi.de

Holger Karl
Hasso-Plattner-Institute
Internet Technologies and Softwarization
Potsdam, Germany
holger.karl@hpi.de

HPI Hasso Plattner Institut
Digital Engineering · Universität Potsdam

Draco: Dynamic Resource Allocation for Concurrent ML Applications

International Symposium on Cluster, Cloud and Internet Computing 2025 | Presentation

Theo Radig and Holger Karl
Internet Technologies and Softwarization
Hasso-Plattner-Institute

Challenges & Goals

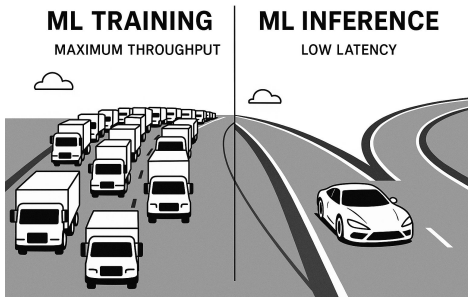
System Design

Summary

Requirements & Contributions

Evaluation

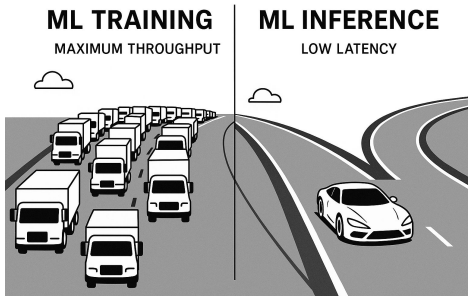
Challenges



Created by ChatGPT.

- ML workloads are **heterogenous**
  - Batch training
  - Low-latency inference

Challenges

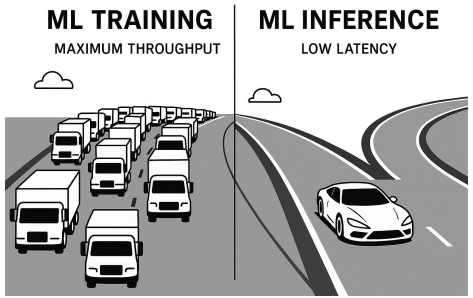


Created by ChatGPT.

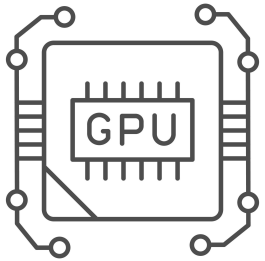
- ML workloads are **heterogenous**
  - Batch training
  - Low-latency inference

- Inference services **guarantee latency**

Challenges



Created by ChatGPT.



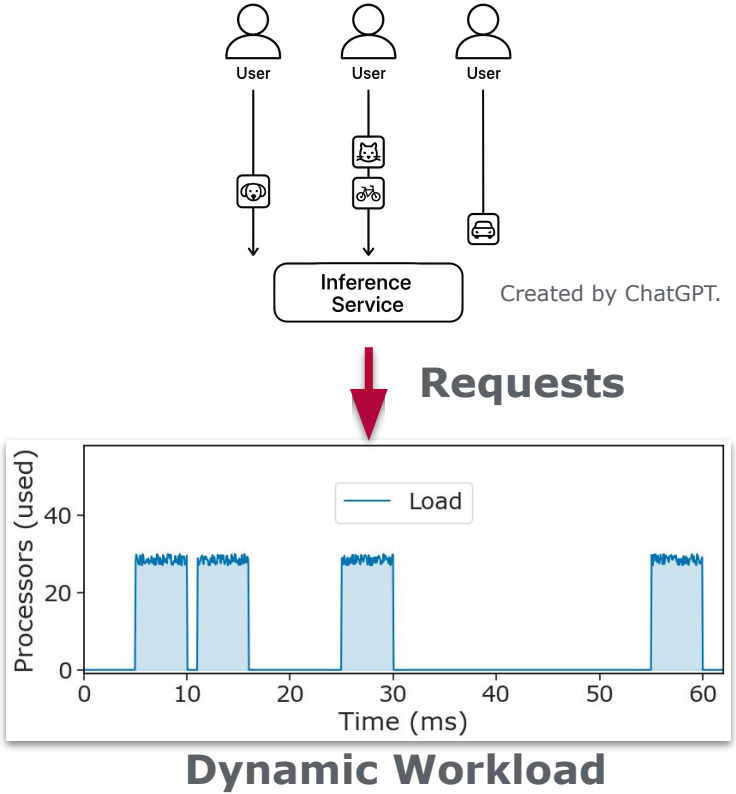
- ML workloads are **heterogenous**
  - Batch training
  - Low-latency inference

