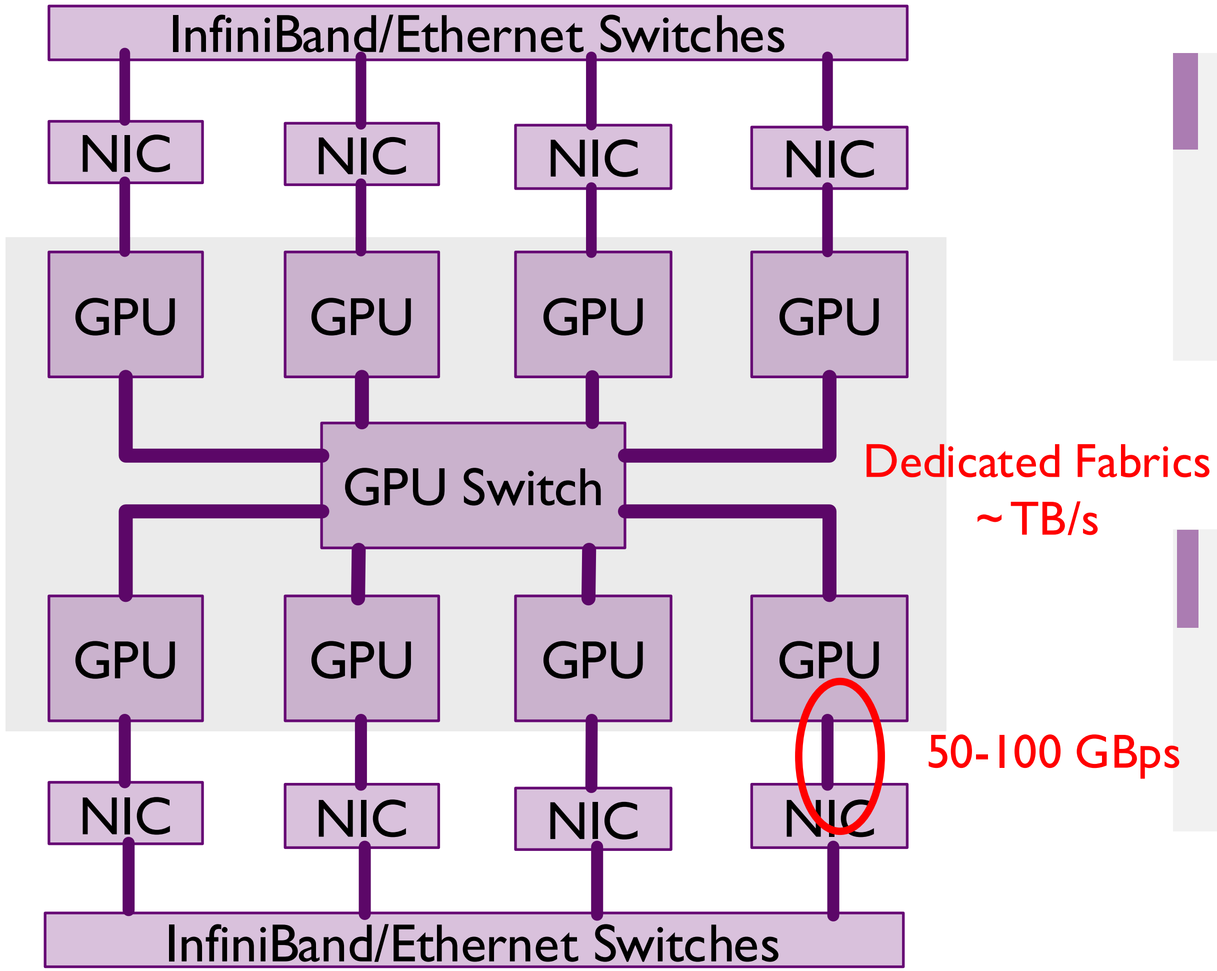
19th USENIX Symposium on Operating Systems Design and Implementation

Enabling Efficient GPU Communication over Multiple NICs with FuseLink

Zhenghang Ren, Yuxuan Li, Zilong Wang, Xinyang Huang, Wenxue Li, Kaiqiang Xu, Xudong Liao, Yijun Sun, Bowen Liu, Han Tian, Junxue Zhang, Mingfei Wang, Zhizhen Zhong, Guyue Liu, Ying Zhang, Kai Chen

