Conspirator: SmartNIC-Aided Control Plane for Distributed ML Workloads

Yunming Xiao[1], Diman Zad Tootaghaj[2], Aditya Dhakal[2], Lianjie Cao[2], Puneet Sharma[2], Aleksandar Kuzmanovic[1]
[1] Northwestern University, [2] Hewlett Packard Labs

NORTHWESTERN UNIVERSITY

Hewlett Packard Labs

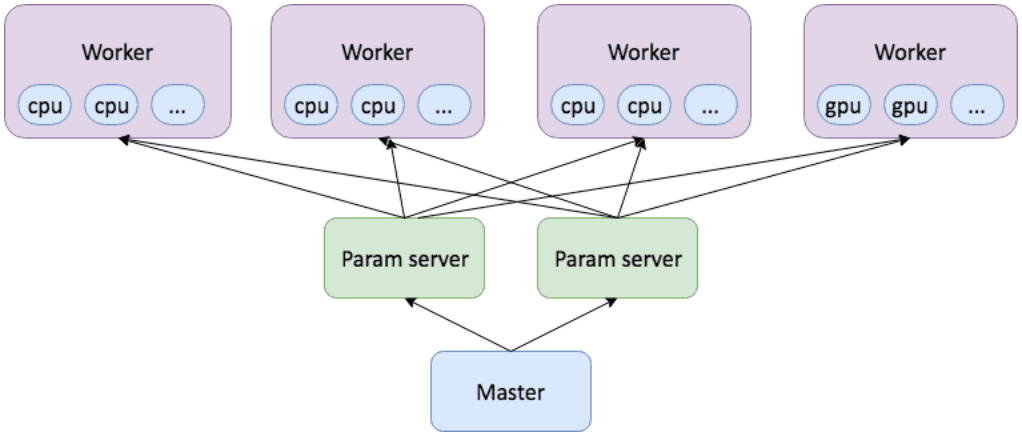
Table of Contents

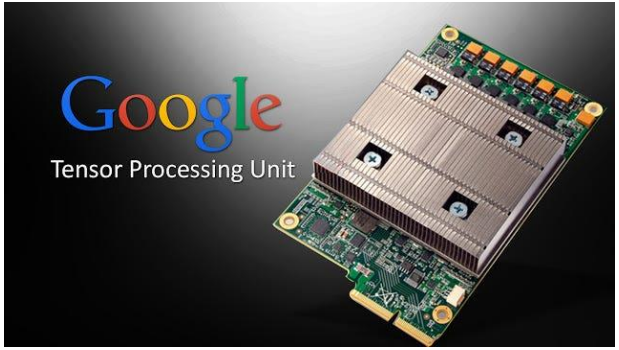
Background and Motivation

Machine learning is prevalent

Current practice of ML training/inference is

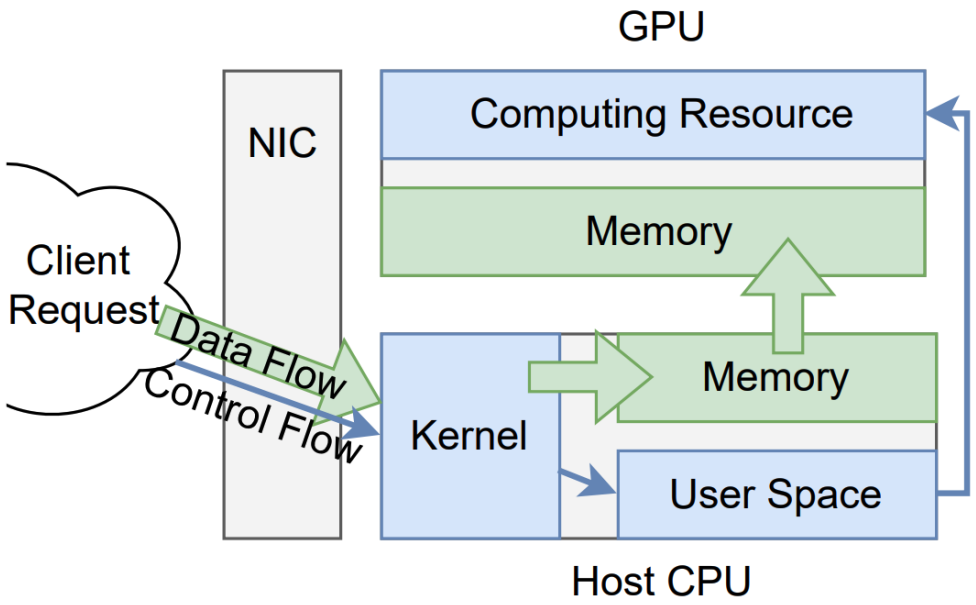
- Distributed
- Relying on accelerators





The control plane is still on CPU

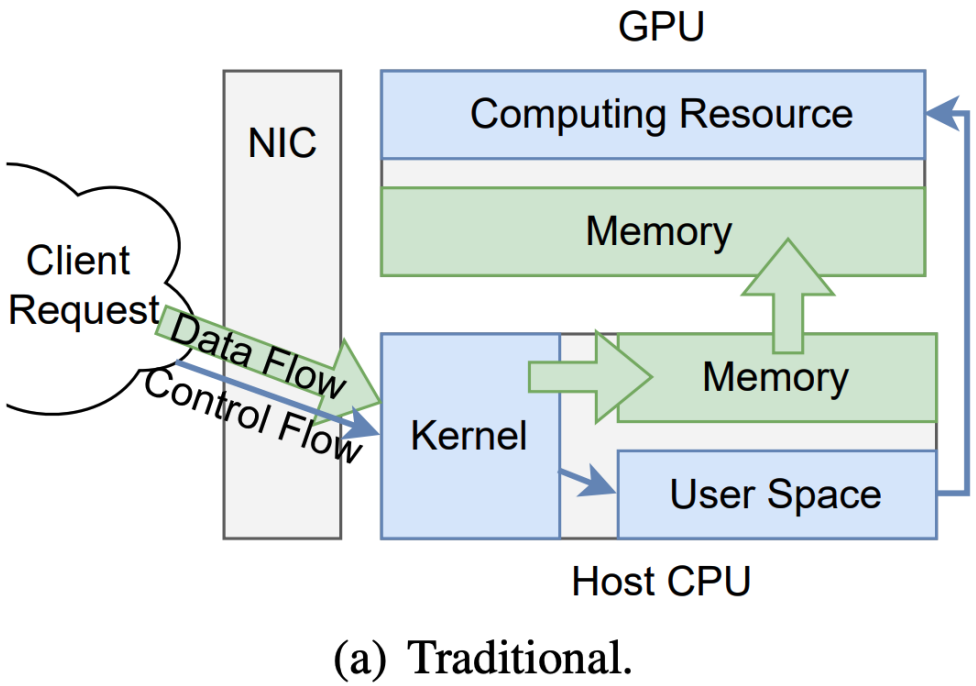
