Sharing the GPU Communication Datapath across ML Workloads

Danyang Zhuo

Duke
UNIVERSITY

Danyang Zhuo

- Duke University
  - Assistant Professor of Computer Science (2020-)
- UC Berkeley
  - Postdoc at Electrical Engineering and Computer Science (2019-2020)
- University of Washington
  - PhD at Paul G. Allen School of Computer Science and Engineering (2013-2019)

My main research areas are

- Networking: RDMA + Collective Communication
- Performance of AI Systems: Training, Inference, Performance Prediction

Why Shared GPU Computing?

1. GPUs are expensive
2. ML workloads are memory- and compute-intensive.

Sharing is how we make the economics work.

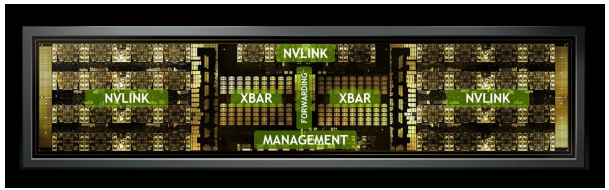
- Edge: One (or a few) consumer GPUs (e.g., RTX 4090, 5090). One developer runs multiple models. One model active at a time (**Temporal Multiplexing)**.
- Datacenter: A cluster of datacenter GPUs (e.g., A100, H100, B200). Many tenants, jobs arriving and leaving. Managed by Slurm / Kubernetes to keep utilization up (**Temporal + Spatial Multiplexing)**.

GPU Communication Datapath

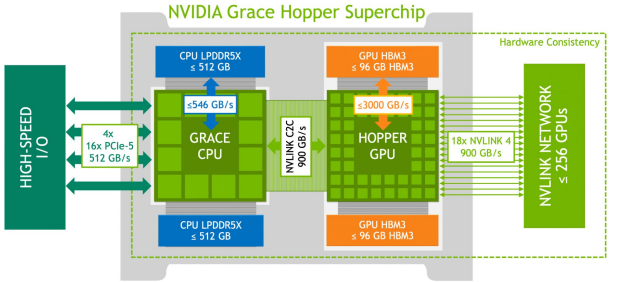
Sharing GPUs means sharing the paths between them and the rest of the system

- PICe
- NVLink, NVLink C2C, TPU ICI
- RDMA: Infiniband, RoCE

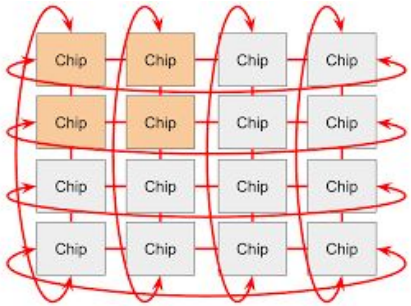
Improving the datapath is a focus of hardware vendors (e.g., NVIDIA, Google)



NVSwitch



Grace Hopper SuperChip



TPU Torus

Pictures from NVIDIA and https://jax-ml.github.io/ websites

How to Share GPU Communication Datapath Effectively?

	Edge	Datacenter
Step 1: Making sure hardware supports multiplexing		
Step 2: Resource-efficient workload orchestration		
Step 3: Reconfigure with confidence		

How to Share GPU Communication Datapath Effectively?

	Edge	Datacenter
Step 1: Making sure hardware supports multiplexing		Collie (NSDI22) + Husky (NSDI23): Testing RDMA Networking Correctness
Step 2: Resource-efficient workload orchestration	Nixie (OSDI26): Efficient Usage of PICe Bandwidth	MCCS (SIGCOMM24): Collective Communication for Shared Datacenter Fabric
Step 3: Reconfigure with confidence		Phantora (NSDI26): Predicting ML System Performance via Simulation

How to Share GPU Communication Datapath Effectively?

	Edge	Datacenter
Step 1: Making sure hardware supports multiplexing		Collie (NSDI22) + Husky (NSDI23): Testing RDMA Networking Correctness
Step 2: Resource-efficient workload orchestration	**Nixie (OSDI26): Efficient Usage of PICe Bandwidth**	MCCS (SIGCOMM24): Collective Communication for Shared Datacenter Fabric
Step 3: Reconfigure with confidence		Phantora (NSDI26): Predicting ML System Performance via Simulation

Edge ML

Let’s consider a workflow:

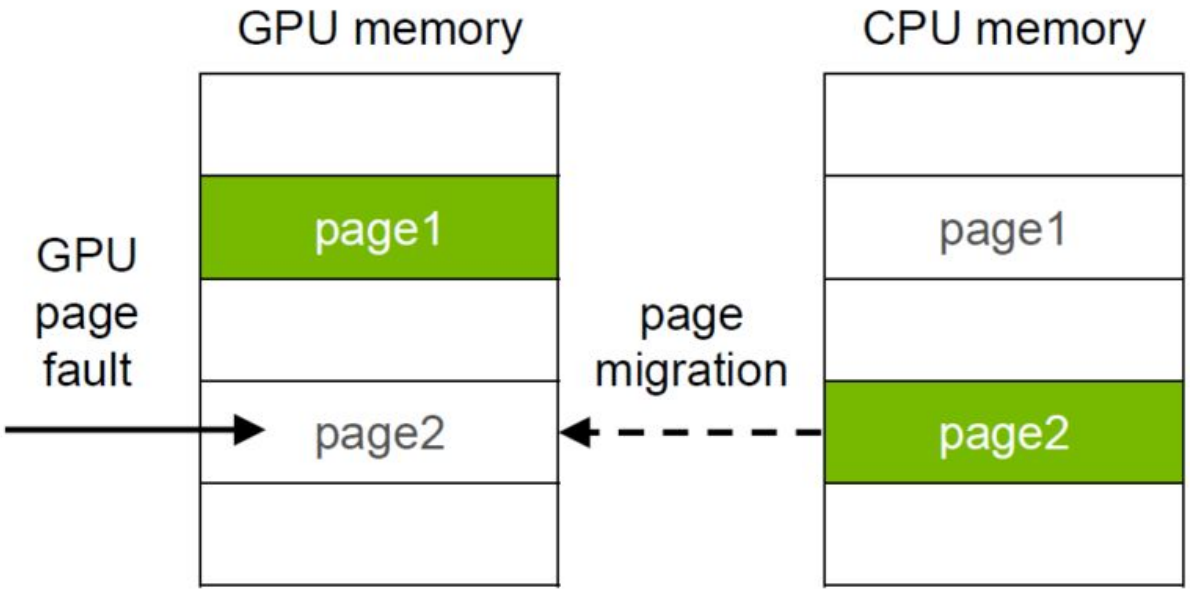
Loop:

- Using LLM to describe a scene (Model A: 30G)
- Using text2image model to generate an image (Model B:30G)

Let’s say you have 32G GPU, which option do you pick?

- Option 1: Run Model A’ (10G) + Model B’ (10G)
- Option 2: Run Model A (30G) + Model B (30G)

NVIDIA Unified Memory



Pictures from NVIDIA websites

Directly using NVIDA Unified Memory

Problem 1: Need excessive pinned memory

- Model A (30G) + Model B (30G) requires pinning 60G CPU memory

Problem 2: Thrashing

- Model A and B competes for GPU memory, and performance is terrible for both

Problem 3: Do not use PICe bandwidth well

- PCIe has full-duplex bandwidth, but empirically only one direction is fully utilized

Nixie: Efficient Temporal Multiplexing for GPUs.

Solution 1: Every data chunk has only 1 copy

Solution 2: Coordinating compute and memory management

Solution 3: Leveraging full duplex PCIe bandwidth

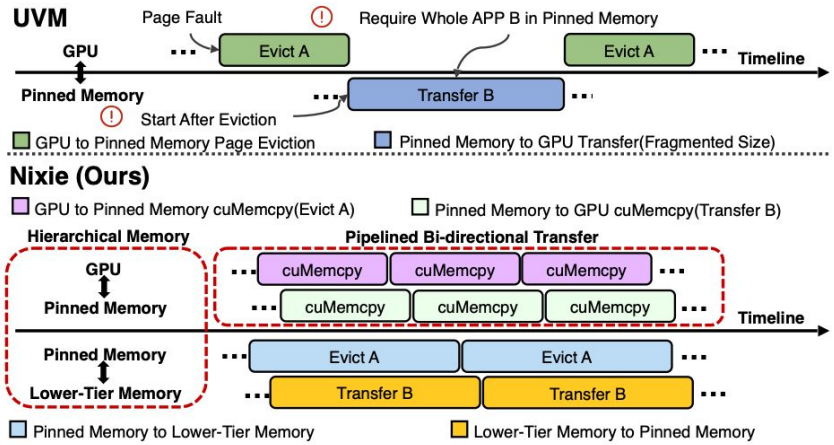


Figure 4: Nixie's communication bandwidth usage (1) between GPU memory and CPU pinned memory, and (2) between CPU pinned memory to lower-tier memory (e.g., CPU paged memory, disk).