- Inference services **guarantee latency**

- **Exclusive** GPU usage
- Spatial sharing **over-provisions**

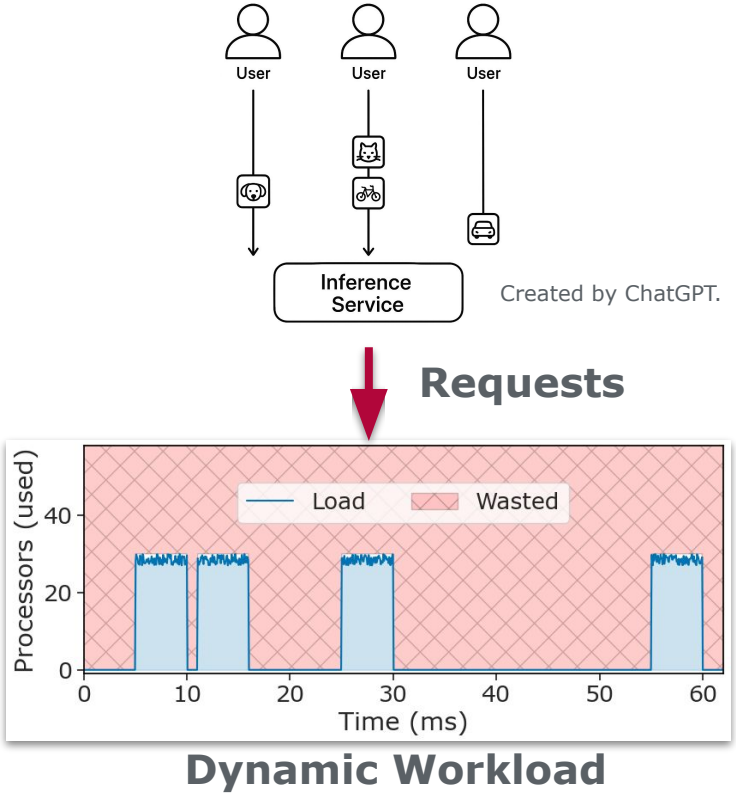
Example

- Requests arrive irregularly



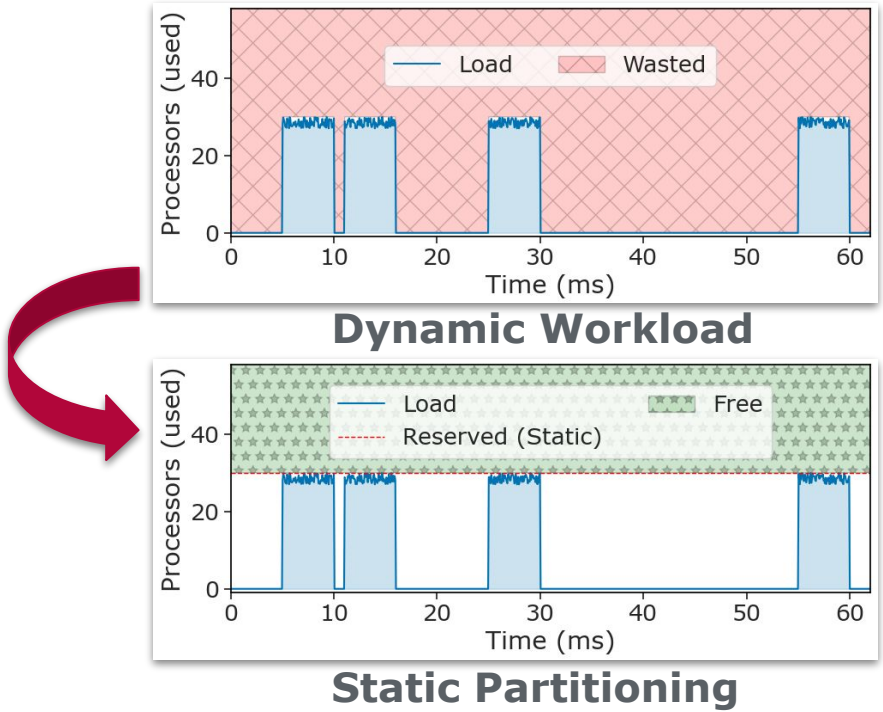
Example

- Requests arrive irregularly
- Compute resources idle



Example

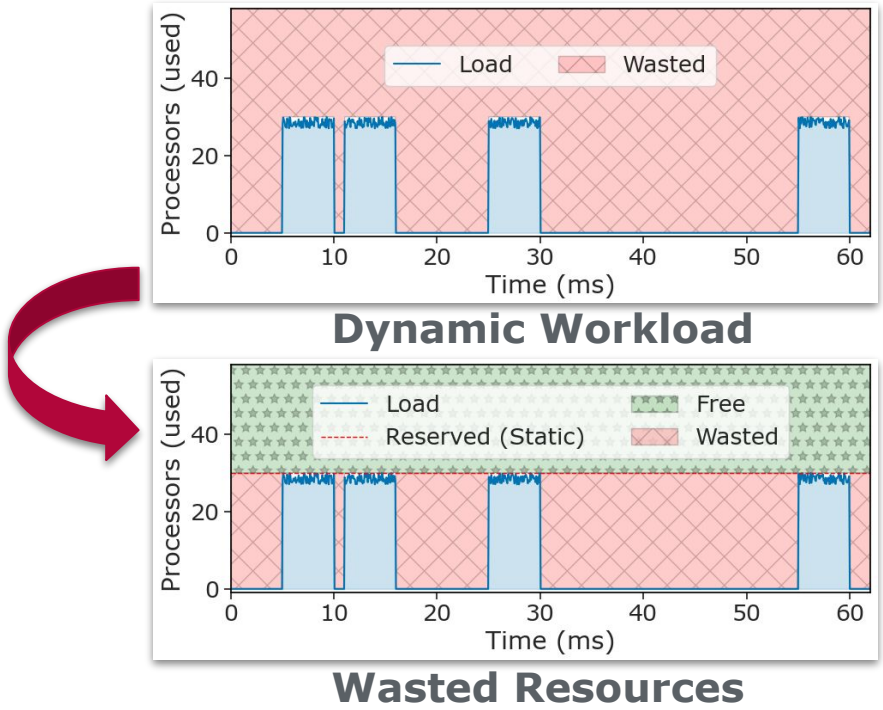
- Requests arrive irregularly
- Compute resources idle
- Spatial sharing (MIG[1] / MPS[2])



[1] NVIDIA Multi-Process Service [2] NVIDIA Multi-Instance GPU

Example

- Requests arrive irregularly
- Compute resources idle
- Spatial sharing (MIG[1] / MPS[2])
- Inference service still over-provisioned
- Wasted resources

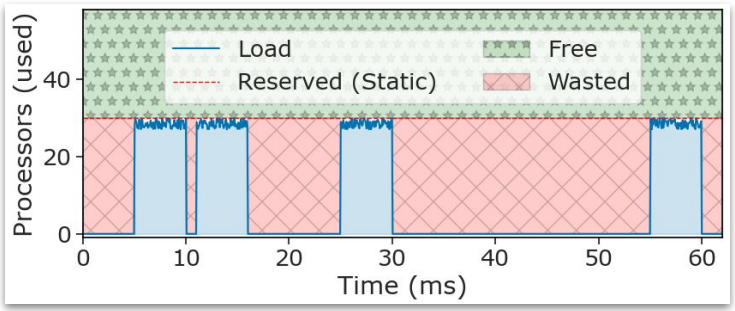


[1] NVIDIA Multi-Process Service [2] NVIDIA Multi-Instance GPU

Constraints & Goals

Constraints (MPS / MIG)

- ❏ Static partitions ensure low latency
- ❏ Cannot be quickly reconfigured
- ❏ Static partitions over-provision
  - ➢ Cost-ineffective



Wasted Resources

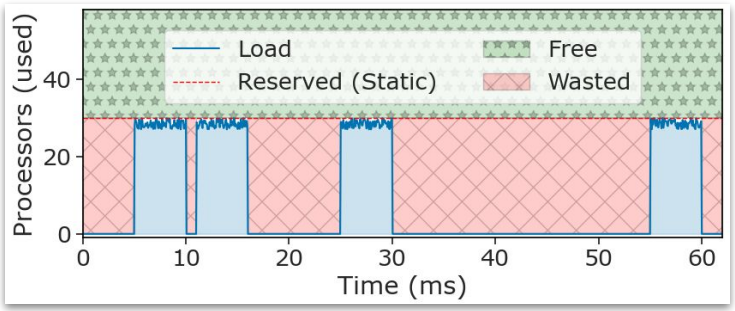
Constraints & Goals

Constraints (MPS / MIG)

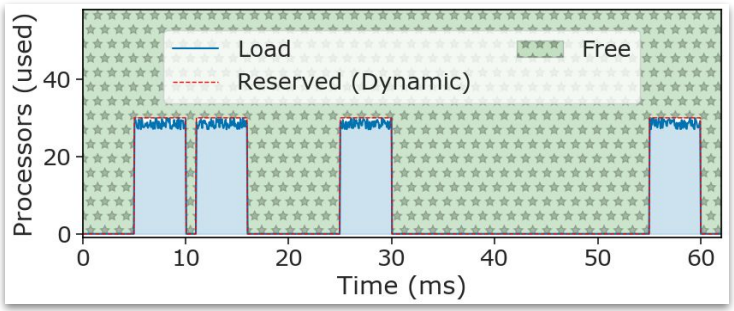
- ❏ Static partitions ensure low latency
- ❏ Cannot be quickly reconfigured
- ❏ Static partitions over-provision
  - ➢ Cost-ineffective

Goals

- ❏ Dynamically allocate / free resources
- ❏ Co-locate inference and training
  - ➢ Low inference latency
  - ➢ High training throughput



Wasted Resources



Dynamic Partitioning

Challenges & Goals

System Design

Summary

Requirements & Contributions

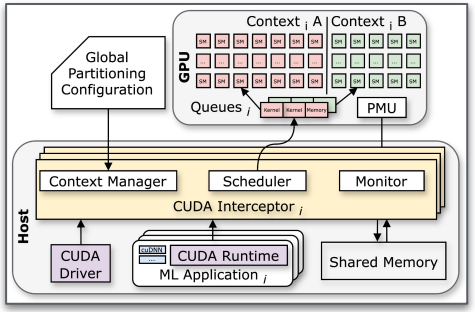
Evaluation

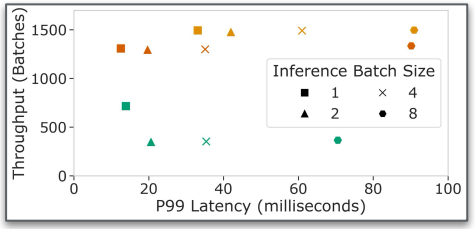
Requirements

- **Applicable**
  - Zero modifications to ML frameworks
- **Online**
  - No offline profiling phase
- **Isolation**
  - Co-located applications operate independently
- **Low overhead**
  - Profiling must not increase latency
  - Moderate memory footprint