- CPU needs to handle many jobs
  - Data pre-processing
  - Network stack
  - Aided computation for ML
  - Background CPU activities
  - …



(a) Traditional.

The control plane is still on CPU

- High CPU utilization rate negatively impacts the ML performance[1]
- Inefficiency: bottleneck on CPUs

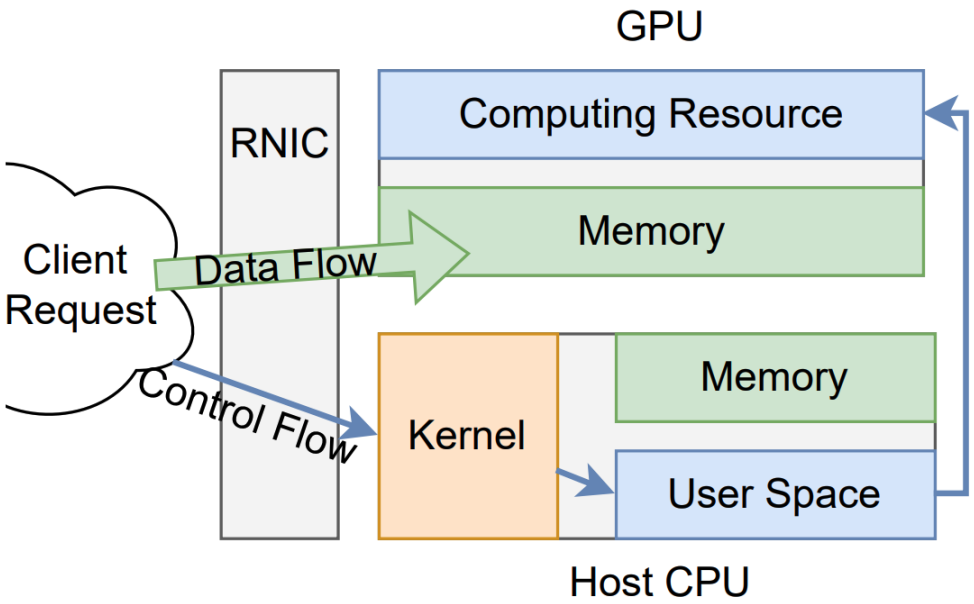


(a) Traditional.

[1] Weng et al., “MLaaS in the wild: Workload analysis and scheduling in large-scale heterogeneous GPU clusters”, NSDI’22.

Solution: minimize the CPU involvements

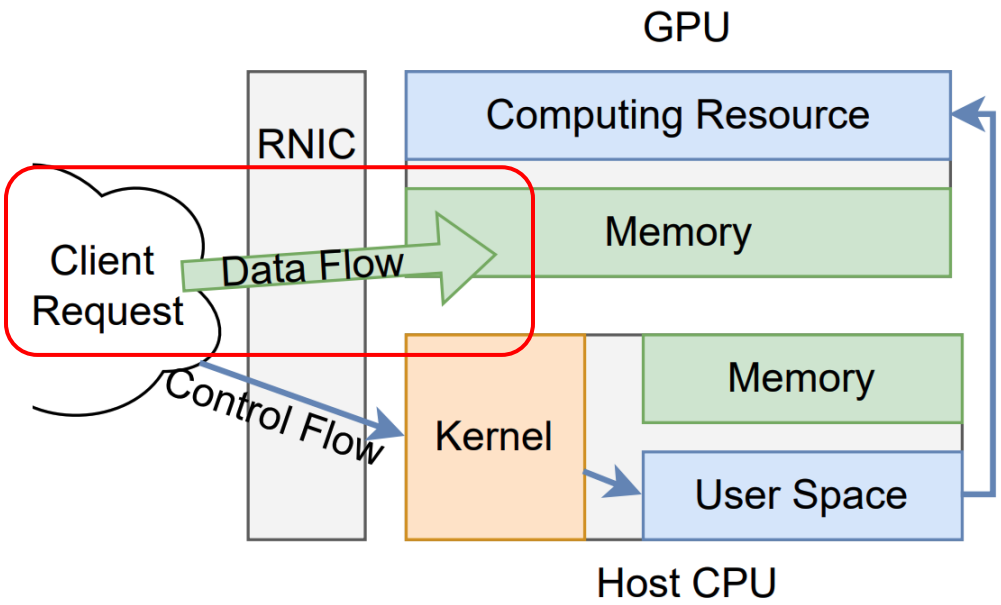
- We should remove any barrier
  - Kernel – DPDK
  - Host CPU – RDMA/GPUDirect