Advancement of high-bandwidth GPU communication



Dedicated interconnect

- Relatively small groups
- Fast growing bandwidth

Network over general NICs

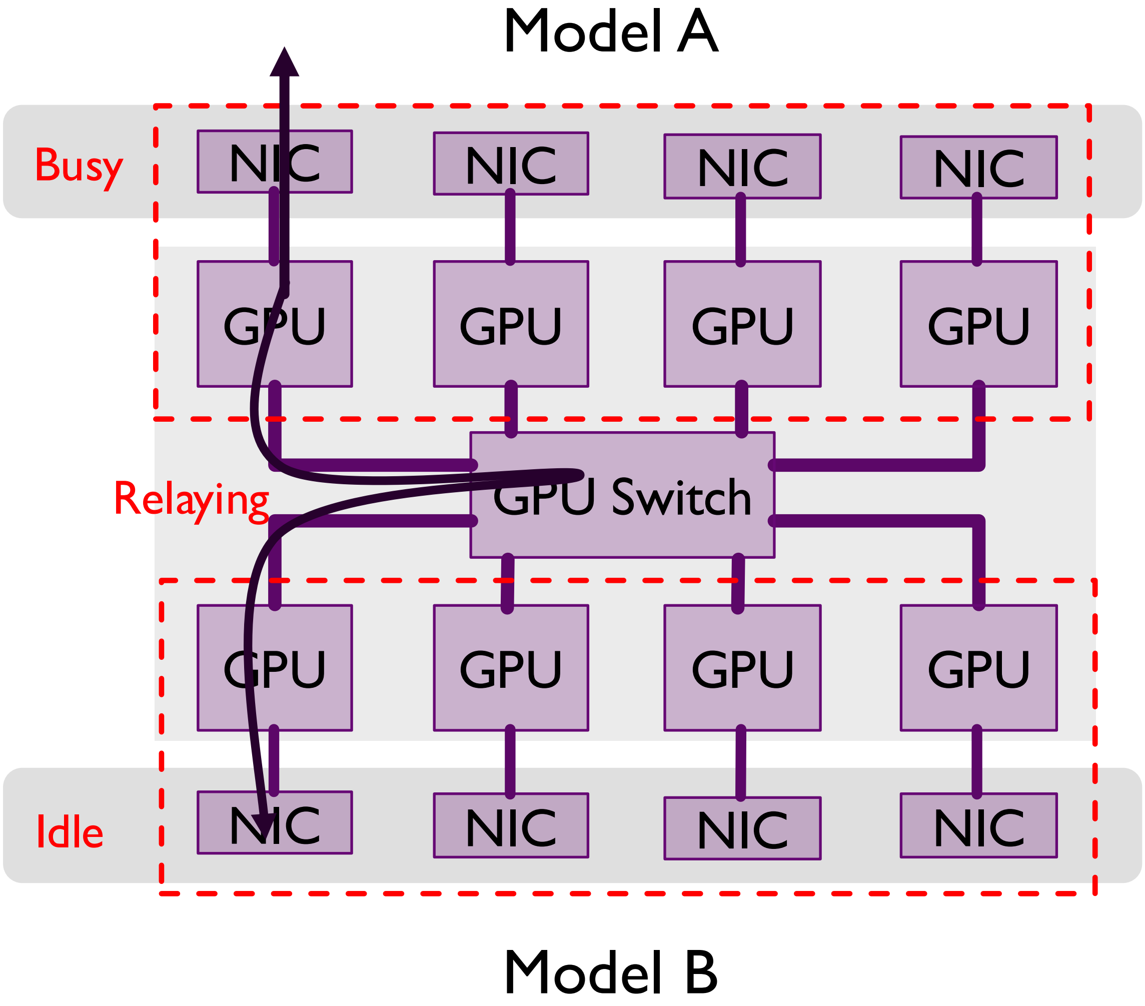
- Large scale
- Slow growing bandwidth

Can be tens of GB for every GPU!
KV Cache/Embeddings/Gradients,…

When can we multiplex the NICs and to accelerate inter-server communication?

With PCIe limitation, can we leverage **dedicated fabrics** to improve bandwidth over **general NICs**?

Observation: traffic imbalance is common



The GPU server has two independent LLM serving deployments with inter-server context (KVCache) transmission.

- Models serve different requests
- Half NICs busy and half idle.

Opportunity: multi-NIC acceleration under imbalanced traffic

Biased Traffic Volume

NICs with less traffic being idle

Using indirect NICs

Data traverse via CPU/UPI becomes bottleneck

Delayed Communication

Late GPUs increase transmission tail

Suboptimal NICs throughput

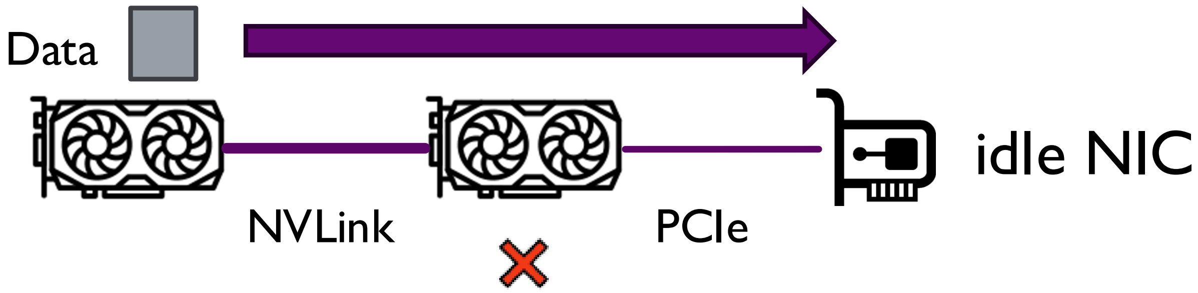
Accelerate via multi-NIC transmission

Can arbitrary ML workloads effectively leverage all available NICs, aggregating links as a "FuseLink" for inter-server communication ?

Design goals

- Efficiently use idle NICs through dedicated intra-server fabrics
- Seamless integration into existing systems (NCCL, Gloo,…)
- Avoid contention & interruption among GPUs

Challenge: incompatibility of dedicated fabrics and NICs



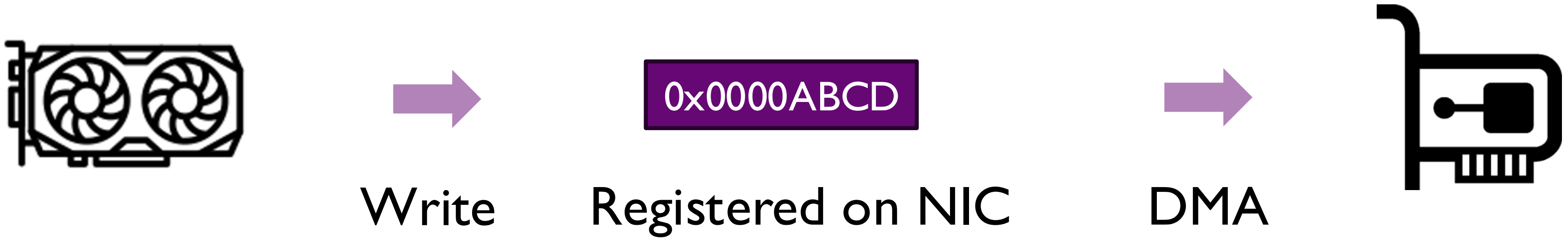
TX Direction

Hardware-offloaded NIC network stack cannot transmit data across GPU interconnects