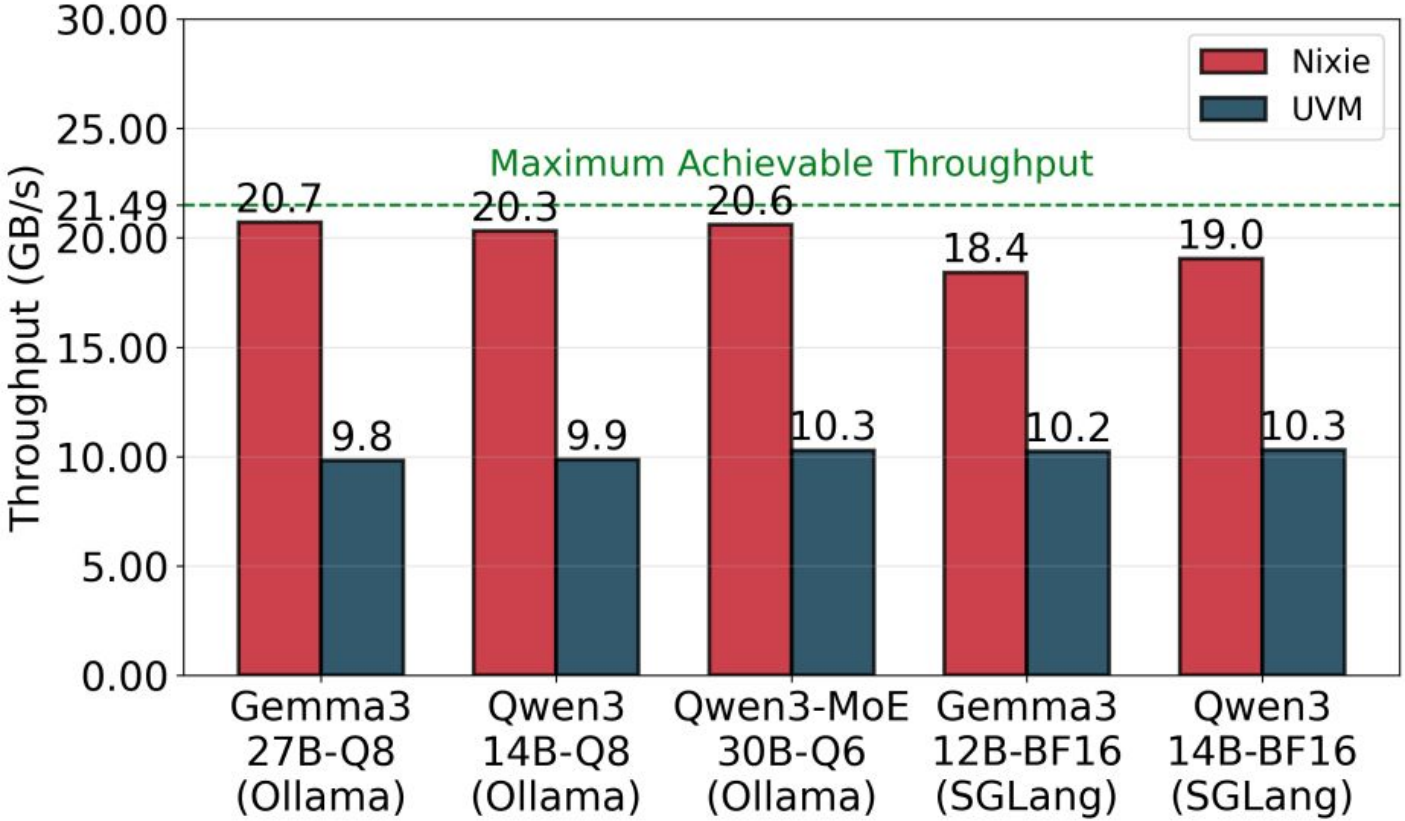


Figure 7: CPU from/to GPU memory copy throughput.

How to Share GPU Communication Datapath Effectively?

	Edge	Datacenter
Step 1: Making sure hardware supports multiplexing		**Collie (NSDI22) + Husky (NSDI23): Testing RDMA Networking Correctness**
Step 2: Resource-efficient workload orchestration	Nixie (OSDI26): Efficient Usage of PICe Bandwidth	MCCS (SIGCOMM24): Collective Communication for Shared Datacenter Fabric
Step 3: Reconfigure with confidence		Phantora (NSDI26): Predicting ML System Performance via Simulation

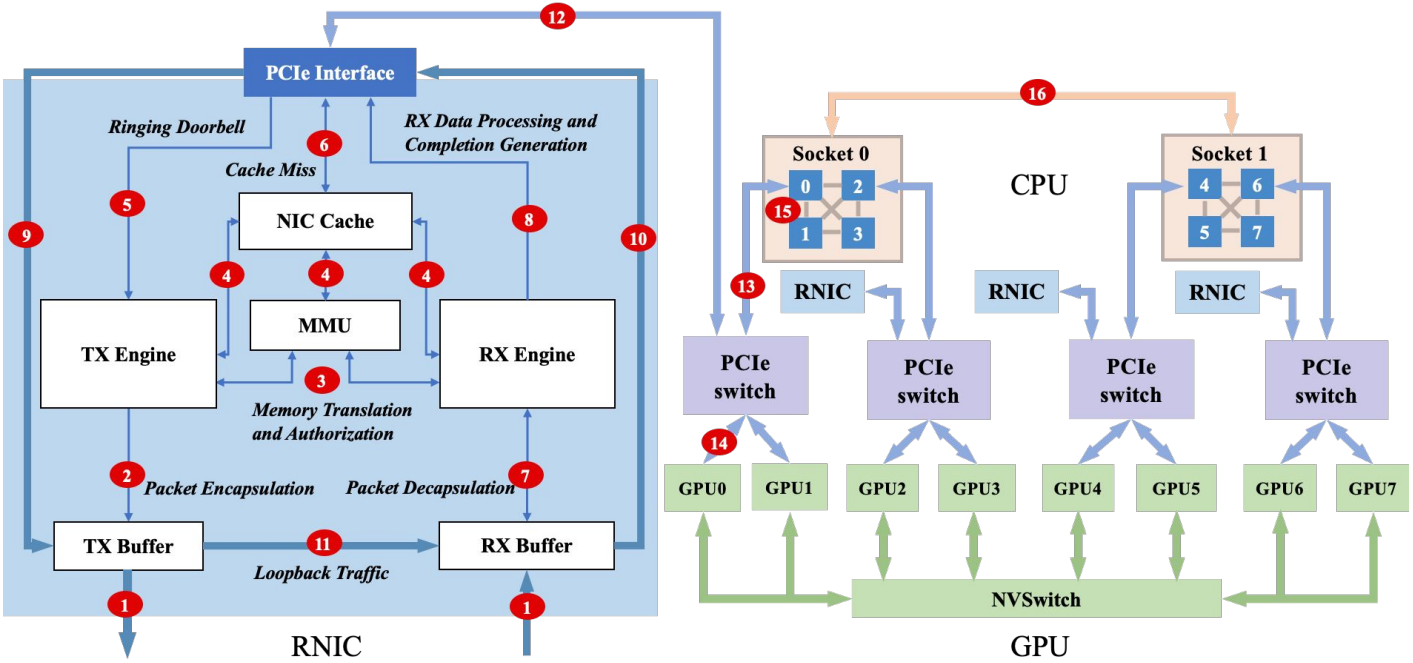
What’s a datacenter?

Pictures from NVIDIA websites

What’s a cloud?

Multiplexing one or more datacenters for different tenants, in order to achieve high hardware utilization.

Security Question: Can one tenant destroy another tenant's network performance?