Contributions

- **Explore NVIDIA Green Contexts**
  - For dynamic SM[1] partitioning
- **Draco**
  - Automatically assign resources
  - Fine-granular partitioning
  - Live sampling the PMU[2]
- **Evaluate**
  - Compare with state-of-the-art MPS
  - Model different workloads





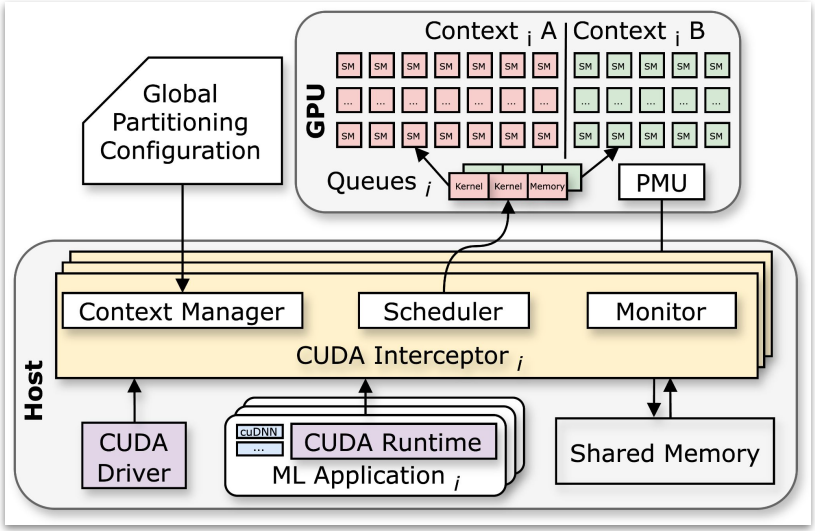
[1] Streaming-Multiprocessor [2] Performance Monitoring Unit

System Design

Architecture

Highlights

- ❏ Effortlessly integrate with ld_preload
- ❏ Intercept CUDA runtime / driver
- ❏ Separate processes (CPU) / contexts (GPU)
- ❏ Low overhead profiling at runtime



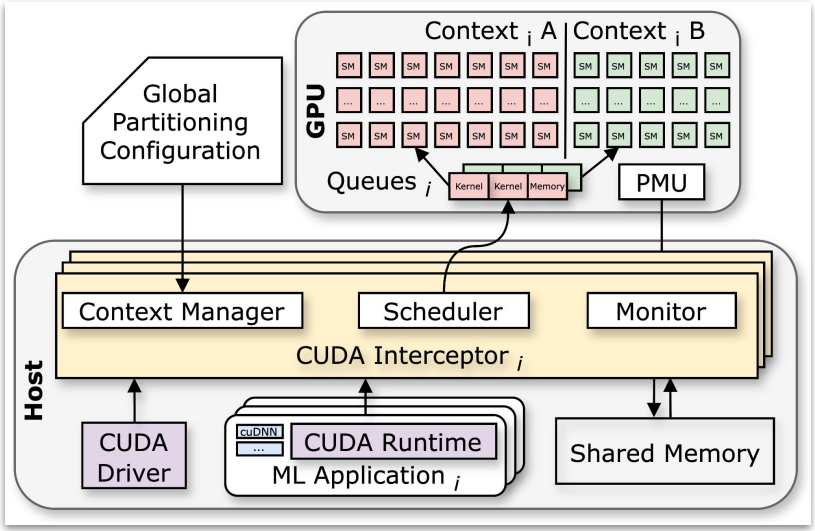
Architecture

Highlights

- Effortlessly integrate with ld_preload
- Intercept CUDA runtime / driver
- Separate processes (CPU) / contexts (GPU)
- Low overhead profiling at runtime

Core Components

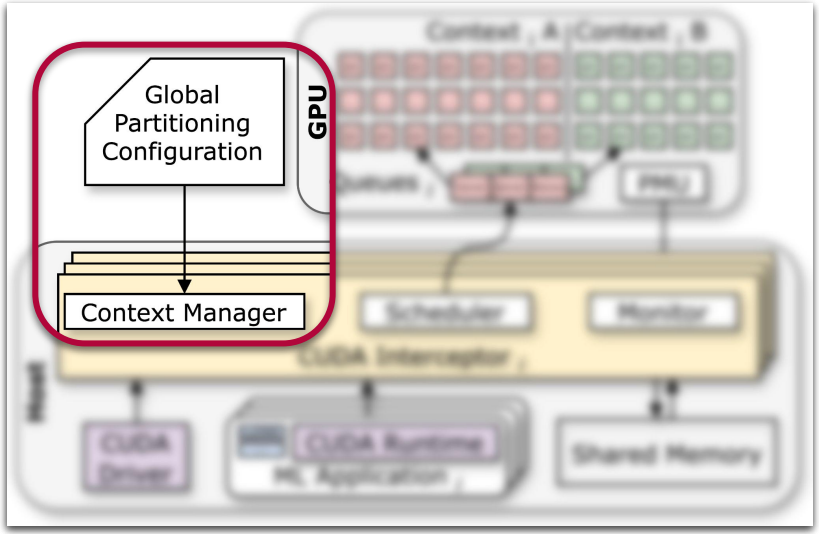
- Context Manager
- Monitor
- Kernel Scheduler



Context Manager

Traditional Context

- Comparable to OS process
- Isolates resources
- Automatically created
- Large memory footprint (**429**MB)



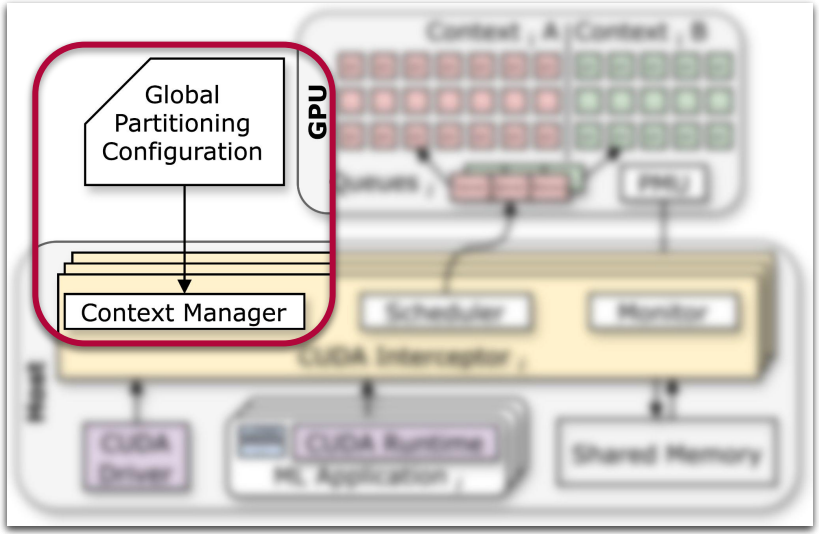
Context Manager

Traditional Context

- ❏ Comparable to OS process
- ❏ Isolates resources
- ❏ Automatically created
- ❏ Large memory footprint (**429**MB)

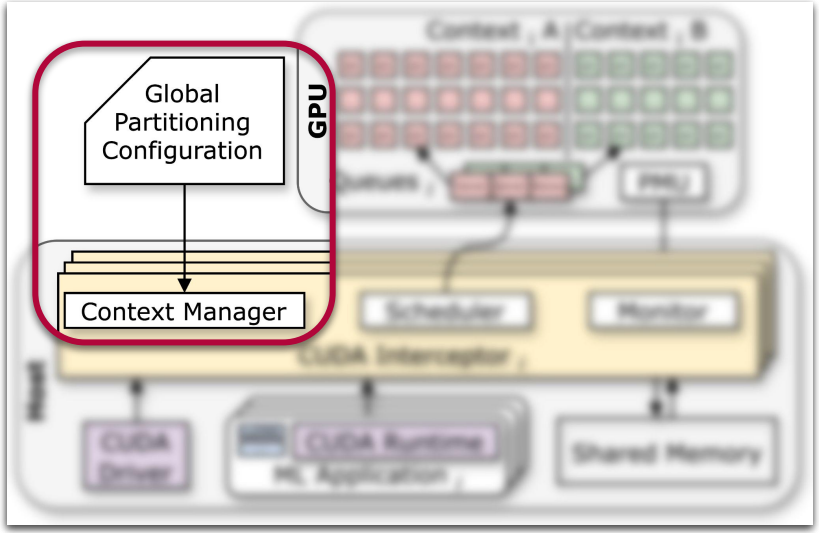
Green Context

- ❏ Manual created
- ❏ Low memory footprint (**4**MB)