RX Direction

NIC traffic cannot be modified to be routed across GPUs

Efficient indirect NIC communication: memory & network combined



Write

0x0000ABCD

Registered on NIC

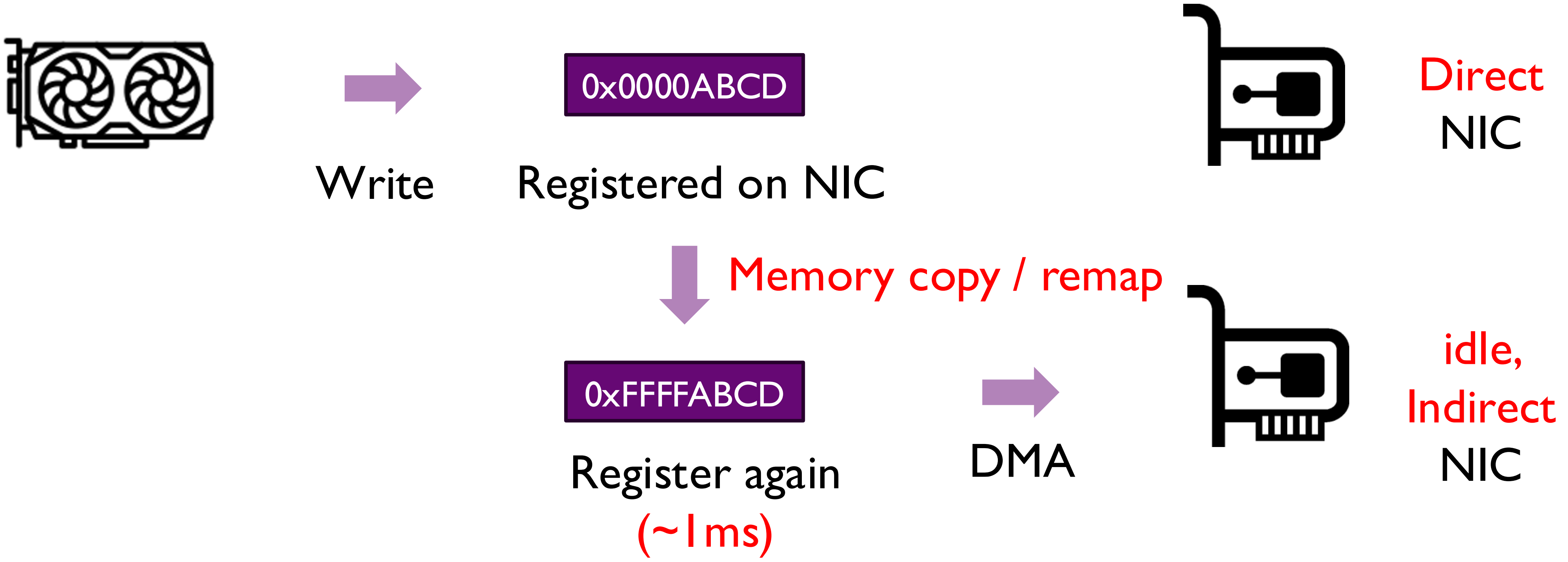
DMA

GPU program fills network buffer registered on NICs

NIC reads registered (pinned) memory through PCIe

Efficient indirect NIC communication: memory & network combined

Straight forward solution: **copy data** to intermediate GPU close to the NIC
Not performant because of extra copy and frequent CPU interruption