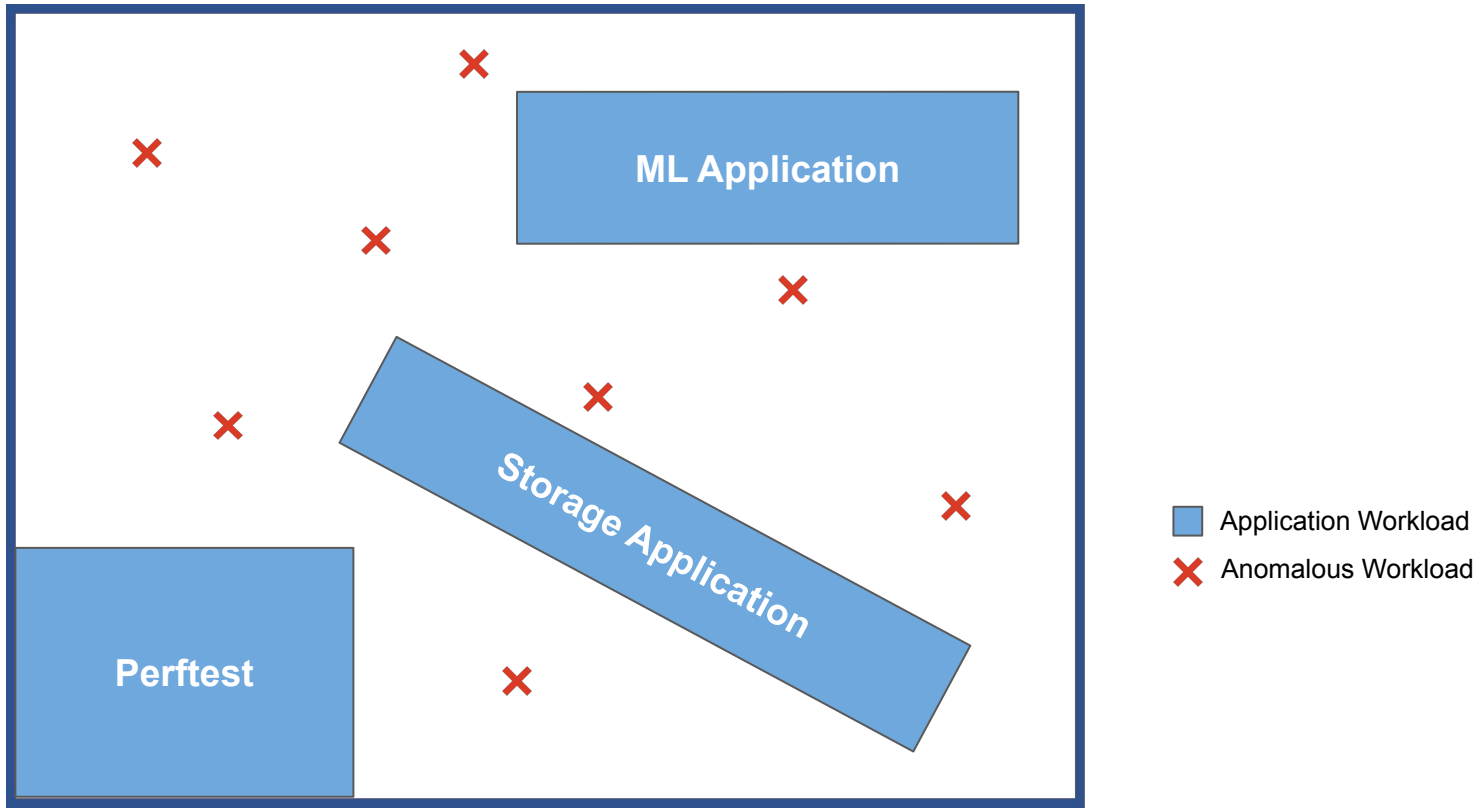


Existing tests

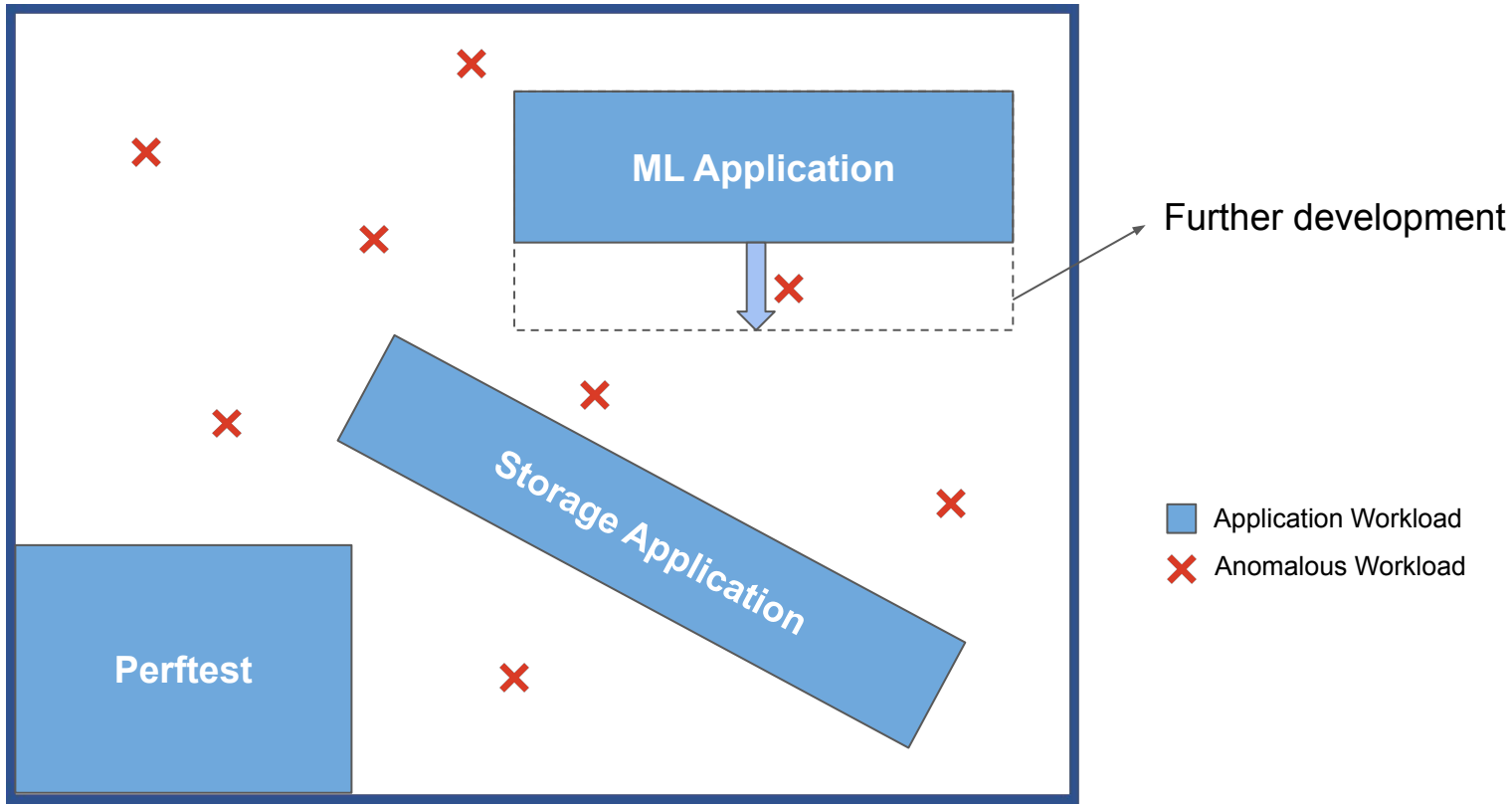
1. Run simple benchmarks (e.g., Perftest, OSU benchmarks) to conduct basic throughput and latency tests.

2. Run a representative set of the applications (e.g., distributed machine learning application) before real deployment.

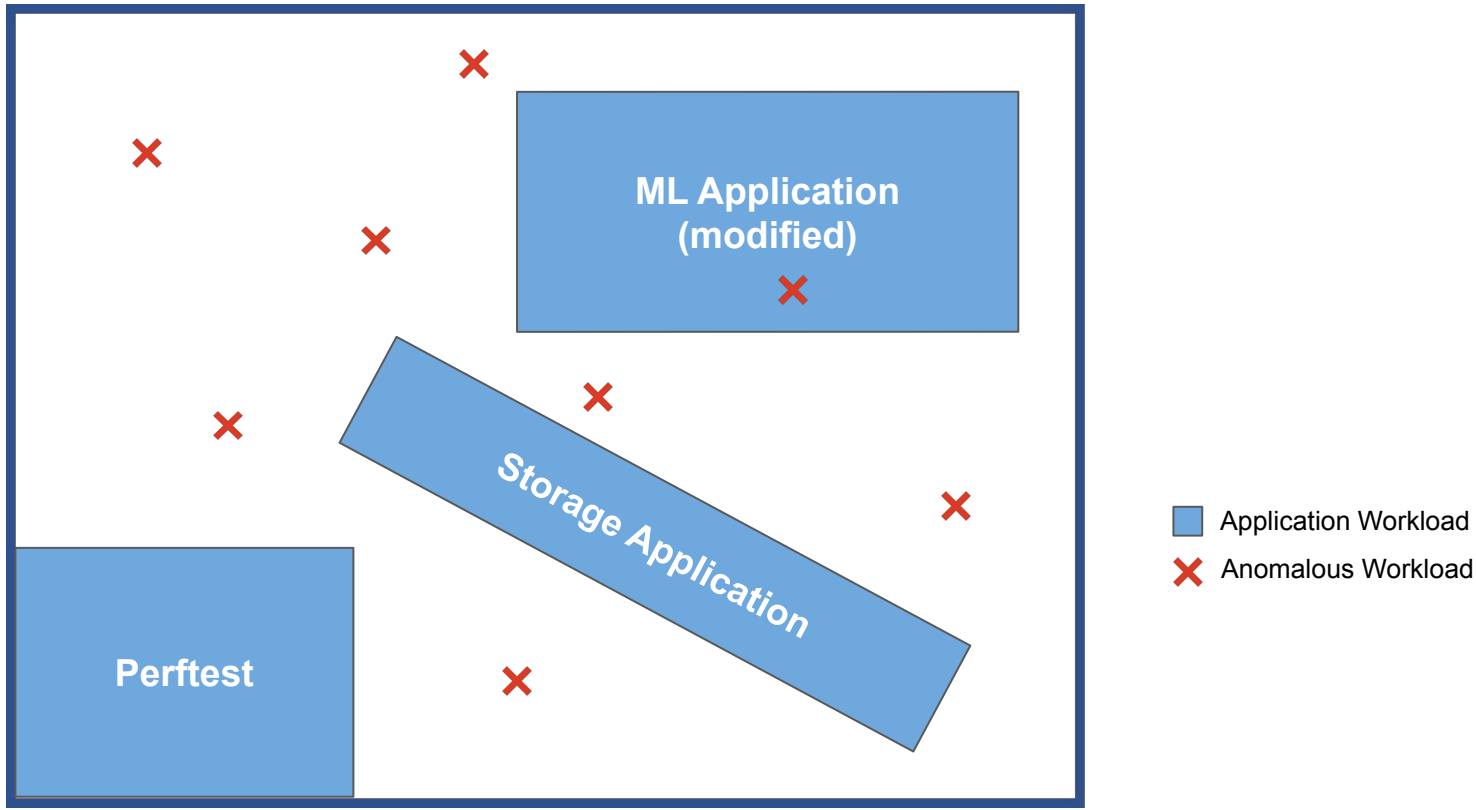
Why these integration tests are insufficient?



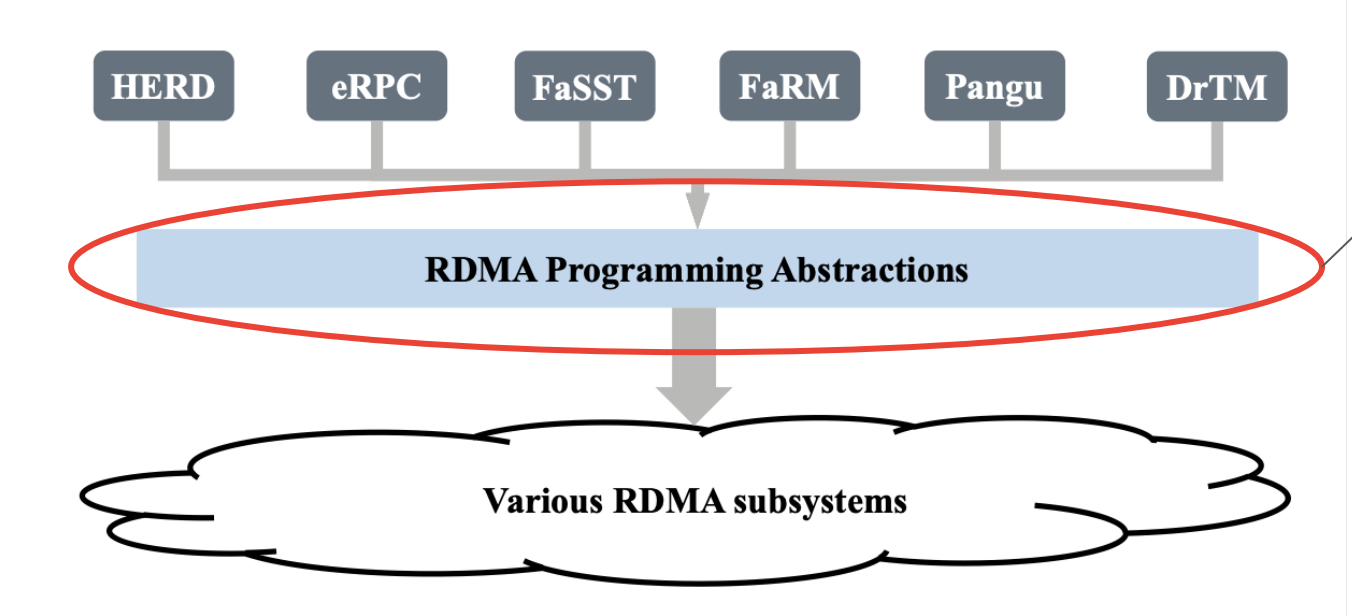
Why these integration tests are insufficient?