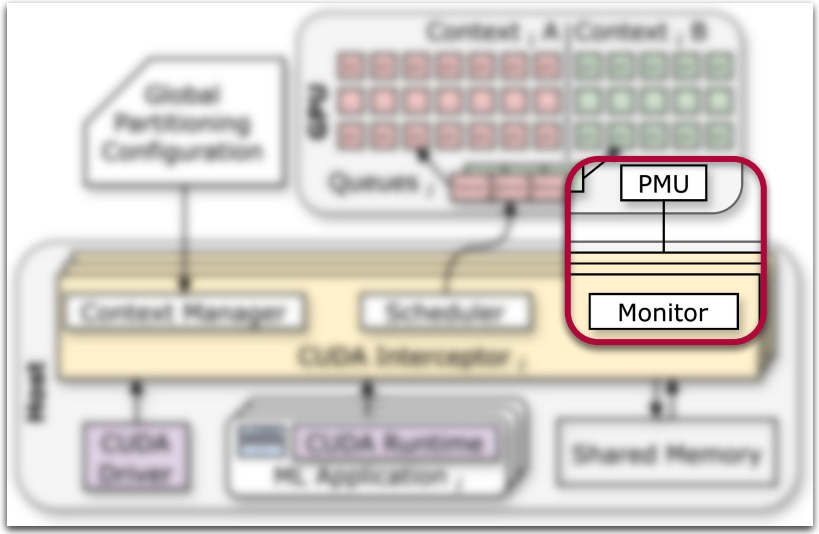
Context Manager

- Self-manage a pool of contexts
- Applications replicate this pool
- Fine-granular partitioning granularity



Monitor

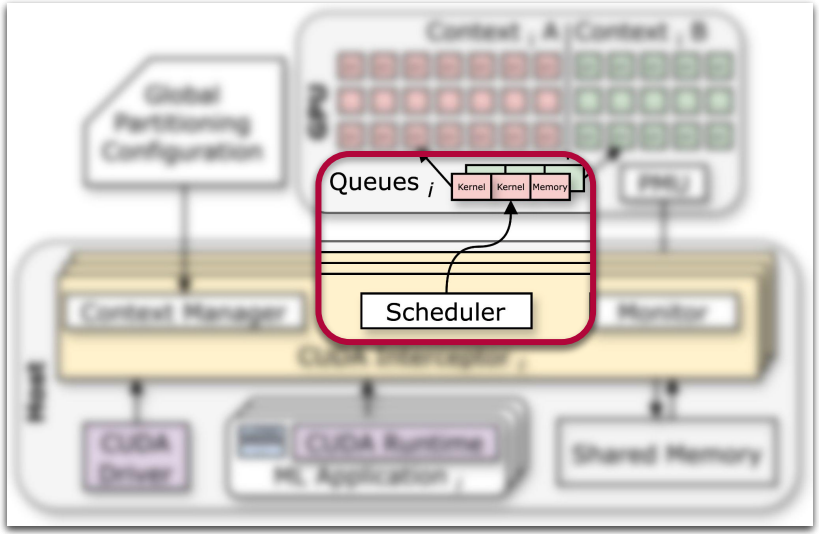
- Hardware-counter sampling every 1ms
- Percentage of active SMs
- Estimate SM requirement for inference
- Detect inference idle times



Kernel Scheduler

Kernel

- Function that executes on a GPU
- Data-parallel operation
- Invoked through CUDA APIs



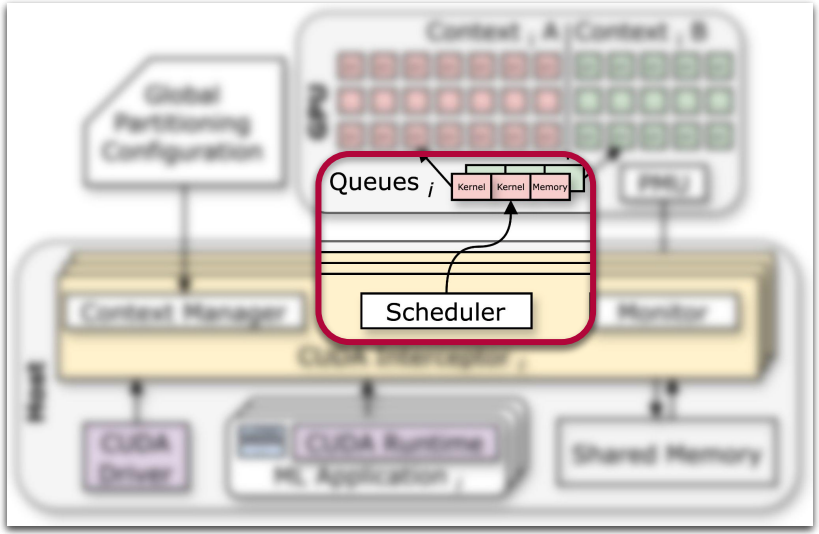
Kernel Scheduler

Kernel

- Function that executes on a GPU
- Data-parallel operation
- Invoked through CUDA APIs

Scheduler

- For every kernel
  - Delay submission
  - Initiate context switch



Challenges & Goals

Requirements & Contributions

System Design

Evaluation

Summary

Testbed & Workloads

Testbed

- NVIDIA A30 GPU (**56** SMs)
- Intel Xeon Platinum 8468V (**96** Cores)
- Latest CUDA version

Testbed & Workloads

Testbed

- NVIDIA A30 GPU (**56** SMs)
- Intel Xeon Platinum 8468V (**96** Cores)
- Latest CUDA version

Workloads

Workload	Model	Operation	Batch Size	MPS Thread Limit (%)
A				
B				
C				

Testbed & Workloads

Testbed

- NVIDIA A30 GPU (**56** SMs)
- Intel Xeon Platinum 8468V (**96** Cores)
- Latest CUDA version

Workloads

Workload	Model	Operation	Batch Size	MPS Thread Limit (%)
A	Bert-base-uncased	Training	16	
B	Bert-base-uncased	Training	16	
C	Bert-base-uncased	Training	16	

Testbed & Workloads

Testbed

- NVIDIA A30 GPU (**56** SMs)
- Intel Xeon Platinum 8468V (**96** Cores)
- Latest CUDA version

Workloads

Workload	Model	Operation	Batch Size	MPS Thread Limit (%)
A	ResNet101 Bert-base-uncased	Inference Training	1, 2, 4, 8 16	
B	Distil-bert Bert-base-uncased	Inference Training	1, 2, 4, 8 16	
C	Bert-base-uncased Bert-base-uncased	Inference Training	1, 2, 4, 8 16	

Testbed & Workloads

Testbed

- ❏ NVIDIA A30 GPU (**56** SMs)
- ❏ Intel Xeon Platinum 8468V (**96** Cores)
- ❏ Latest CUDA version