Write

0x0000ABCD

Registered on NIC

Direct NIC

Memory copy / remap

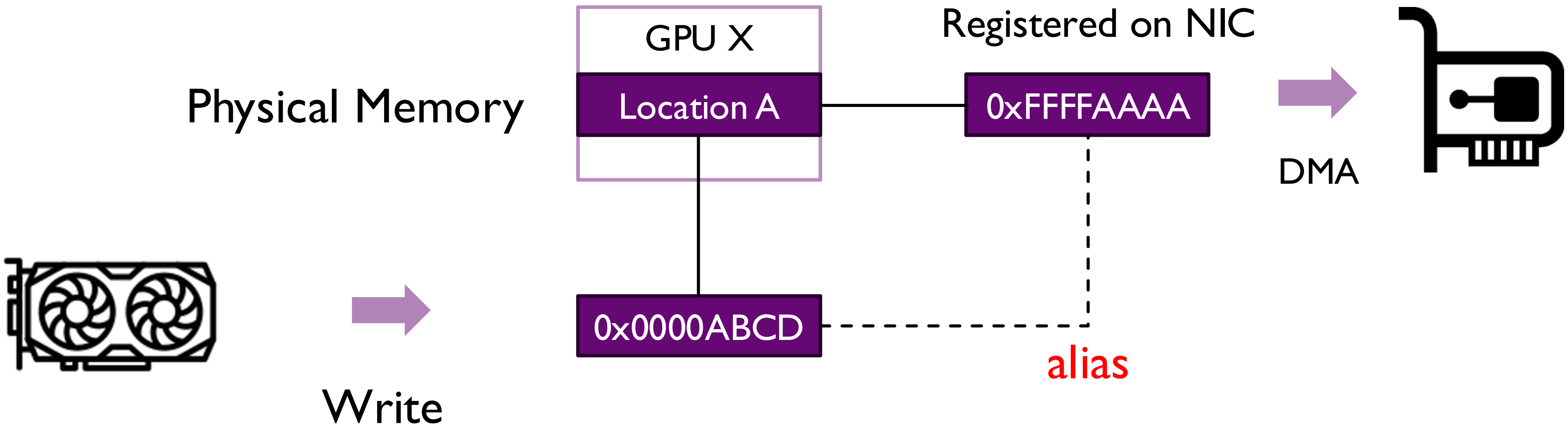
0xFFFFABCD

Register again (~1ms)

DMA

idle, Indirect NIC

Efficient indirect NIC communication: memory & network combined



GPU X

Physical Memory

Location A

Registered on NIC

0xFFFFAAAA

DMA

Write

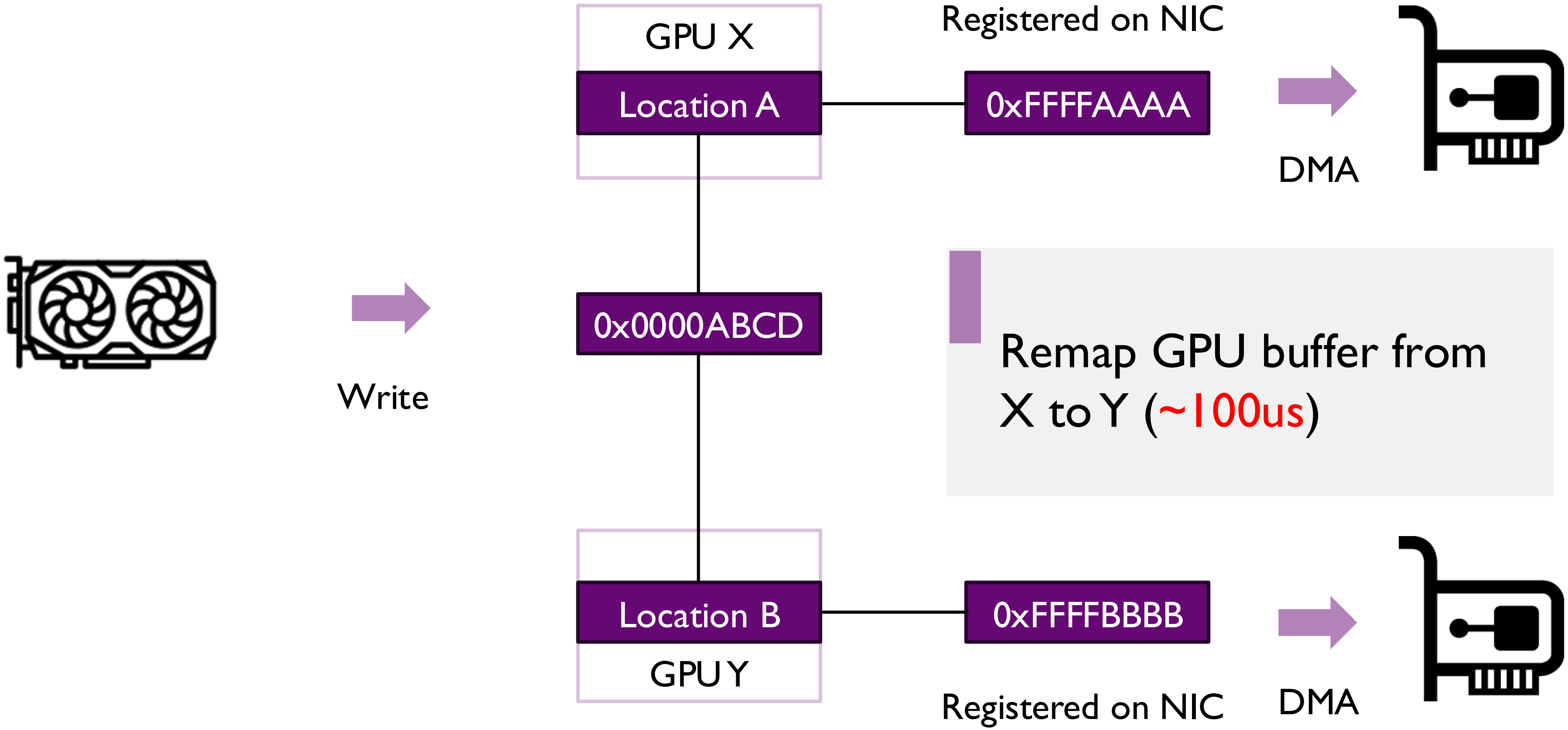
0x0000ABCD

alias

GPUs and NICs has different memory address with the same backend (alias)