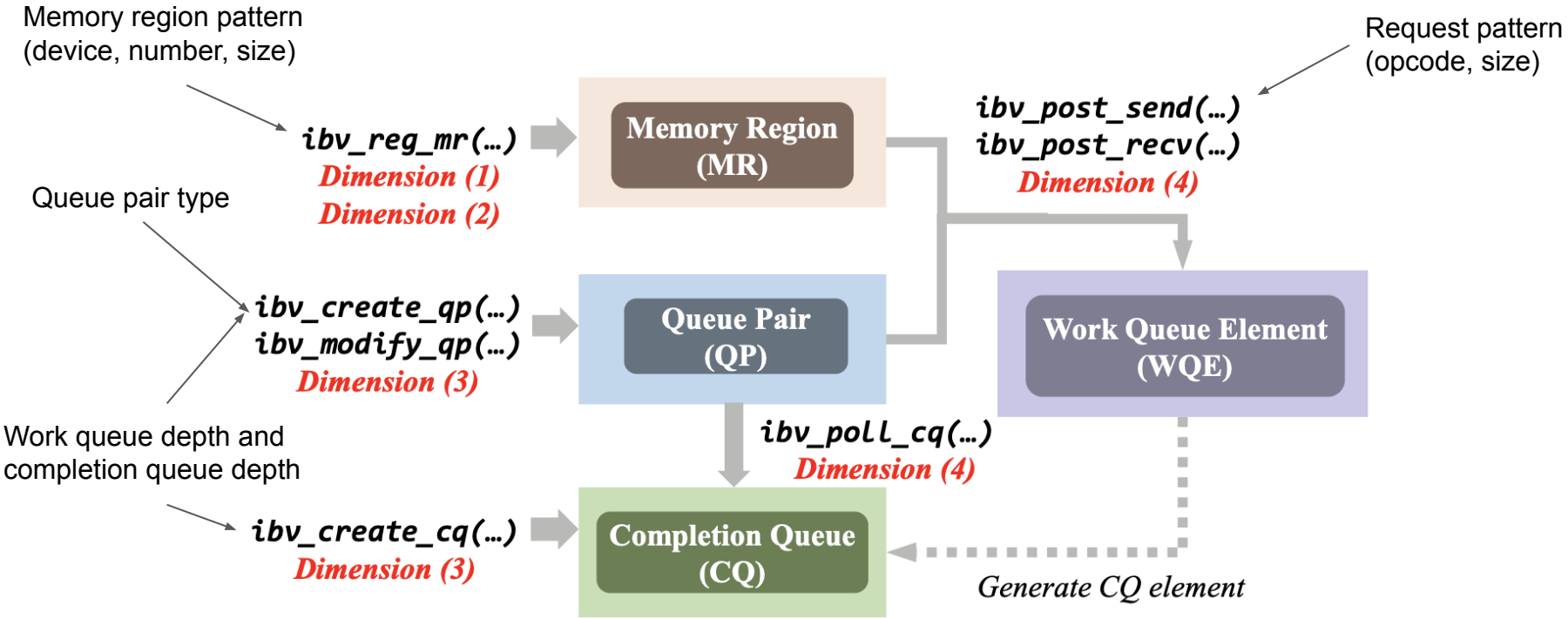
Why these integration tests are insufficient?



Solution #1: Finding the narrow waist



Solution #1: Finding the narrow waist

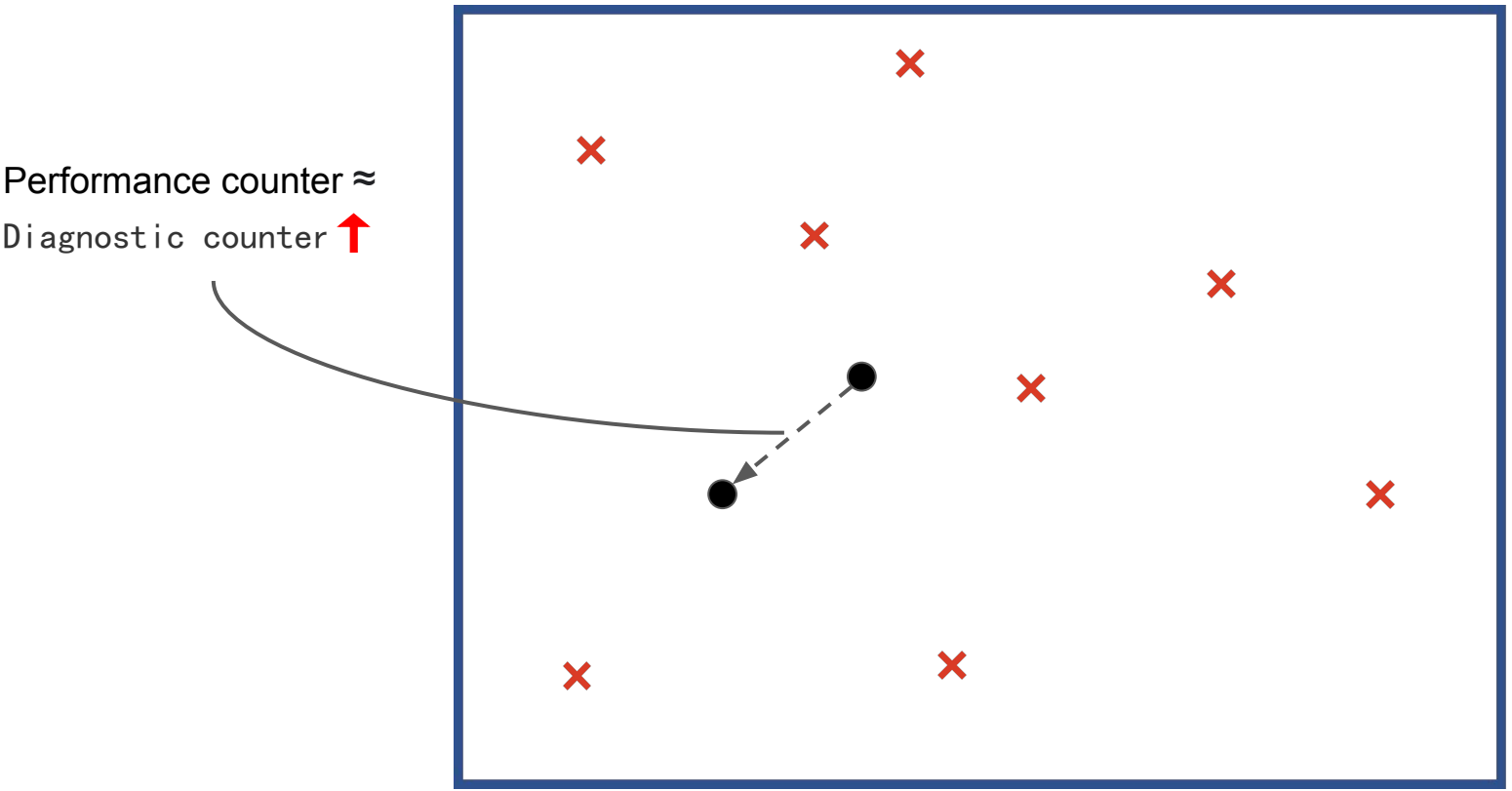


Solution #2: Hardware counter as a search signal

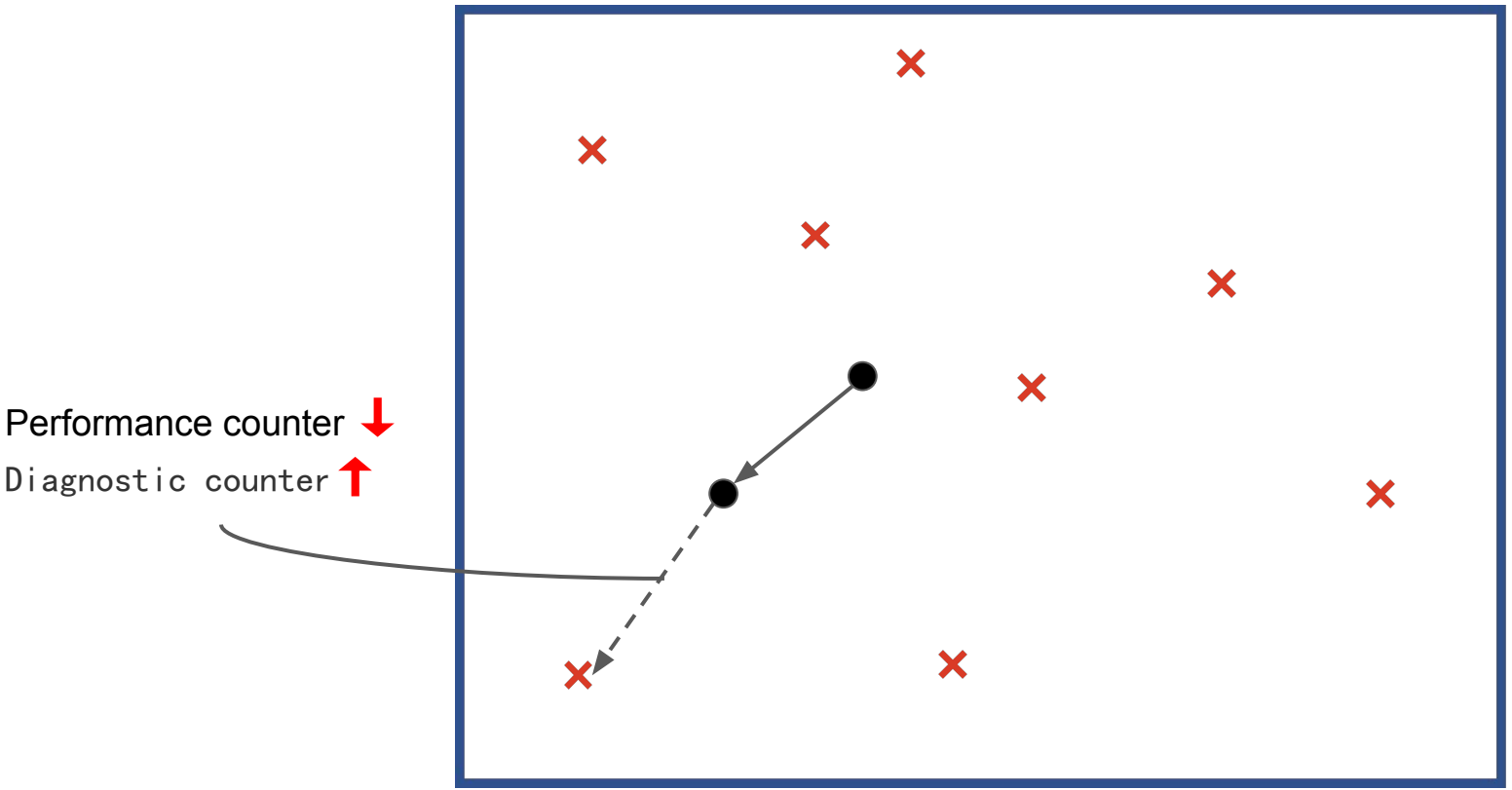
- We use two types of counters
  - Performance counters (e.g., bits per second)
  - Diagnostic counters (e.g., PCIe backpressure)

- The lower/higher the performance/diagnostic counter is, the test case is more likely to trigger an anomaly.