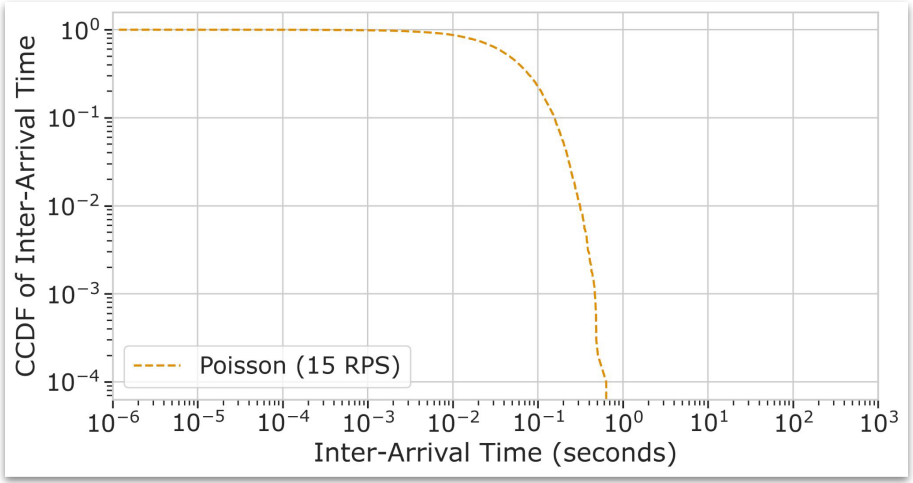
Workloads

Workload	Model	Operation	Batch Size	MPS Thread Limit (%)
A	[illegible]	[illegible]	[illegible]	[illegible]
	[illegible]	Training	16	[illegible]
B	Distil-bert	Inference	1, 2, 4, 8	60, 80, 80, 90
	Bert-base-uncased	Training	16	40, 20, 20, 10
C	[illegible]	[illegible]	[illegible]	[illegible]
	[illegible]	Training	16	[illegible]

Inter-Arrival Times

Poisson

- ❏ Exponentially distributed
- ❏ No long gaps
- ❏ Low variability and burstiness

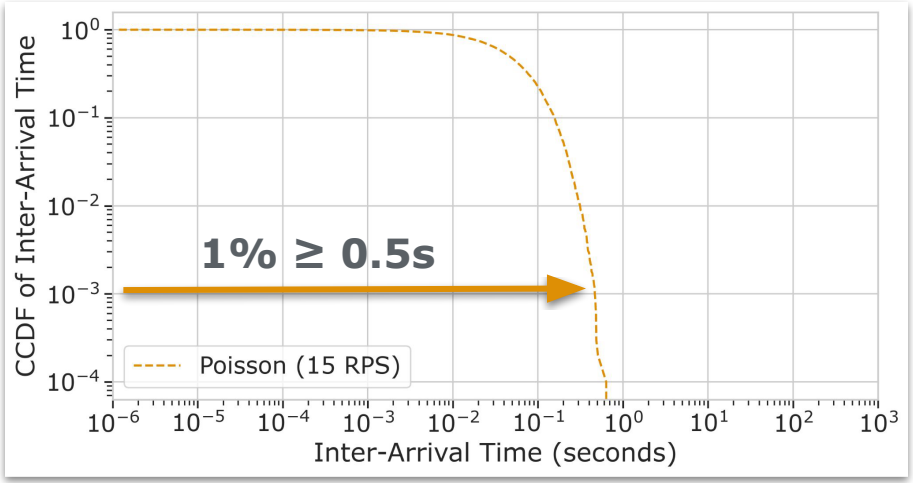


Inter-arrival Times

Inter-Arrival Times

Poisson

- ❏ Exponentially distributed
- ❏ No long gaps
- ❏ Low variability and burstiness



Inter-arrival Times

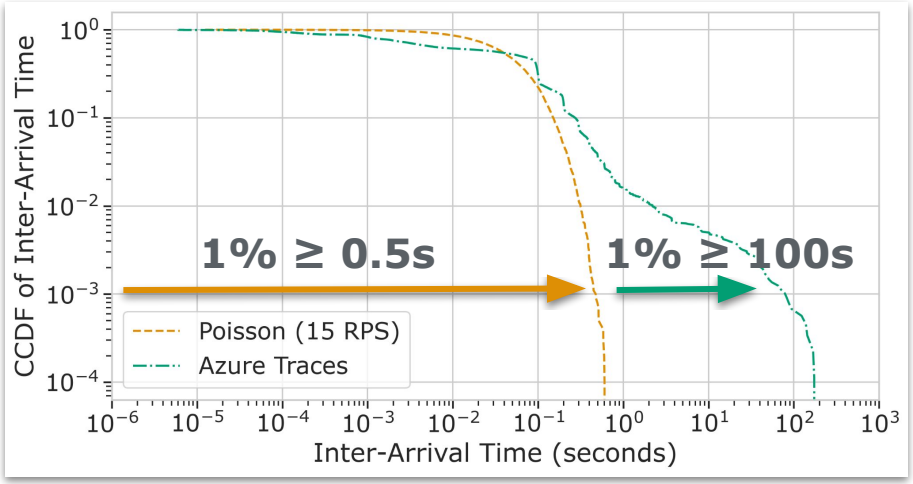
Inter-Arrival Times

Poisson

- ❏ Exponentially distributed
- ❏ No long gaps
- ❏ Low variability and burstiness

Real-world Traces

- ❏ Same median inter-arrival times
- ❏ Rare long gaps (≥ **100s**)

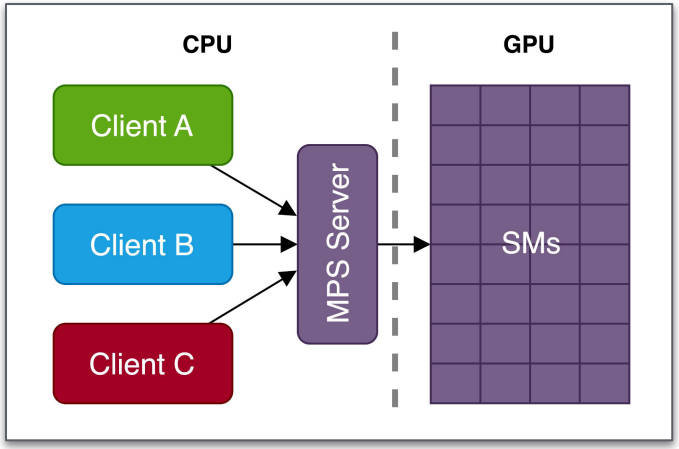


Inter-arrival Times

Systems under Test

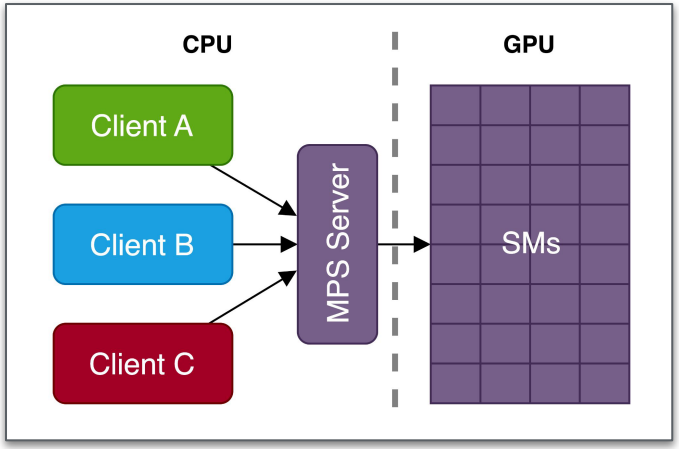
	Transparent	Dynamic	Autonomous

Systems under Test



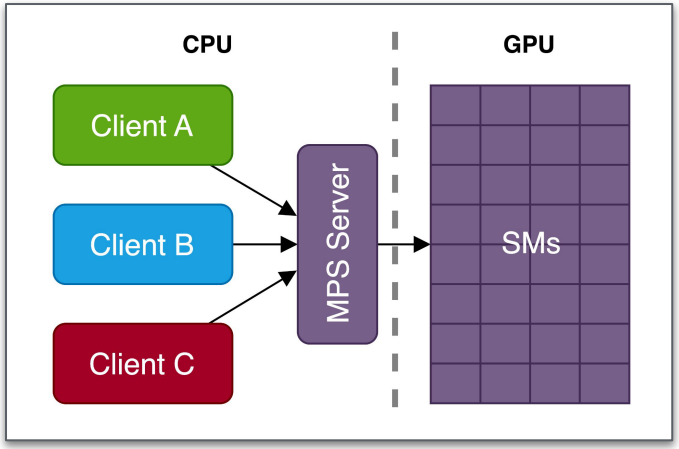
	Transparent	Dynamic	Autonomous
MPS **U**nlimited			