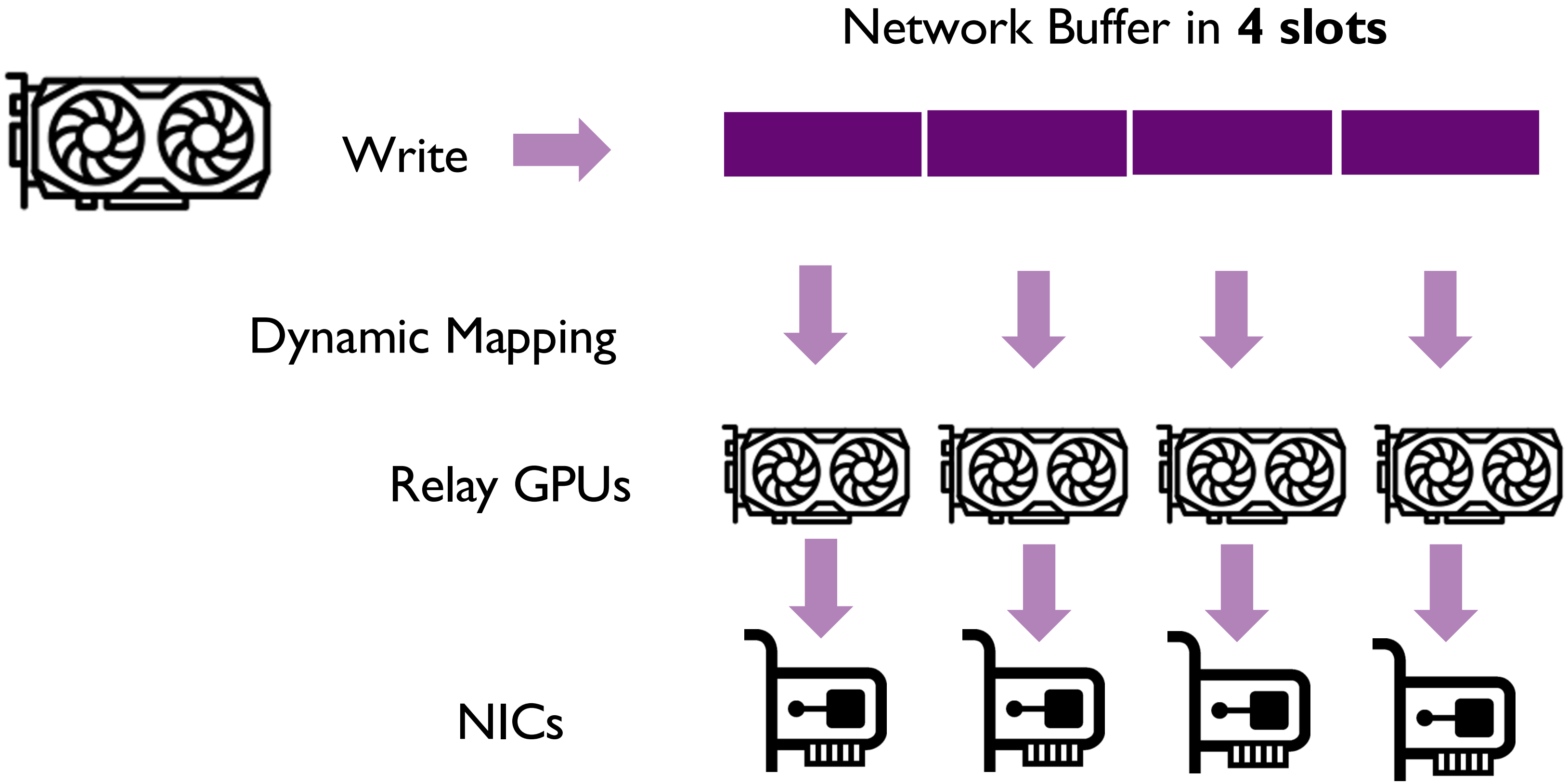
No need to change memory registration (~1ms)

Efficient indirect NIC communication: memory & network combined

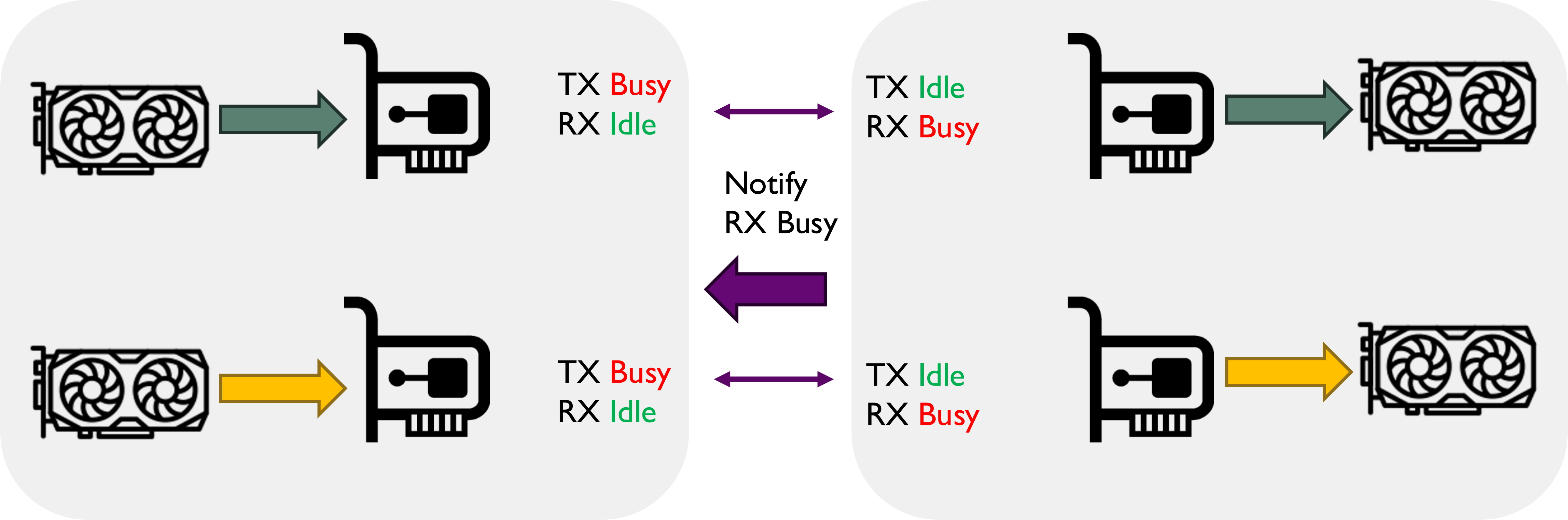


Remap GPU buffer from X to Y (~100us)