(b) CPU-bypass Design.

But RDMA is incompatible with accelerator scheduling

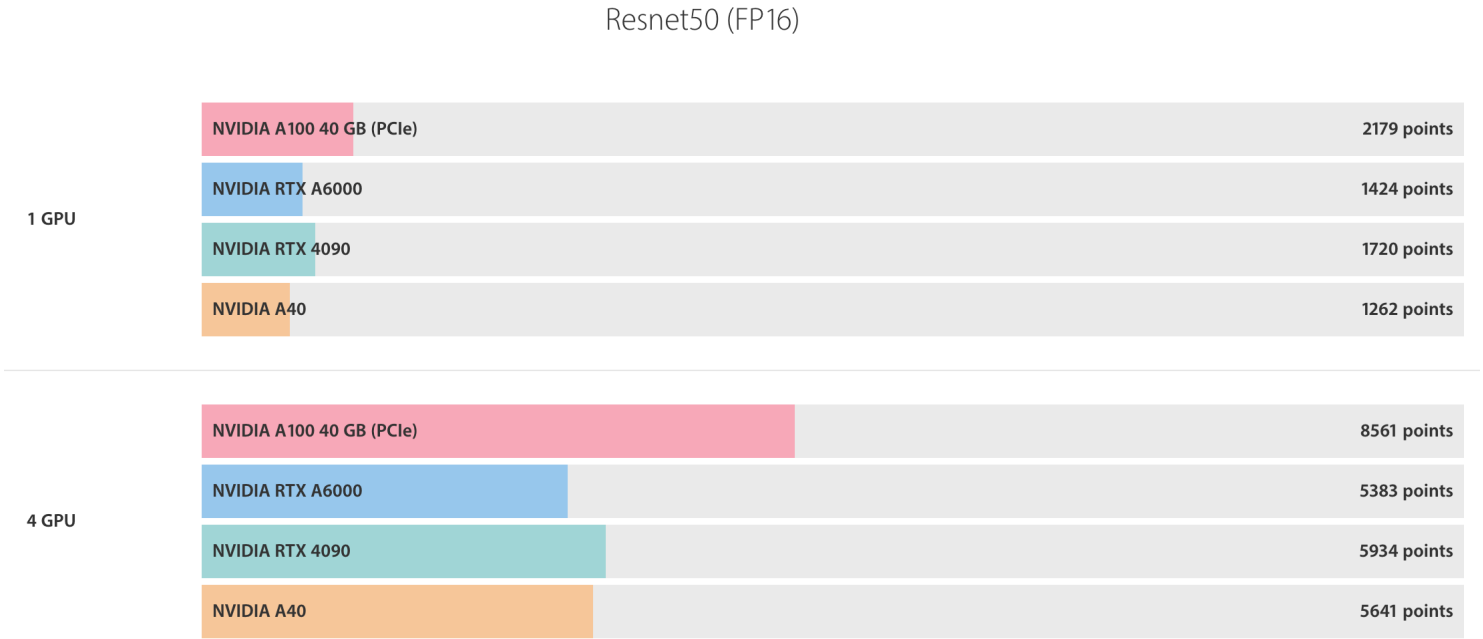
- Consider multi-tenant or public-facing environment
- With GPUDirect RDMA, the client needs to determine where to put the data
- But it cannot properly do so
  - Network latency
  - Security and privacy



(b) CPU-bypass Design.