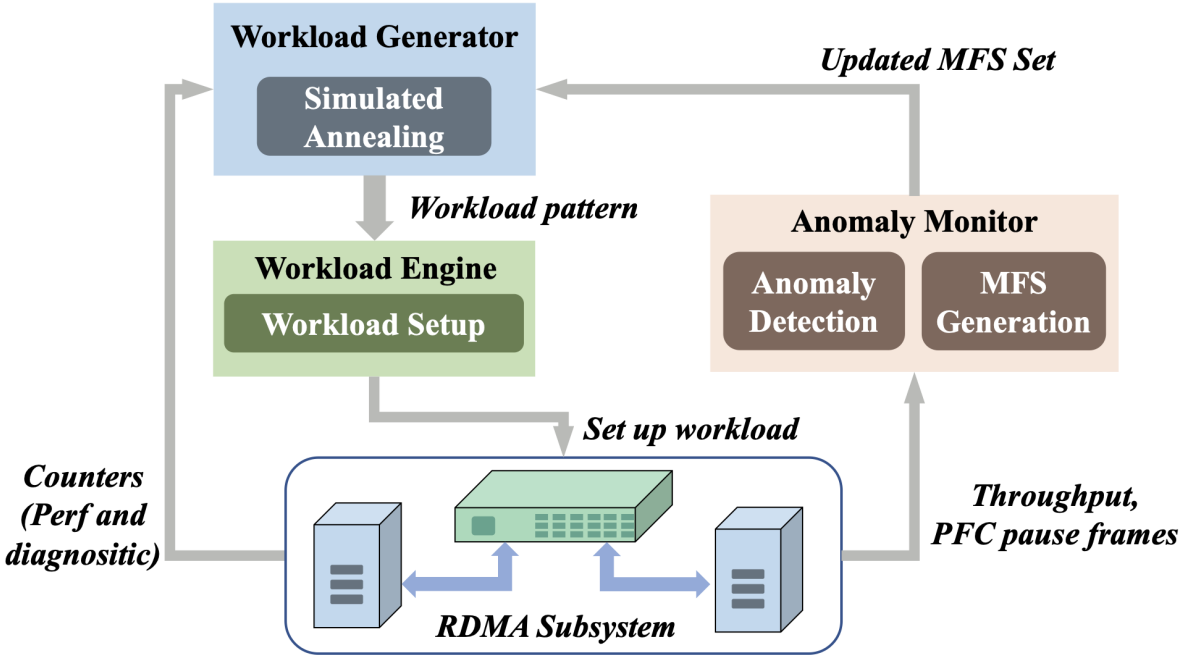
Solution #2: Hardware counter as a search signal



Solution #2: Hardware counter as a search signal



Implementation



DETAILS

This section provides a summary of potential vulnerabilities that this security update addresses and their impact. Descriptions use CWE™, and base scores and vectors use CVSS v3.1 standards.

CVE ID	Description	Base Score	Vector
CVE-2023-0204	NVIDIA ConnectX-5, ConnectX-6, and ConnectX6-DX contain a vulnerability in the NIC firmware, where an unprivileged user can cause improper handling of exceptional conditions, which may lead to denial of service.	6.5	CWE-703 AV:L/AC:L/PR:L/UI:N/S:C/C:N/I:N/A:H
CVE-2023-0203	NVIDIA ConnectX-5, ConnectX-6, and ConnectX6-DX contain a vulnerability in the NIC firmware, where an unprivileged user can exploit insufficient granularity of access control, which may lead to denial of service.	5.0	CWE-1220 AV:N/AC:L/PR:L/UI:N/S:C/C:N/I:N/A:L
CVE-2023-0205	NVIDIA ConnectX-5, ConnectX-6, and ConnectX6-DX contain a vulnerability in the NIC firmware, where an unprivileged user can exploit insufficient granularity of access control, which may lead to denial of service.	5.0	CWE-1220 AV:N/AC:L/PR:L/UI:N/S:C/C:N/I:N/A:L

The NVIDIA risk assessment is based on an average of risk across a diverse set of installed systems and may not represent the true risk to your local installation. NVIDIA recommends evaluating the risk to your specific configuration.

SECURITY UPDATES

The following table lists the NVIDIA products affected, versions affected, and the updated version that includes this security update.

CVE IDs Addressed	Product	Affected Versions	Updated Version
CVE-2023-0203 CVE-2023-0204 CVE-2023-0205	NVIDIA ConnectX Firmware	All versions prior to 35.1012	35.1012

NOTES

- Earlier firmware releases that support this product are also affected. If you are using an earlier release, upgrade to the latest release version.

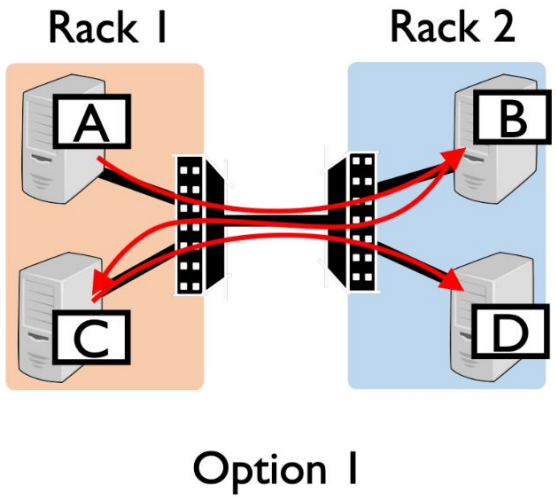
ACKNOWLEDGEMENTS

NVIDIA thanks Xinhao Kong, Jingrong Chen, Wei Bai, Yechen Xu, Mahmoud Elhaddad, Shachar Raindel, Jitendra Padhye, Alvin R. Lebeck, and Danyang Zhuo for reporting these issues.

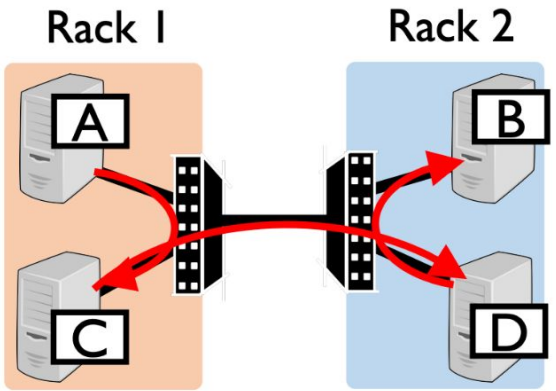
How to Share GPU Communication Datapath Effectively?

	Edge	Datacenter
Step 1: Making sure hardware supports multiplexing		Collie (NSDI22) + Husky (NSDI23): Testing RDMA Networking Correctness
Step 2: Resource-efficient workload orchestration	Nixie (OSDI26): Efficient Usage of PICe Bandwidth	**MCCS (SIGCOMM24): Collective Communication for Shared Datacenter Fabric**
Step 3: Reconfigure with confidence		Phantora (NSDI26): Predicting ML System Performance via Simulation

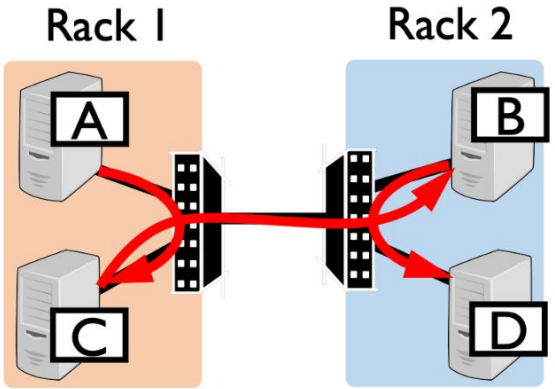
Choosing Collective Communication Algorithm?