Efficient indirect NIC communication: memory & network combined

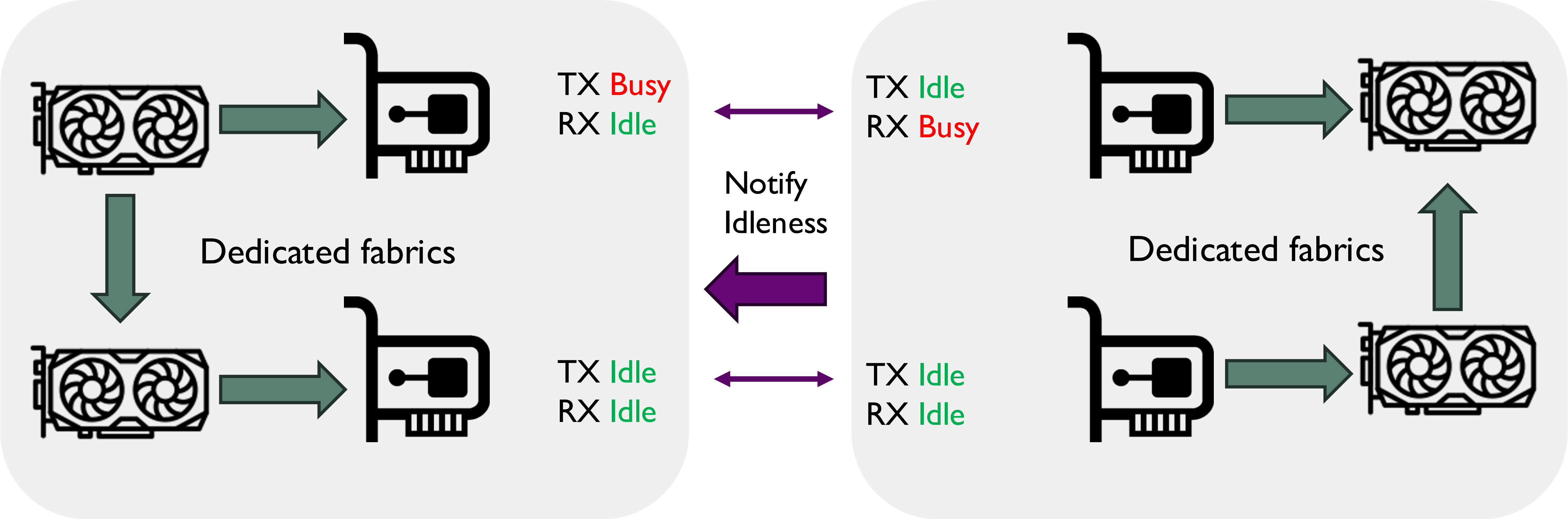


Dynamic load balance without contention & interruption



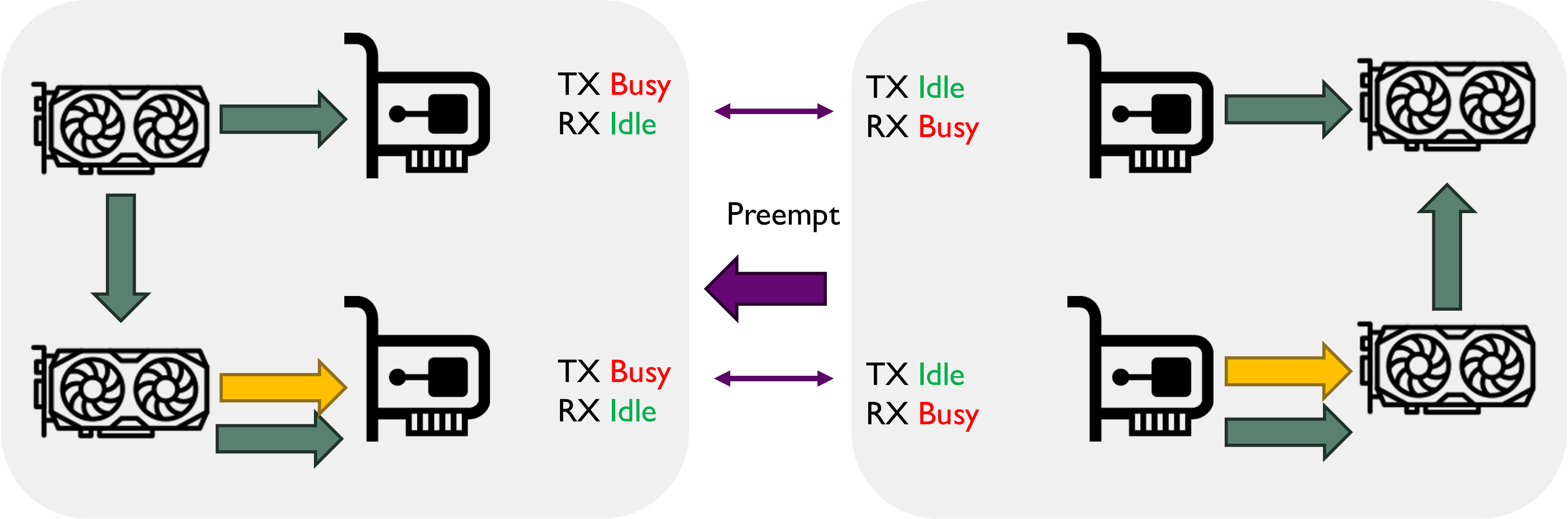
- Record NIC TX/RX workload status within each server
- Exchange TX/RX status, agree on NIC selection