Systems under Test



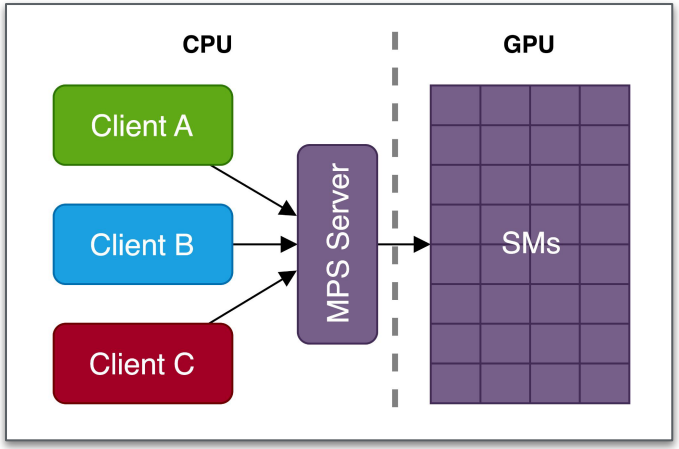
	Transparent	Dynamic	Autonomous
MPS **U**nlimited	X		

Systems under Test



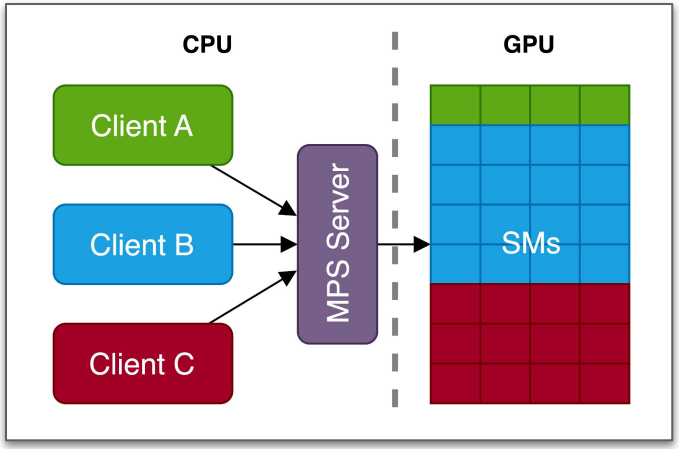
	Transparent	Dynamic	Autonomous
MPS **U**nlimited	✗	✔	

Systems under Test



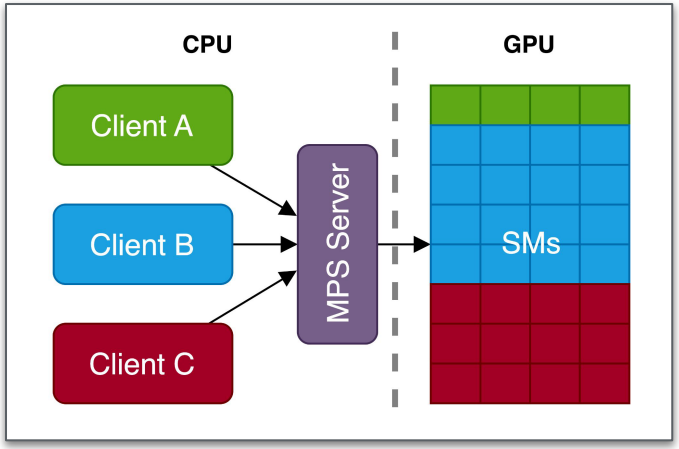
	Transparent	Dynamic	Autonomous
MPS **U**nlimited	✗	✔	✔

Systems under Test



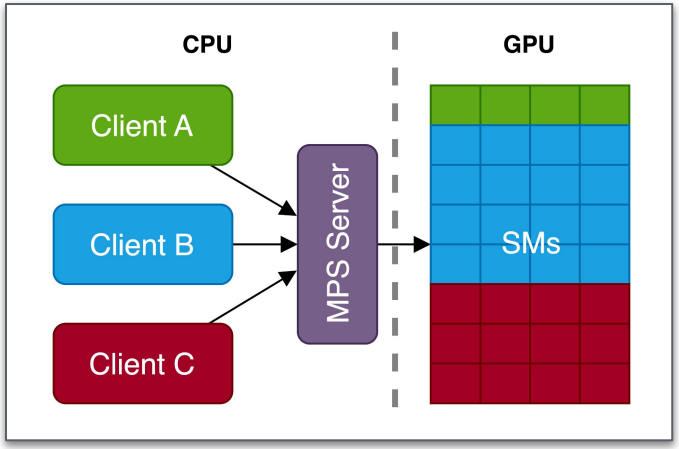
	Transparent	Dynamic	Autonomous
MPS **U**nlimited	✗	✔	✔
MPS **L**imited			

Systems under Test



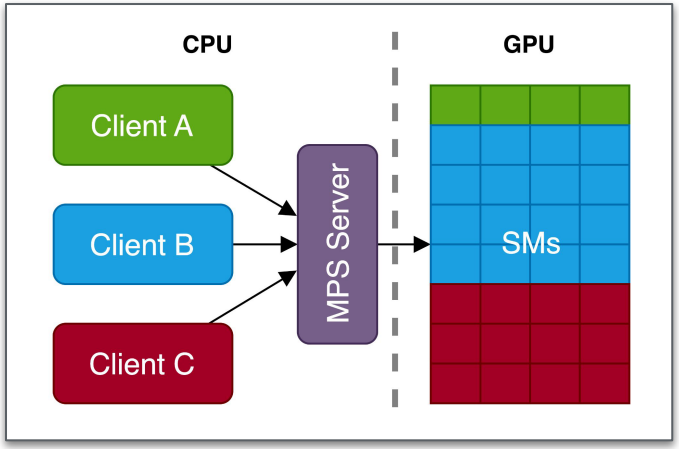
	Transparent	Dynamic	Autonomous
MPS **U**nlimited	✗	✔	✔
MPS **L**imited	✗		