Why accelerator scheduling is critical

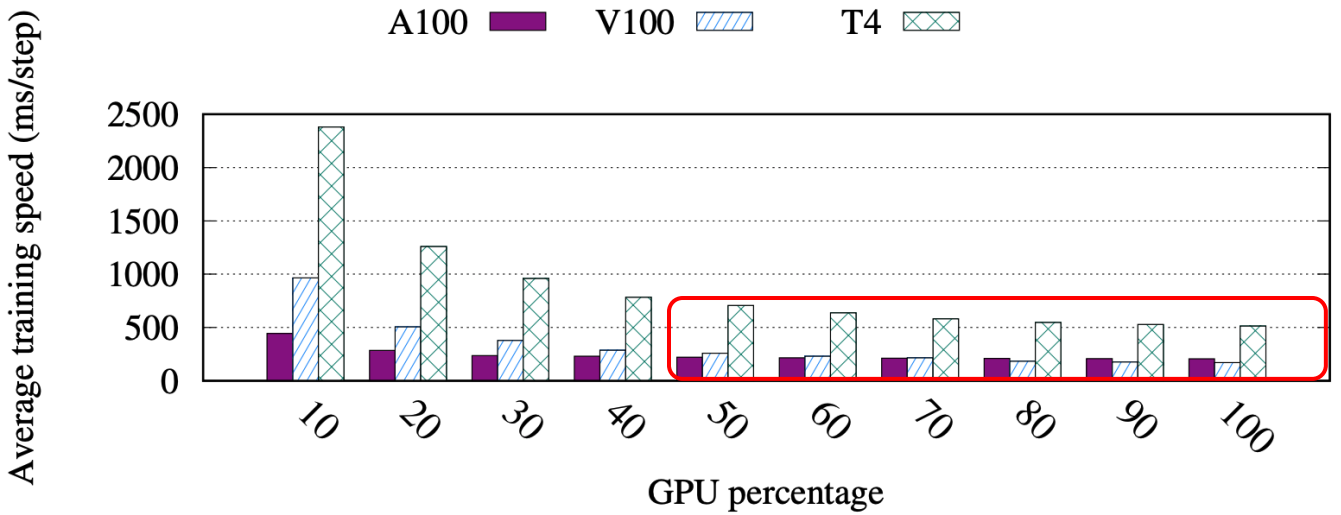
- Heterogeneity of accelerators



[2] https://www.gpu-mart.com/blog/best-gpus-for-ai-and-deep-learning-2024

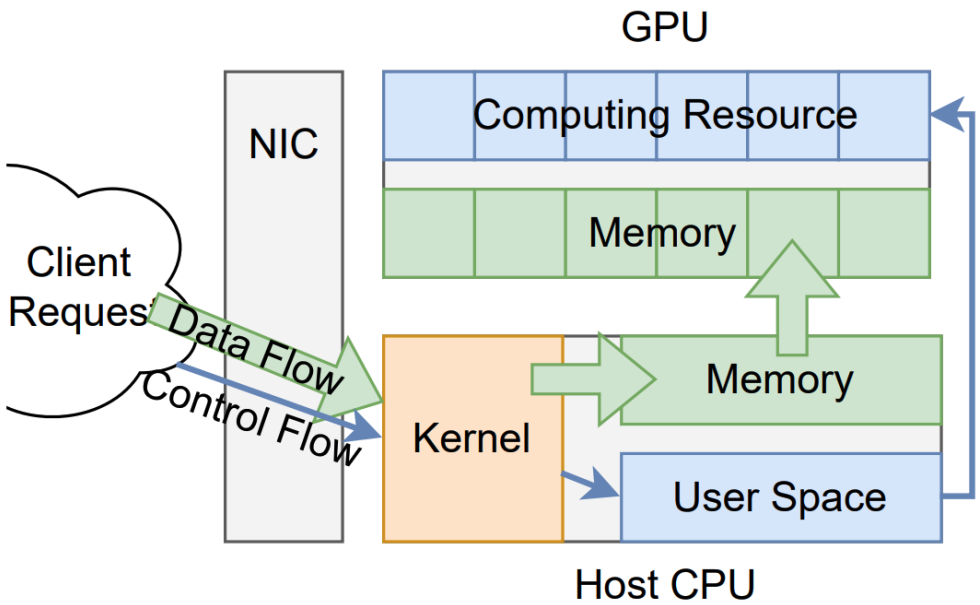
Why accelerator scheduling is critical

- Heterogeneity of accelerators
- GPU virtualization and sharing



Why accelerator scheduling is critical

- Heterogeneity of accelerators
- GPU virtualization and sharing



(c) SMIF.

[3] Cho et al., “SLA-driven ML inference framework for clouds with heterogeneous accelerators”, MLSys’22.

Can we resolve both inefficiencies at the same time?

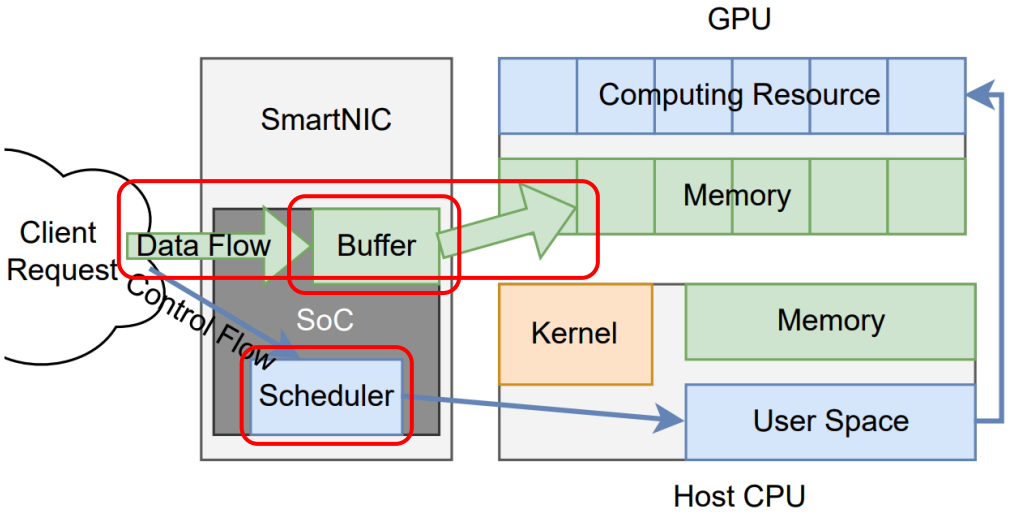
- SmartNIC comes into help
  - Off-path* SmartNIC is equipped with a general purpose SoC
  - This helps to effectively bypass host CPU while enabling accelerator scheduling