Dynamic load balance without contention & interruption



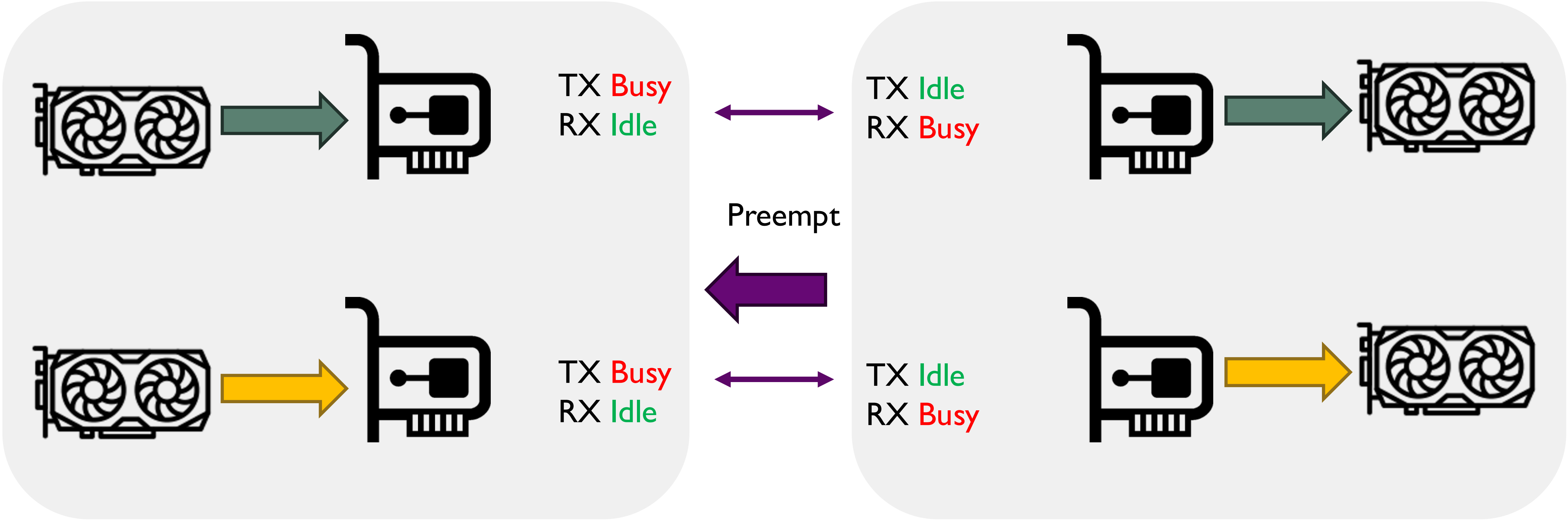
- RX & TX being idle
- Using two NICs through intra-server relaying

Dynamic load balance without contention & interruption



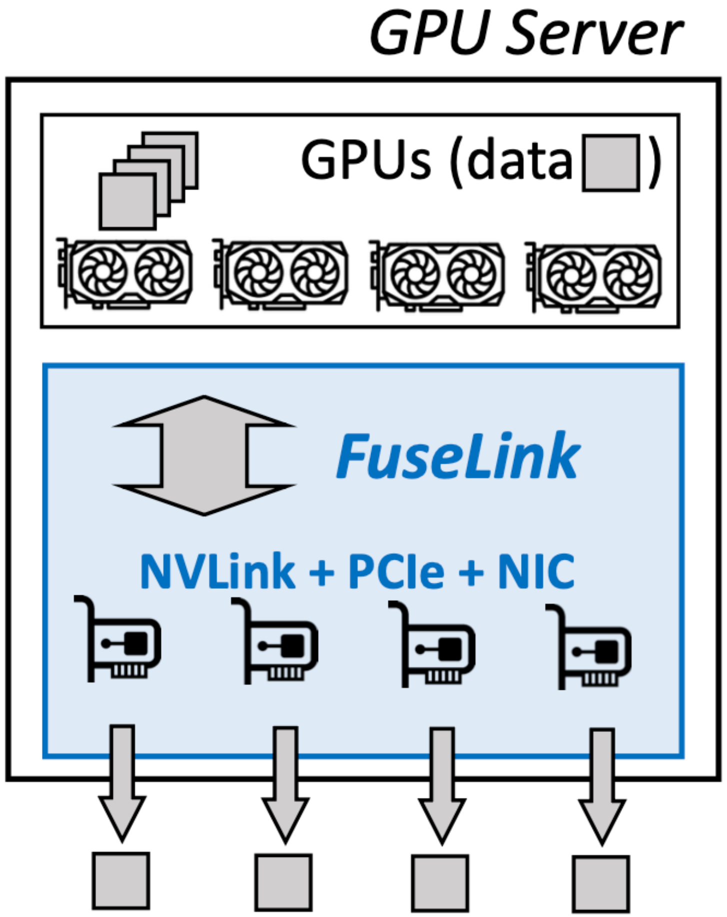
- NIC contention detected
- Preempt relay traffic and fallback to direct NIC