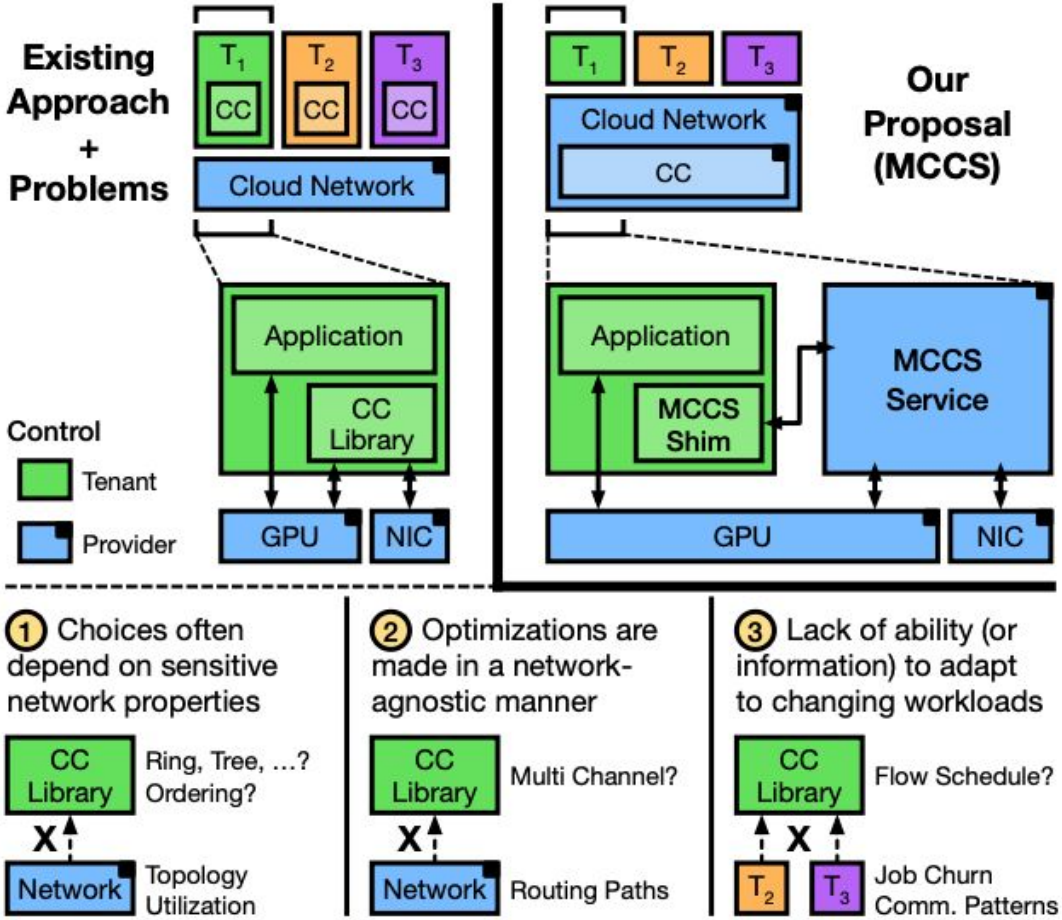
Option 1



Option 2



Option 3


Existing Approach + Problems
T1
T2
T3
CC
CC
CC
Cloud Network
Application
CC Library
GPU
NIC
Control
Tenant
Provider
Our Proposal (MCCS)
T1
T2
T3
Cloud Network
CC
Application
MCCS Shim
MCCS Service
GPU
NIC
1 Choices often depend on sensitive network properties
CC Library
Ring, Tree, …? Ordering?
X
Network
Topology Utilization
2 Optimizations are made in a network-agnostic manner
CC Library
Multi Channel?
X
Network
Routing Paths
3 Lack of ability (or information) to adapt to changing workloads
CC Library
Flow Schedule?
X
T2
T3
Job Churn Comm. Patterns

How to Share GPU Communication Datapath Effectively?

	Edge	Datacenter
Step 1: Making sure hardware supports multiplexing		Collie (NSDI22) + Husky (NSDI23): Testing RDMA Networking Correctness
Step 2: Resource-efficient workload orchestration	Nixie (OSDI26): Efficient Usage of PICe Bandwidth	MCCS (SIGCOMM24): Collective Communication for Shared Datacenter Fabric
Step 3: Reconfigure with confidence		**Phantora (NSDI26): Predicting ML System Performance via Simulation**

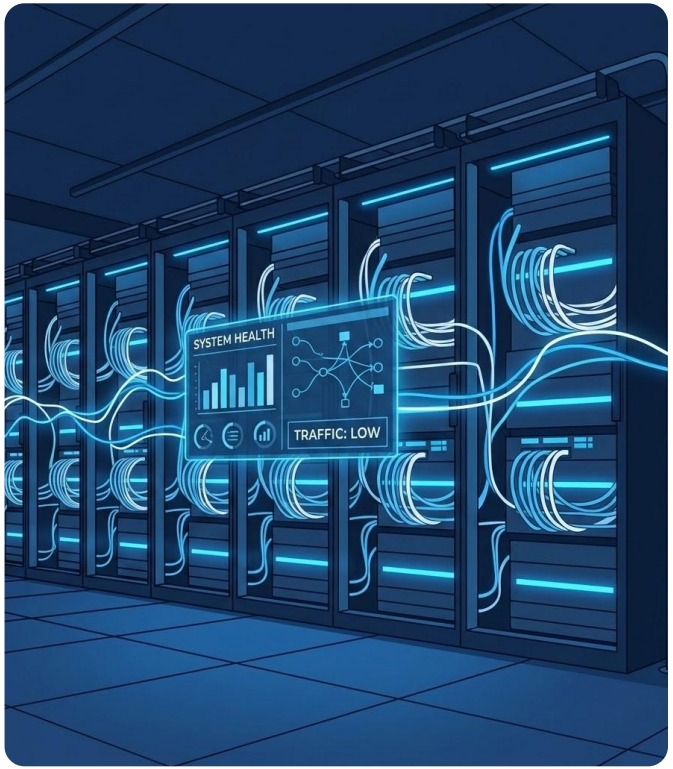
Performance Prediction is Critical

Economic Constraint

Real-world tuning costs enormous resources

Resource Planning

Provisioning decisions before hardware availability



Performance Modelling is Difficult

Mathematical Model (e.g., cos
model, roofline)

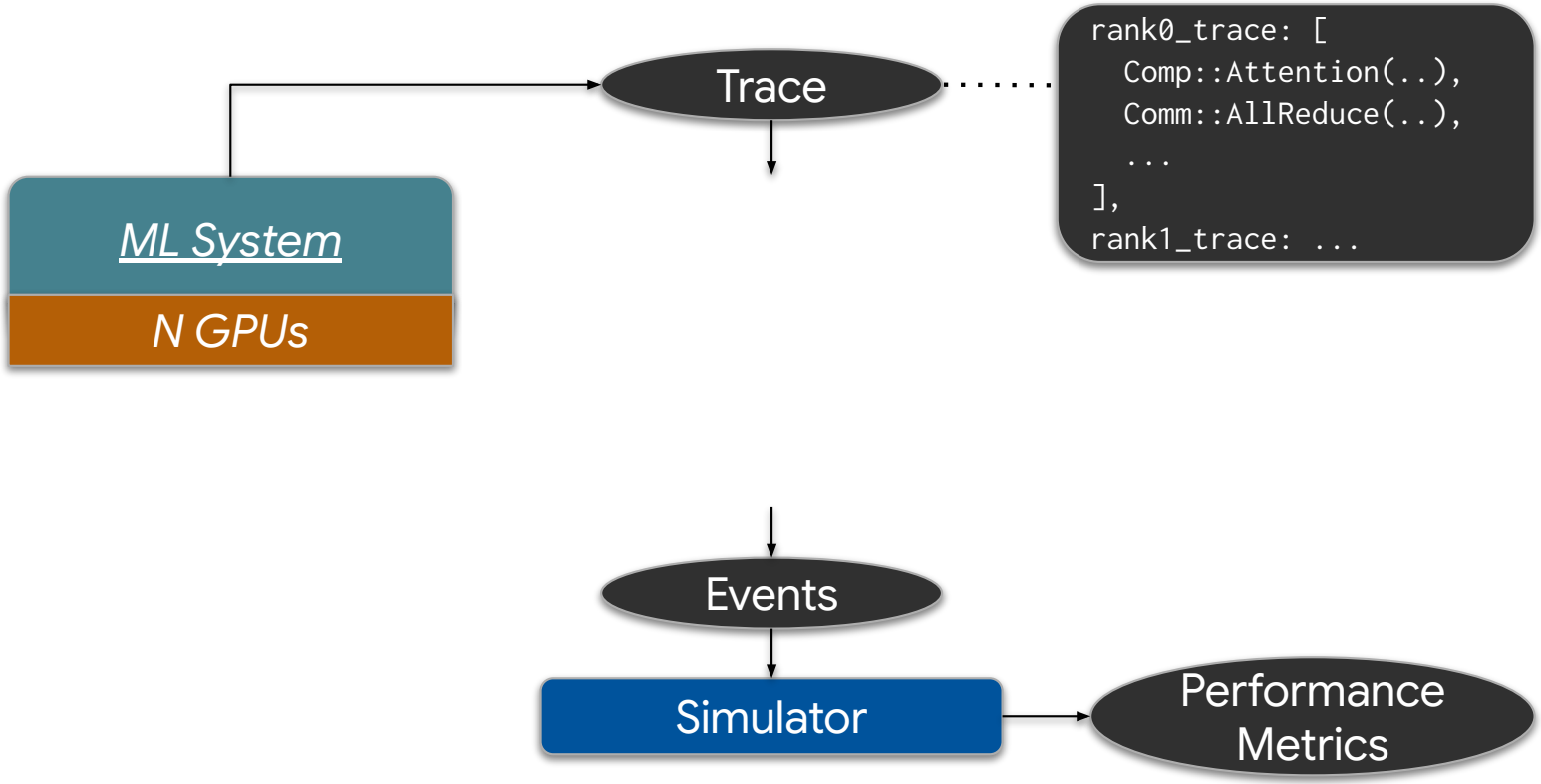
1. Hard to do cost model for different parallelisms design
2. Frameworks have their specific features

Alpa: Automating Inter- and Intra-Operator Parallelism for Distributed Deep Learning