Design Option	Efficient CPU Cycle Usage	Flexible Scheduling
Client $\overset{\text{GPUDirect}}{\longleftrightarrow}$ GPU	✓	×
Client $\overset{\text{RDMA}}{\longleftrightarrow}$ CPU $\overset{\text{PCIe}}{\longleftrightarrow}$ GPU	×	✓
Client $\overset{\text{RDMA}}{\longleftrightarrow}$ SNIC $\overset{\text{DMA}}{\longleftrightarrow}$ GPU	✓	✓

System Design

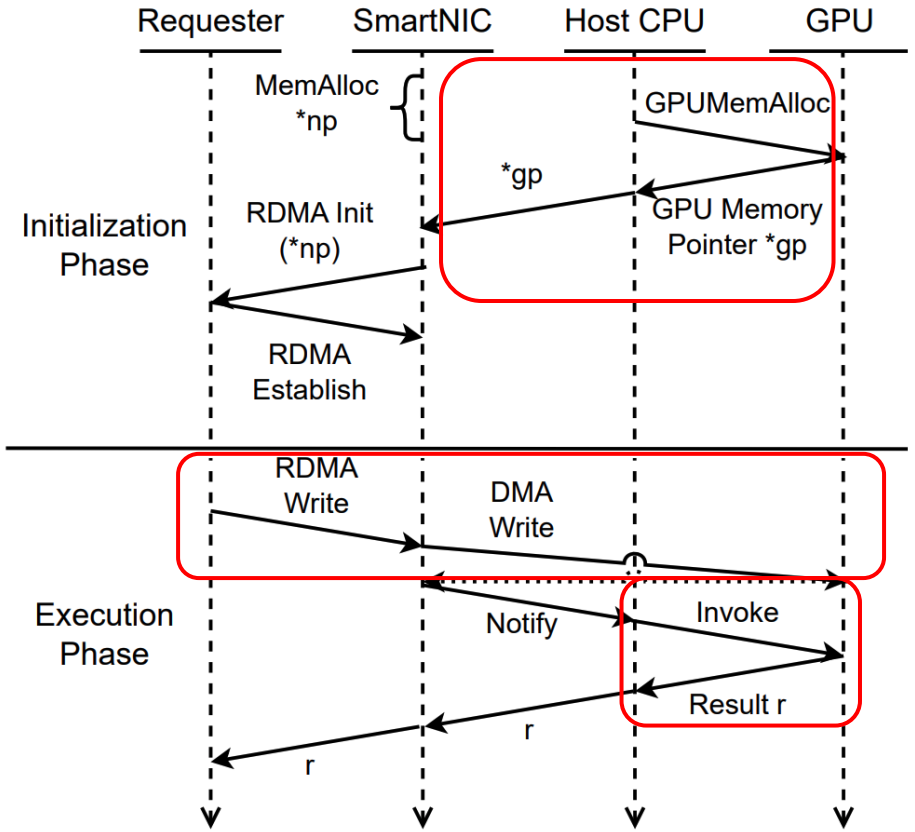
Conspirator, A SmartNIC-Aided Control Plane

- The SmartNIC SoC provisions a local buffer for incoming requests
- Scheduling decision is made at the SmartNIC SoC
- Host CPU is not involved for data transfer



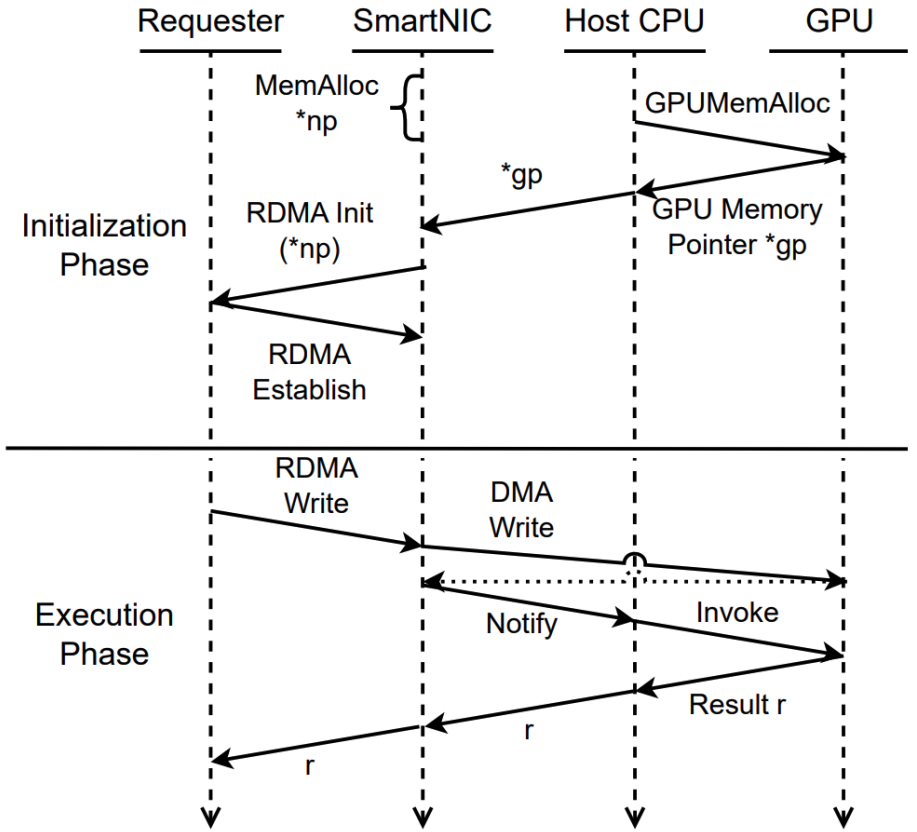
Procedure of Conspirator

- The host CPU allocates GPU memory and sends the pointers to the SmartNIC
- The client sends data to the SmartNIC, which then forwards it directly to the GPU
- During execution, the host CPU is only involved in triggering the ML execution at GPU