Systems under Test



	Transparent	Dynamic	Autonomous
MPS **U**nlimited	✗	✔	✔
MPS **L**imited	✗	✗	

Systems under Test

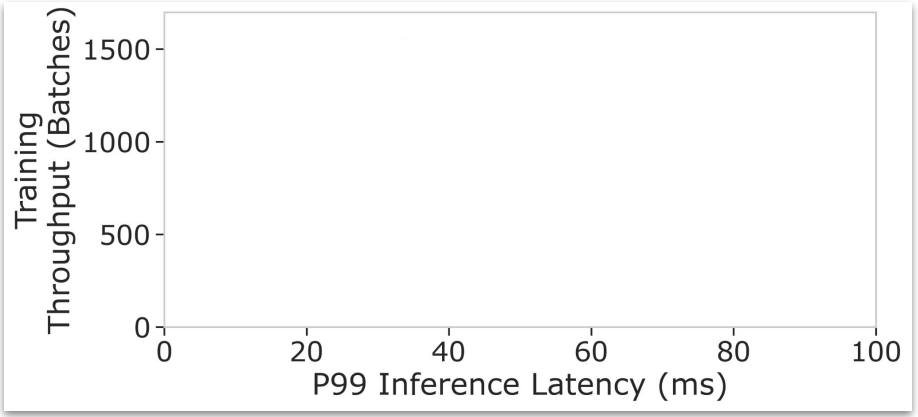


	Transparent	Dynamic	Autonomous
MPS **U**nlimited	✗	✔	✔
MPS **L**imited	✗	✗	✗

Systems under Test

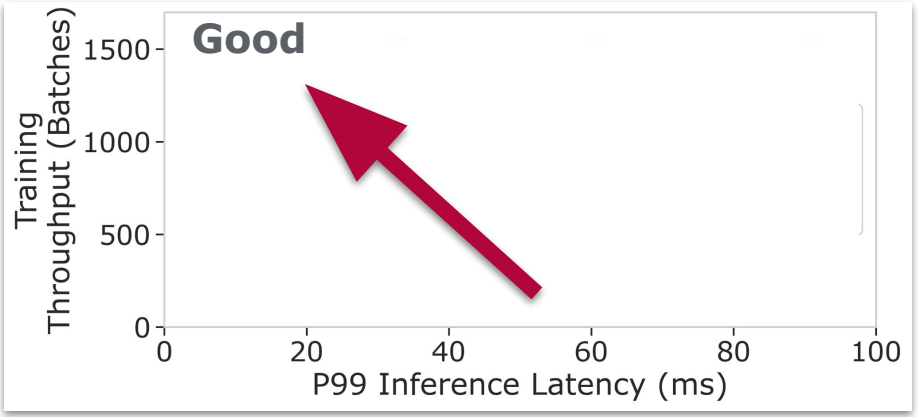
	Transparent	Dynamic	Autonomous
MPS **U**nlimited	✗	✔	✔
MPS **L**imited	✗	✗	✗
Draco	✔	✔	✔

Results



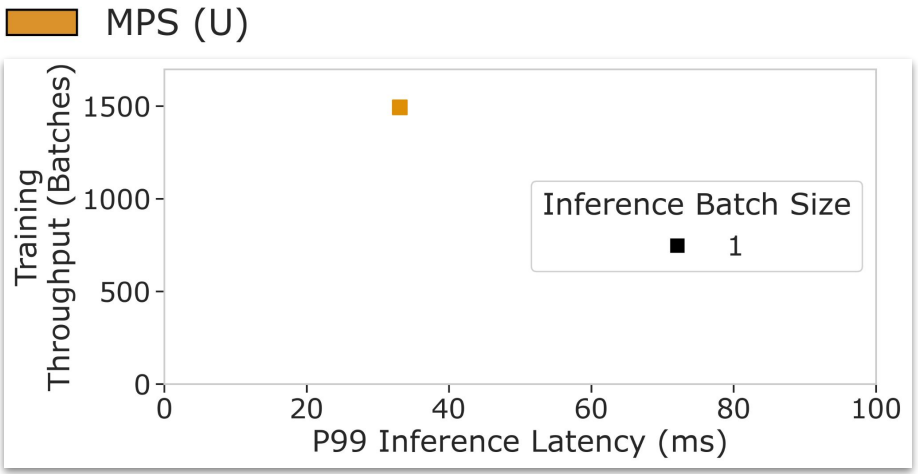
Workload B

Results



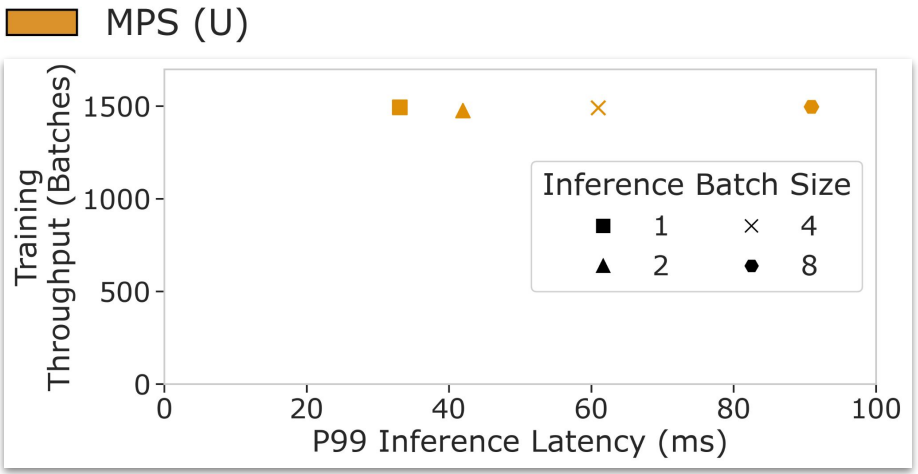
Workload B

Results



Workload B

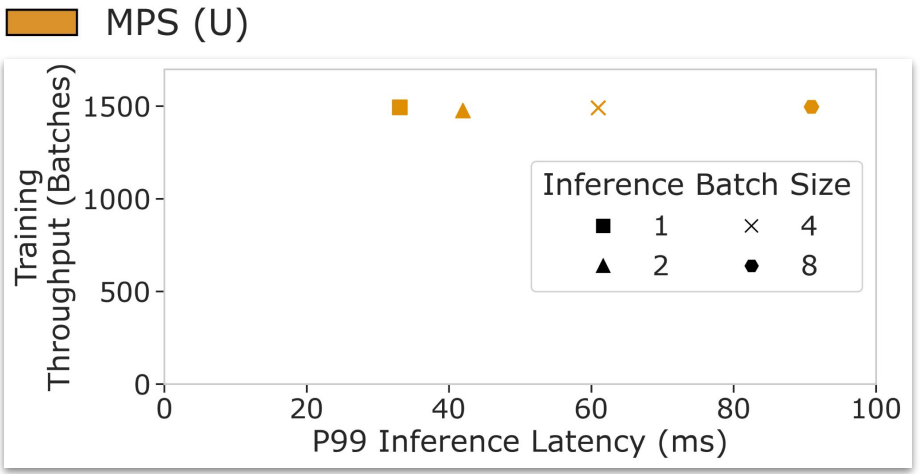
Results



Workload B

Results

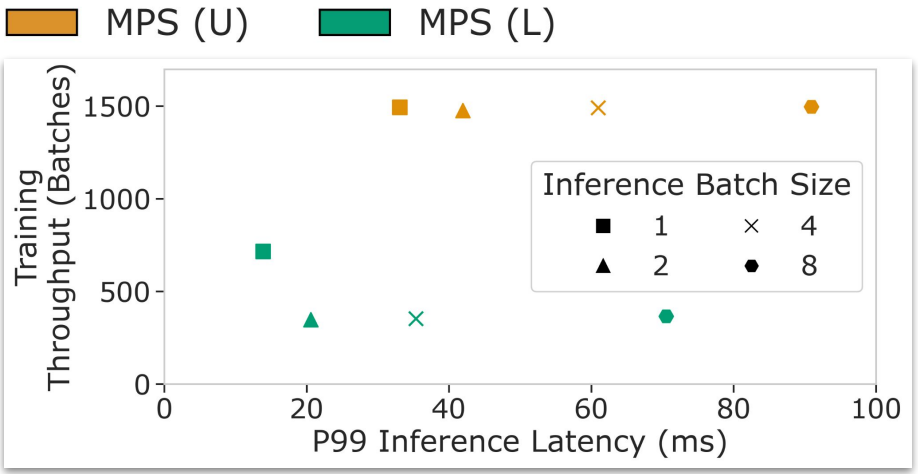
- MPS (U) favours throughput over latency