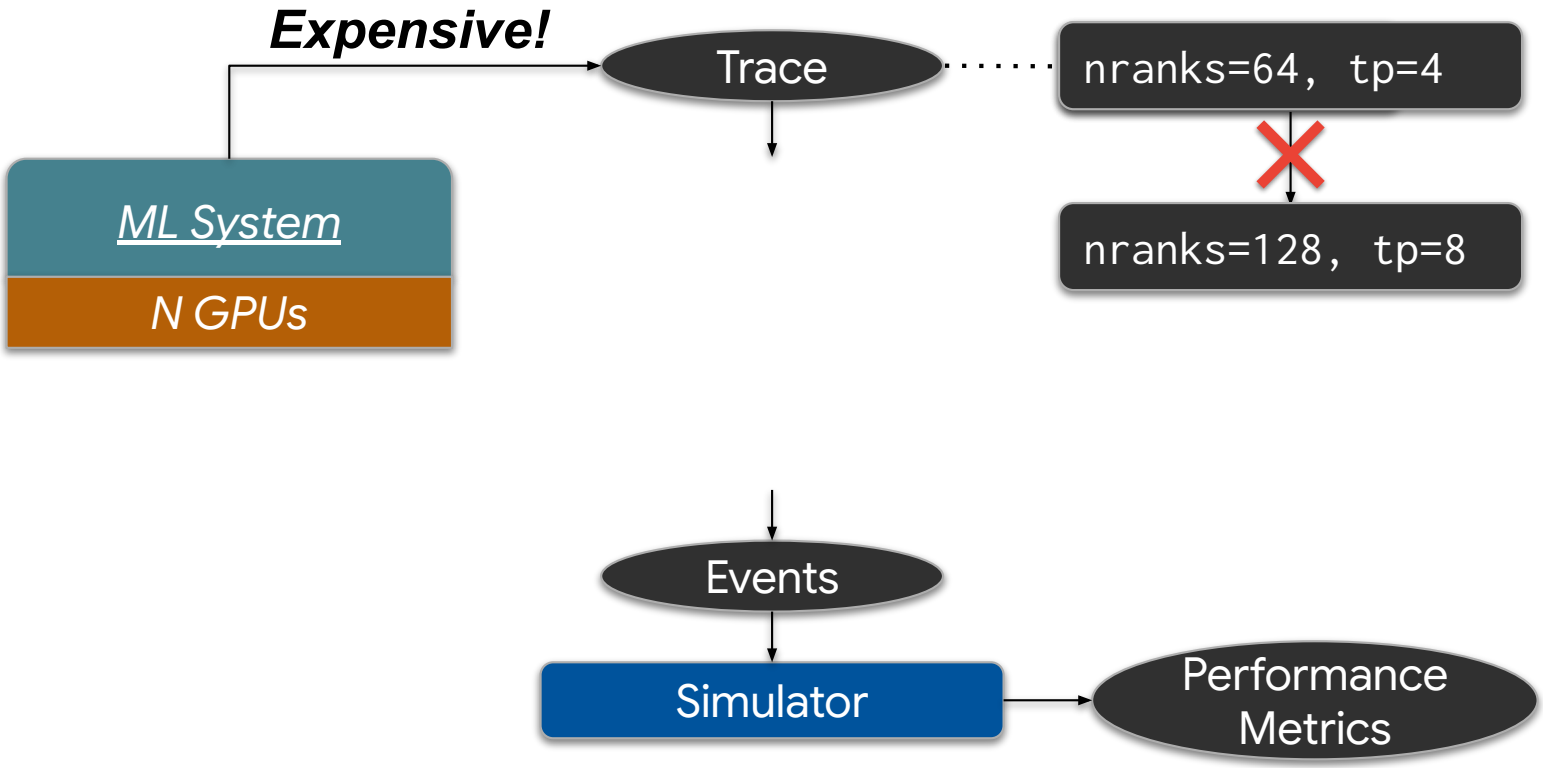
Lianmin Zheng[1,*] Zhuohan Li[1,*] Hao Zhang[1,*] Yonghao Zhuang[4]
Zhifeng Chen[3] Yanping Huang[3] Yida Wang[2] Yuanzhong Xu[3] Danyang Zhuo[6] Eric P. Xing[5]
Joseph E. Gonzalez[1] Ion Stoica[1]

[1]UC Berkeley [2]Amazon Web Services [3]Google [4]Shanghai Jiao Tong University
[5]MBZUAI, Carnegie Mellon University [6]Duke University

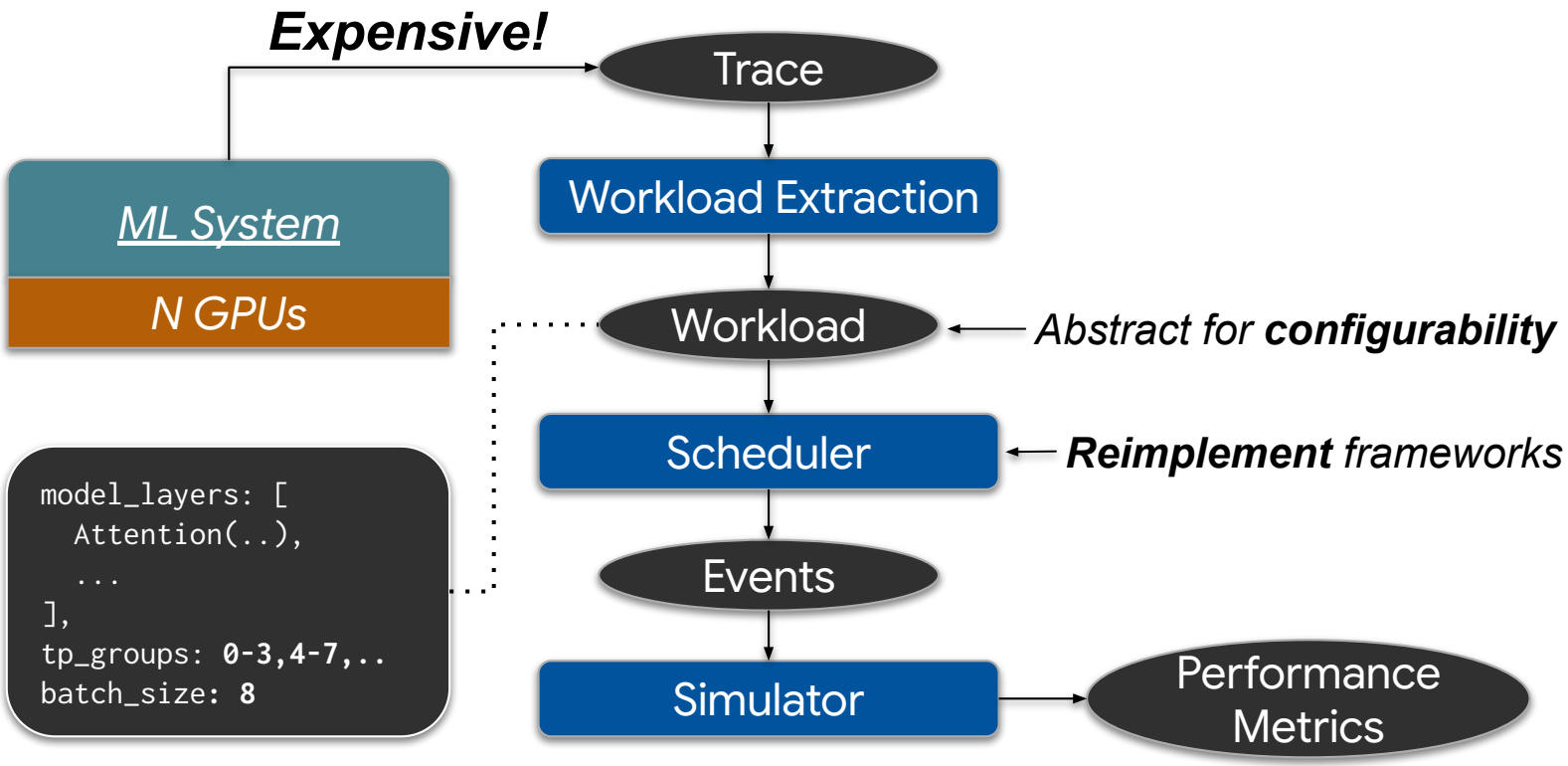
Trace-based Simulations



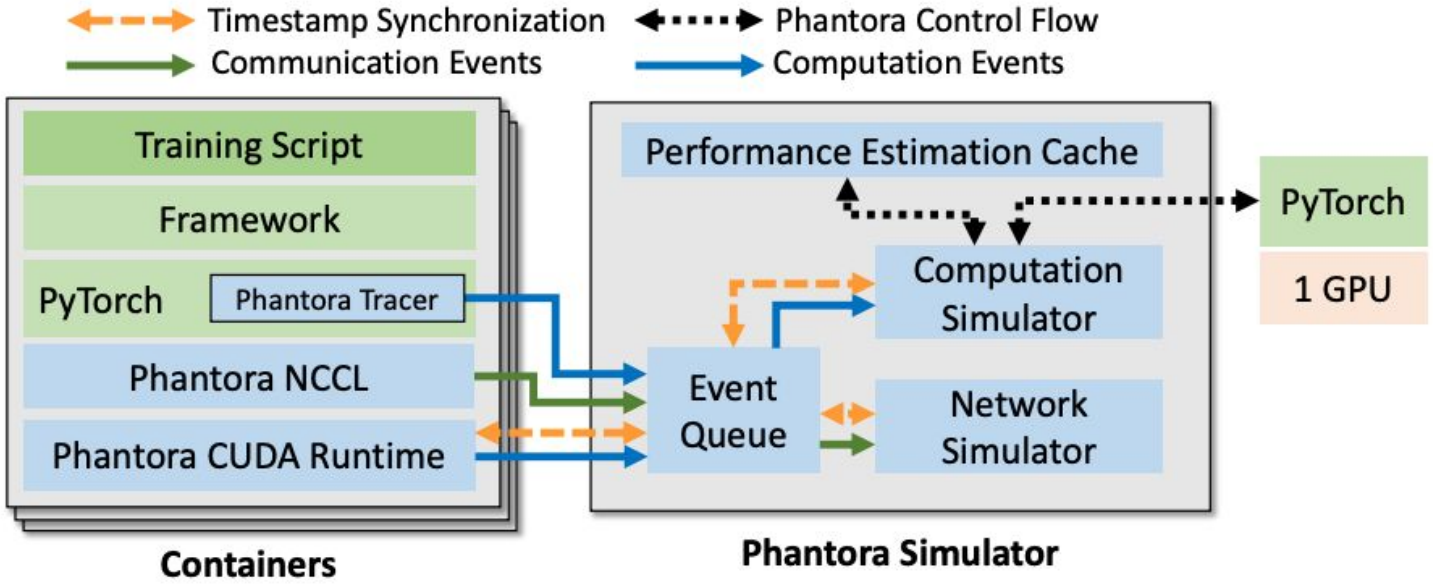
Trace-based Simulations



Trace-based Simulations



Phantora Simulator



Emulating CUDA Runtime and NCCL

GPU Memory Tracking

Track cudaMalloc & cudaFree, returning errors when exceeding configured capacity.