Procedure of Conspirator

- Result is returned through host CPU for minimizing changes on existing ML code
- This procedure is per client. In practice, SmartNIC handles concurrent requests from multiple clients



ML Scheduling on Accelerators

- Scheduling is a mixed integer linear programming problem
- We have proved that is it NP-Hard
- We propose a heuristic to approximate the optimal solution

$$\min_{x_{ijwt}} \ \varepsilon_1 \sum_j Y_j + \varepsilon_2 \sum_i m_i \delta_i \tag{1}$$

$$s.t. \ \sum_t \sum_j \sum_w x_{ijwt} = 1, \forall i \in [1,N] \tag{1a}$$

$$\sum_j \sum_w x_{ijwt} \le R_{it}, \forall i \in [1,N], \forall t \in [1,T] \tag{1b}$$

$$\sum_t \sum_i R_{it} \cdot x_{ijwt} \le \alpha_{jw} \cdot y_{jw}, \forall j \in [1,J], \forall w \in [1,W_j] \tag{1c}$$

$$\sum_w \alpha_{jw} \cdot y_{jw} \le C_j, \forall j \in [1,J] \tag{1d}$$

$$Y_j \ge \frac{\sum_w y_{jw}}{N}, \forall j \in [1,J] \tag{1e}$$

$$\delta_i = 1 - \sum_t \sum_j \sum_w x_{ijwt} \cdot k_{ijwt}, \forall i \in [1,N] \tag{1f}$$

$$A_{jt} \ge \frac{\sum_t \sum_w x_{ijwt}}{N}, \forall j \in [1,J], \forall t \in [1,T] \tag{1g}$$

$$\sum_t A_{jt} \le 1, \forall j \in [1,J] \tag{1h}$$

$$\delta_i, x_{ijwt}, k_{ijwt}, A_{jt}, y_{jw}, Y_j \in \{0,1\} \tag{1i}$$

High-Level Ideas in Scheduling

- GPU is split into fractions using MIG
- Flexible GPU fraction allocation
- Data privacy is guaranteed (following real-world demands)
- Potential migration of ongoing jobs

Conspirator Implementation

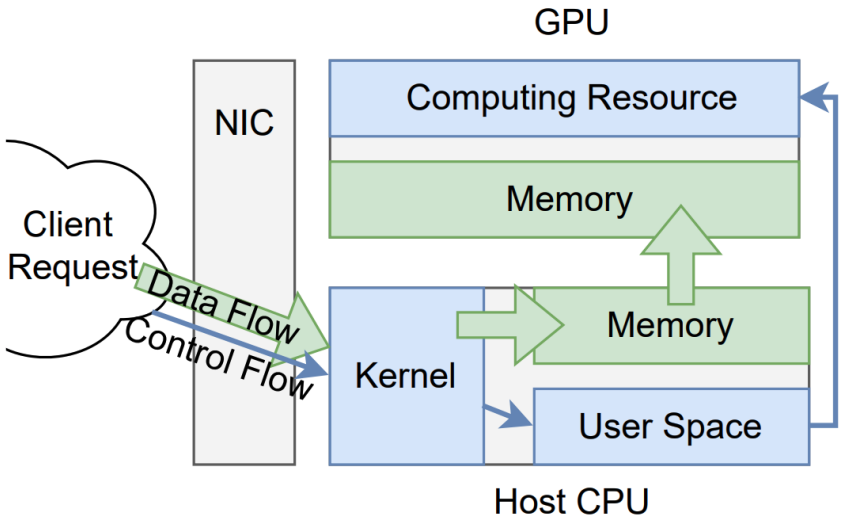
- Communication is realized using NVIDIA DOCA library and custom CUDA extension
- Allows reusing any existing ML code in Python with one line of modification: the creation of the input tensor