Workload B

Results

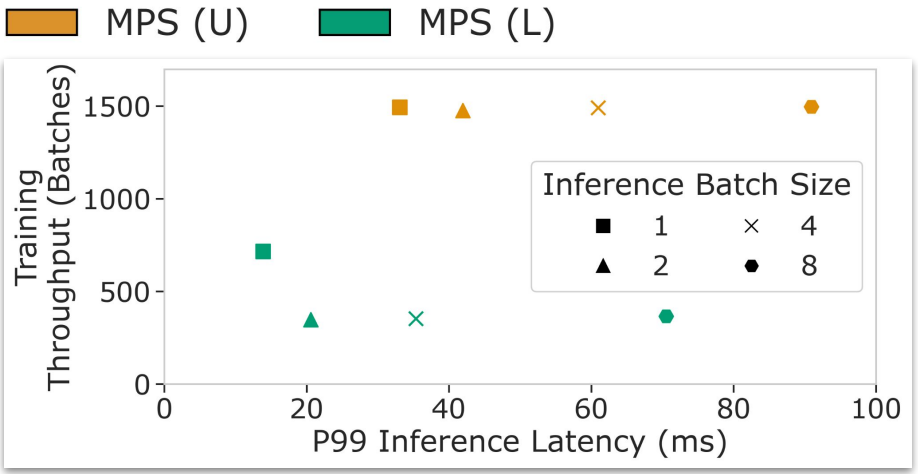
- MPS (U) favours throughput over latency



Workload B

Results

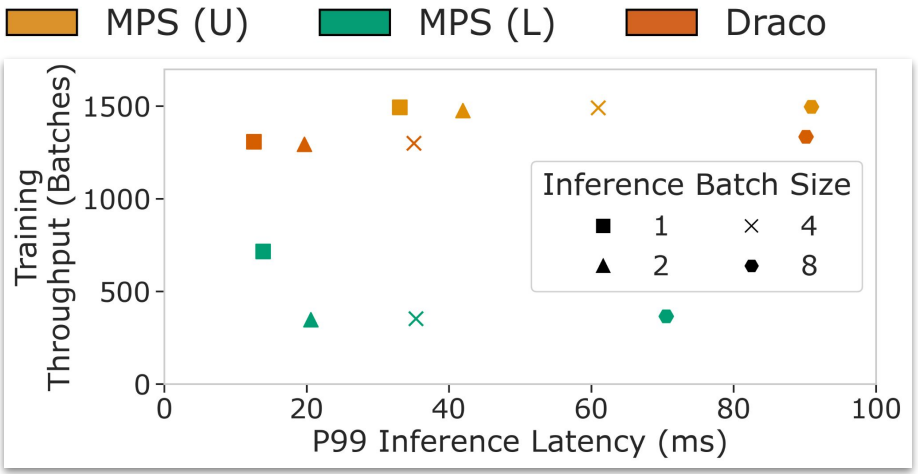
- MPS (U) favours throughput over latency
- MPS (L) favours latency over throughput



Workload B

Results

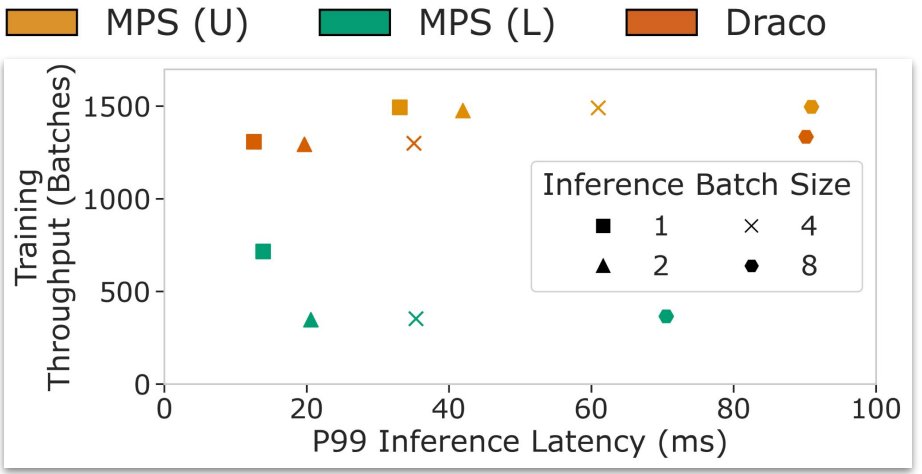
- MPS (U) favours throughput over latency
- MPS (L) favours latency over throughput



Workload B

Results

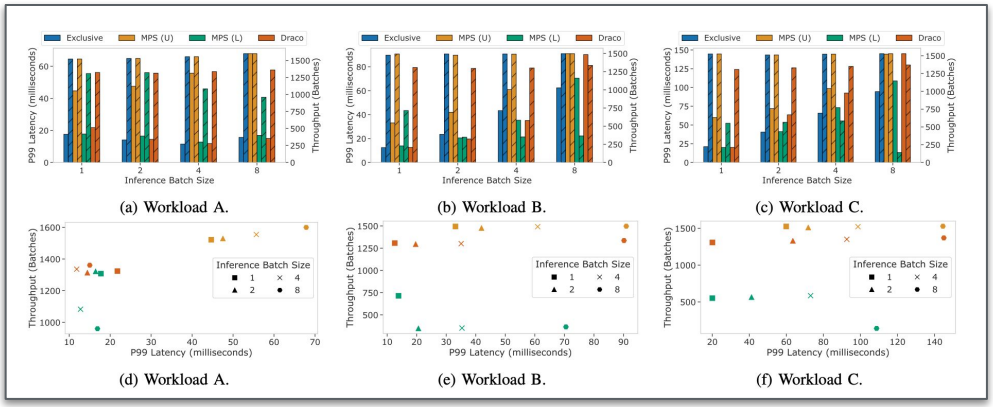
- MPS (U) favours throughput over latency
- MPS (L) favours latency over throughput
- Draco inference latency matches MPS (L) **and** achieves high training throughput

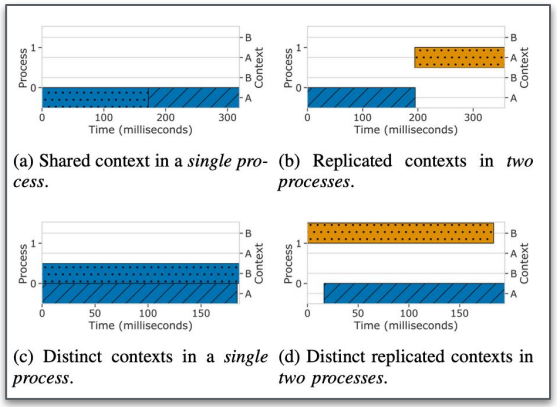


Workload B

Results

- Workloads A, C and more experiments in the paper





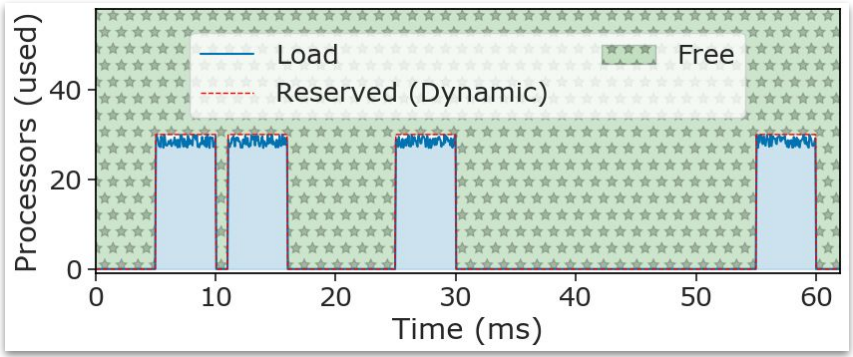
Summary


Challenges & Goals
System Design
Requirements & Contributions
Evaluation

Summary

- ❏ Draco often outperforms MPS
- ❏ Low inference latency
- ❏ High training throughput
- ❏ Draco is **applicable** and **autonomous**
- ❏ Not limited to ML workloads

- ❏ Open source project
- ❏ Schedule multiple inference clients



Dynamic Partitioning

Hasso Plattner Institut
Digital Engineering · Universität Potsdam

Thanks for your attention! Questions?

Theo.Radig@hpi.de

SPONSORED BY THE
Federal Ministry of Education and Research

KI Service Zentrum
by Hasso-Plattner-Institut