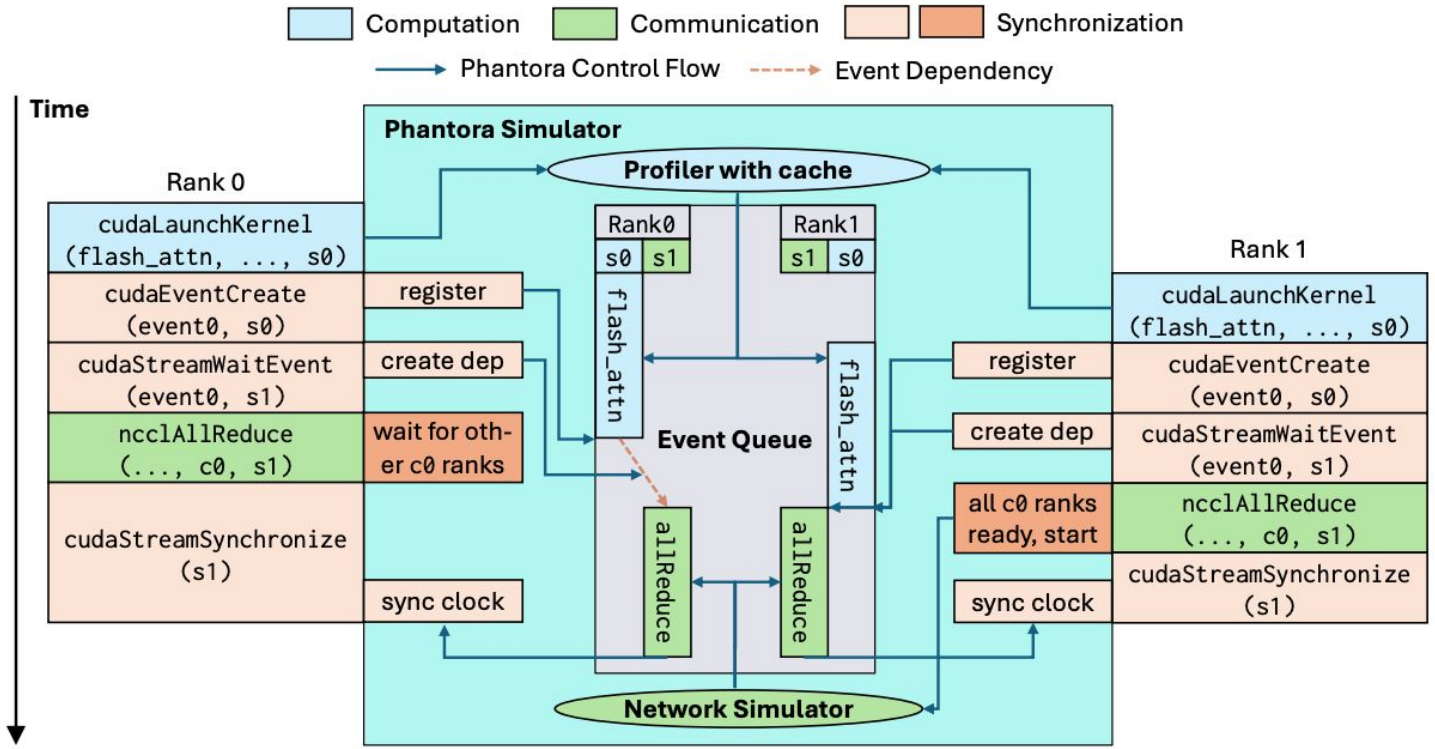
CUDA Streams & Events

Precise management of dependencies and synchronizations, including overlaps.

NCCL Communications

Handling complex synchronizations required for NCCL setup and collective operations.

Emulating CUDA Runtime and NCCL



Using Unmodified Training Frameworks

```
## install-requirements.sh
pip install megatron-core deepspeed torchtitan

## train.py
+from phantora_utils import (
+  enable_function_tracer,
+  disable_function_tracer,
+)

......

if __name__ == “__main__”:
+   enable_function_tracer()
    train()
+   disable_function_tracer()
```

Using Unmodified Training Frameworks

```
logger.info(
    f"step: {train_state.step:2}  "
    f"loss: {global_avg_loss:7.4f}  "
    f"memory: {gpu_mem_stats.max_reserved_gib:5.2f}GiB"
    f"({gpu_mem_stats.max_reserved_pct:.2f}%)  "
    f"wps: {round(wps):,}  "
    f"mfu: {mfu:.2f}%{color.reset}"
)
```

```
[rank0]:2024-09-19 14:46:00,990 - root - INFO - step:  5  loss:  0.0000
memory: 41.56GiB(51.95%)  wps: 2,876  mfu: 53.38%
[rank0]:2024-09-19 14:46:37,206 - root - INFO - step: 10  loss:  0.0000
memory: 41.56GiB(51.95%)  wps: 2,887  mfu: 53.60%
[rank0]:2024-09-19 14:47:13,114 - root - INFO - step: 15  loss:  0.0000
memory: 41.56GiB(51.95%)  wps: 2,902  mfu: 53.87%
[rank0]:2024-09-19 14:47:47,622 - root - INFO - step: 20  loss:  0.0000
memory: 41.56GiB(51.95%)  wps: 2,904  mfu: 53.90%
[rank0]:2024-09-19 14:48:22,964 - root - INFO - step: 25  loss:  0.0000
memory: 41.56GiB(51.95%)  wps: 2,905  mfu: 53.92%
[rank0]:2024-09-19 14:48:58,821 - root - INFO - step: 30  loss:  0.0000
memory: 41.56GiB(51.95%)  wps: 2,904  mfu: 53.90%
[rank0]:2024-09-19 14:49:34,490 - root - INFO - step: 35  loss:  0.0000
memory: 41.56GiB(51.95%)  wps: 2,886  mfu: 53.57%
[rank0]:2024-09-19 14:50:30,248 - root - INFO - step: 40  loss:  0.0000
memory: 41.56GiB(51.95%)  wps: 2,894  mfu: 53.72%
```

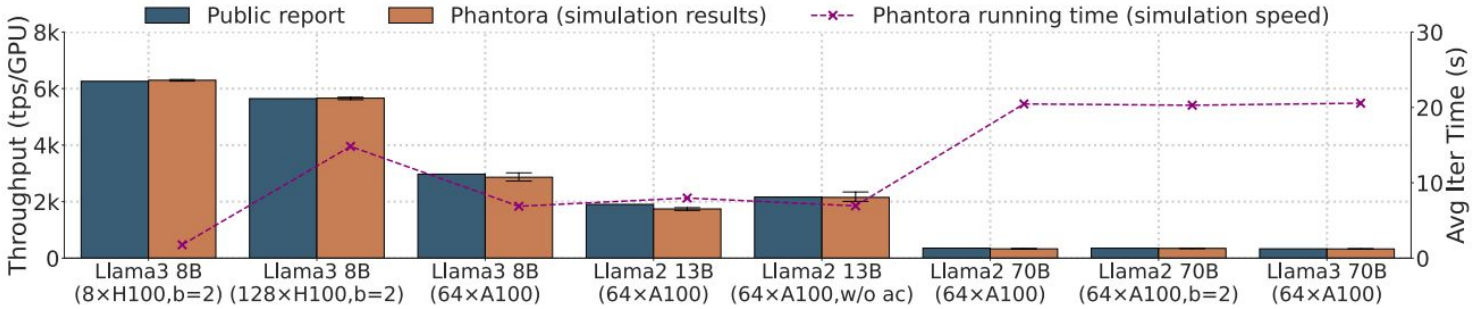
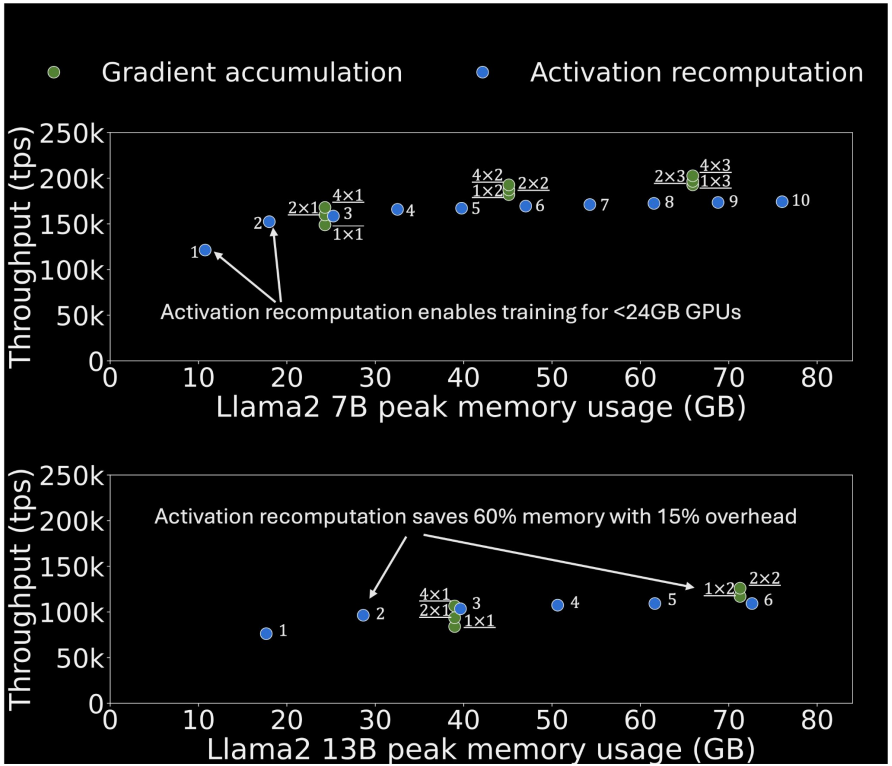


Figure 9: **Accuracy and speed of Phantora (large scale):** Training throughput reported by TorchTitan [21] using FSDP2, simulation results and simulation speed of Phantora on the testbeds. The error bars show 95% confidence interval. "ac" means activation checkpointing in TorchTitan [21].

Case Study: Tuning Megatron config under limited VRAM



Acknowledgements

I thank my PhD students for doing the heavy lifting, specifically:
Nixie (OSDI26): Yechen Xu
Collie (NSDI22): Xinhao Kong
Husky (NSDI23): Xinhao Kong, Jingrong Chen, Yechen Xu
MCCS (SIGCOMM24): Yongji Wu, Yechen Xu, Jingrong Chen
Phantora (NSDI26): Jianxing Qin, Jingrong Chen, Xinhao Kong, Yongji Wu

Collaborators for Nixie (OSDI26), Collie (NSDI22), Husky (NSDI23), MCCS (SIGCOMM24), and Phantora (NSDI26): Yifei Wang, Nathaniel Ren, Yiran Chen, Tianjun Yuan, Liang Luo, Zhaodong Wang, Ying Zhang, Tingjun Chen, Alvy Lebeck, Matthew Lentz, Wei Bai, Mahmoud Elhaddad, Shachar Raindel, Jitendra Padhye, Yibo Zhu, Huaping Zhou, Zhuo Jiang, Jianxi Ye, Chuanxiong Guo.

Summary

- GPUs are extremely expensive, making sharing inevitable.
- GPU datapath (i.e., PICe, NVLink, RDMA) has become a major performance bottleneck.
- We need to rethink how the datapath resources are managed, including
  - PICe bandwidth when switching between different applications
  - RDMA NIC resources
  - Choices of collective communication algorithms in a shared network
  - Implications for end-to-end AI system performance
- If you have thoughts in this direction, please don’t hesitate to contact me!
  - danyang@cs.duke.edu