- Our heuristic is used for making GPU scheduling decisions

Evaluation

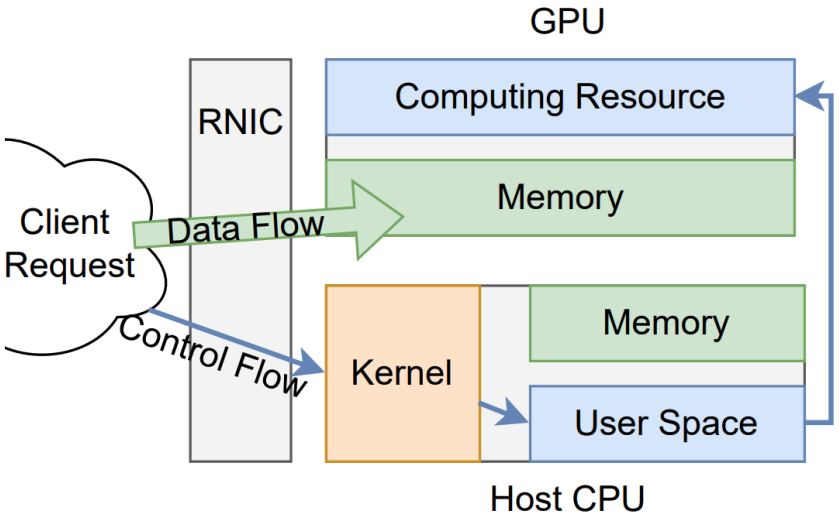
Evaluation Setup

- Testbed: Bluefield 3 SmartNIC + A100 GPU
- We compare against different architecture mentioned earlier
  ① TCP server with host CPU handling



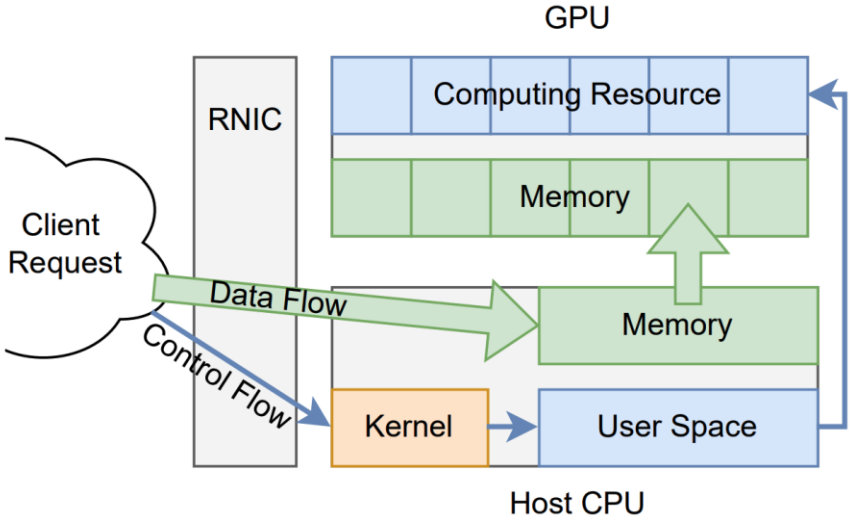
Evaluation Setup

- Testbed: Bluefield 3 SmartNIC + A100 GPU
- We compare against different architecture mentioned earlier
  - ① TCP server with host CPU handling
  - ② RDMA data to GPU memory (GPUDirect)



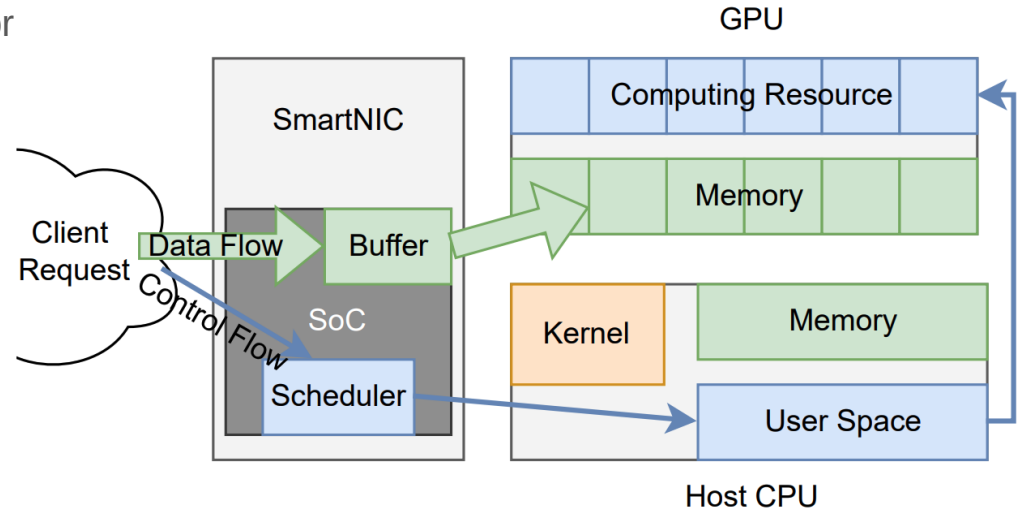
Evaluation Setup

- Testbed: Bluefield 3 SmartNIC + A100 GPU
- We compare against different architecture mentioned earlier
  - ① TCP server with host CPU handling
  - ② GPUDirect
  - ③ RDMA data to host memory (Conspirator w/o SmartNIC)



Evaluation Setup

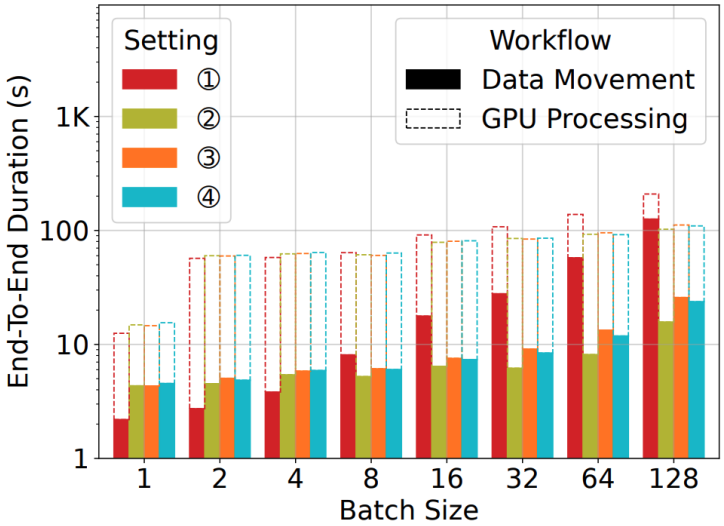
- Testbed: Bluefield 3 SmartNIC + A100 GPU
- We compare against different architecture mentioned earlier
  - ① TCP server with host CPU handling
  - ② GPUDirect
  - ③ Conspirator w/o SmartNIC
  - ④ Conspirator



① TCP server with host CPU handling
② GPUDirect
③ Conspirator w/o SmartNIC
④ Conspirator

End-to-End Duration

- Results fit intuition: ① is worst, ② is best
- But ② does not allow accelerator scheduling
- Conspirator (④) outperforms similar setting (③) without SmartNIC

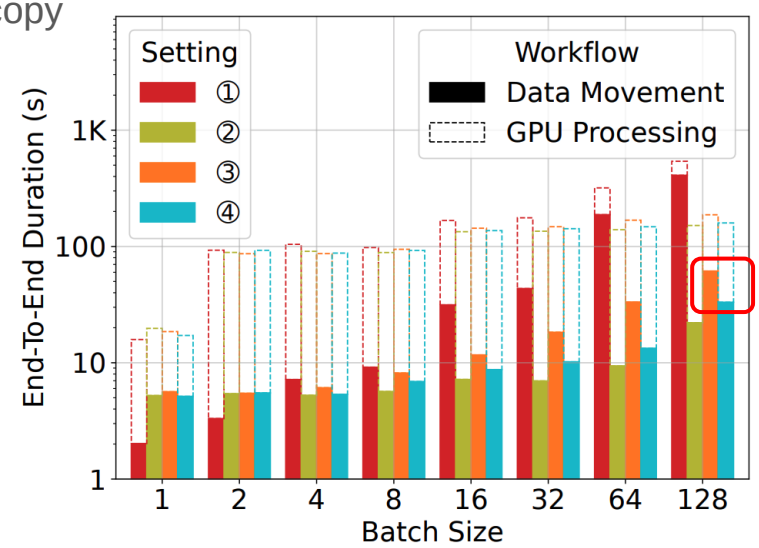


(a) 0% background CPU load.

End-to-End Duration

① TCP server with host CPU handling
② RDMA data to GPU memory
③ RDMA data to host memory
④ Conspirator

- Performance gap between ③ and ④ grows when CPU is more intense
- Benefit is realized through (check out our paper!)
  - Less CPU involvement
  - Faster local data copy



(b) 80% background CPU load.

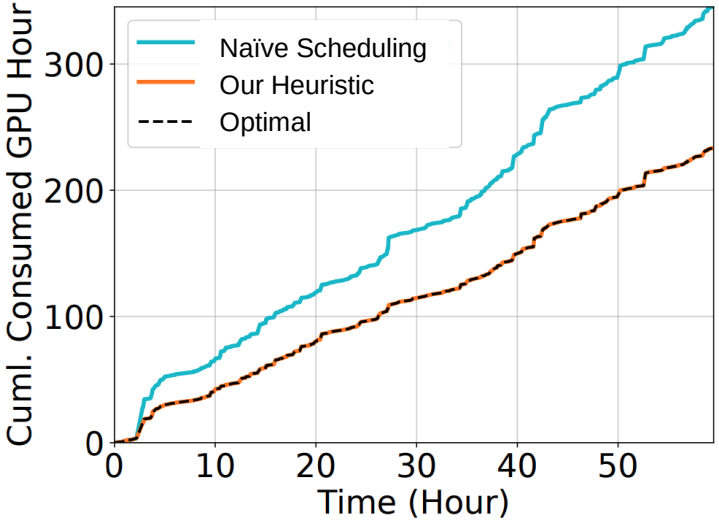
Cost Efficiency

- Conspirator promotes a more cost-effective and power-efficient system configuration

Hardware	Price (Normalized)	Power Consumption	Throughput (Normalized)	Cost-Effectiveness	Power Efficiency
Host CPU	$1,000	800W	1,000	1.0	1.25
SmartNIC	$231	150W	270	**1.17 (+17%)**	**1.8 (+44%)**

Scheduling Benefits

- Proper GPU sharing saves 33% on total consumed GPU hours
- Our heuristics achieves the same performance as optimal scheduling (Alibaba dataset)



Conclusion

- We propose Conspirator, a SmartNIC-aided control plane for optimizing distributed ML workloads
- Conspirator addresses two critical inefficiencies at the same time
  - Bottleneck on CPUs
  - Sub-optimal accelerator scheduling
- Conspirator leverages SmartNIC and is thus more cost-effective (17%) and power-efficient (44%)
- Conspirator leverages proper GPU scheduling to reduce 33% total consumed GPU hours compared to naïve scheduling

If you are interested in Hewlett Packard Labs, Talk to us!

Diman Zad Tootaghaj, email: diman.zad-tootaghaj@hpe.com
Lianjie Cao, email: lianjie.cao@hpe.com