Dynamic load balance without contention & interruption



- NIC contention detected
- Preempt relay traffic and fallback to direct NIC

System Overview

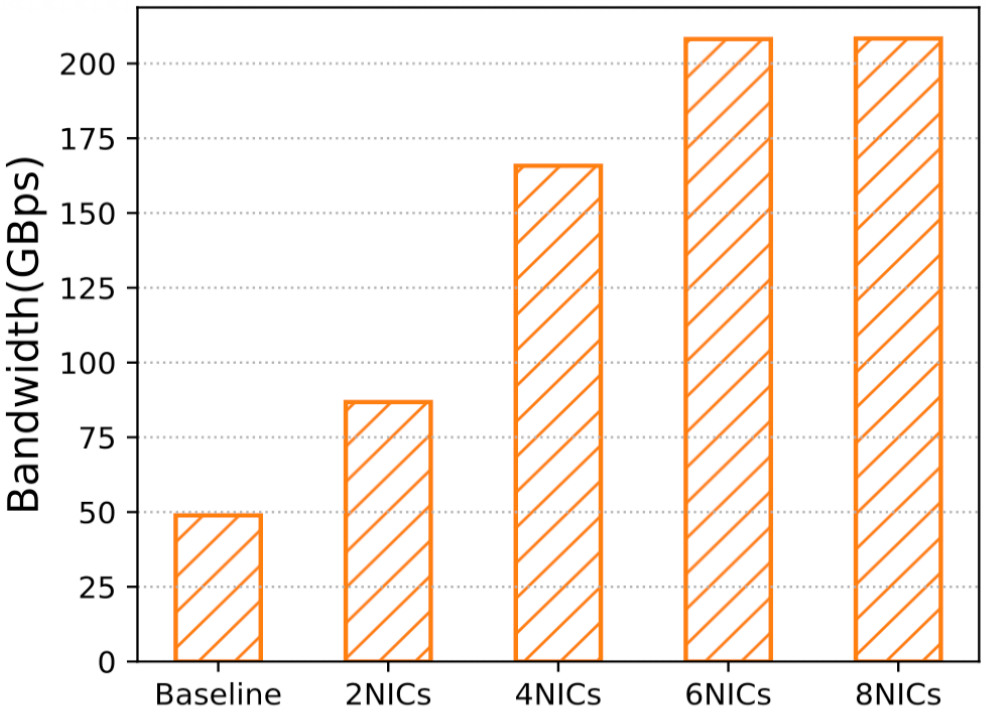


FuseLink system components

- **NIC idleness detection**: mark NICs as idle when nothing is being sent/received.
- **NVLink + NIC transport**: send traffic to intermediate GPUs and transmit through idle NICs efficiently.
- **Contention elimination**: ensure GPUs fully occupy direct NICs during communication.

Evaluation: FuseLink bandwidth over NVLink + NIC

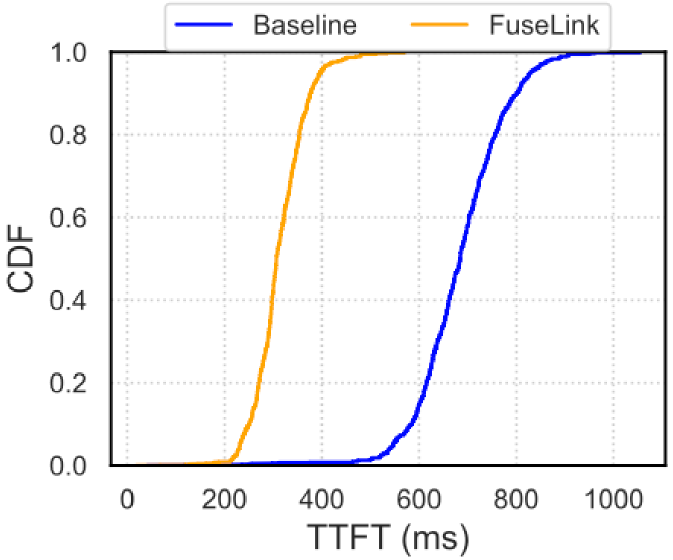
Each server: eight GPU, eight-lane NVLink (~160 GBps) and 400 Gbps (50 GBps) NICs.



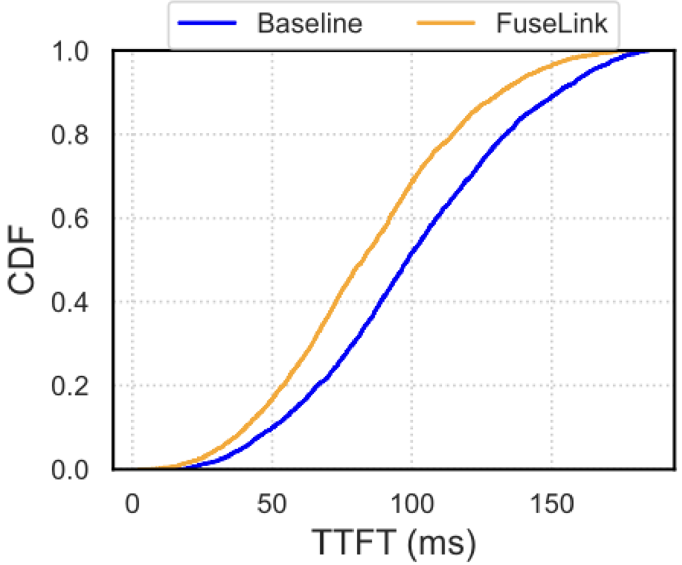
FuseLink bandwidth for two GPUs using different number of NICs

Bandwidth reaches limit when both direct NIC and NVLink are fully utilized.

Evaluation: LLM serving of independent instances



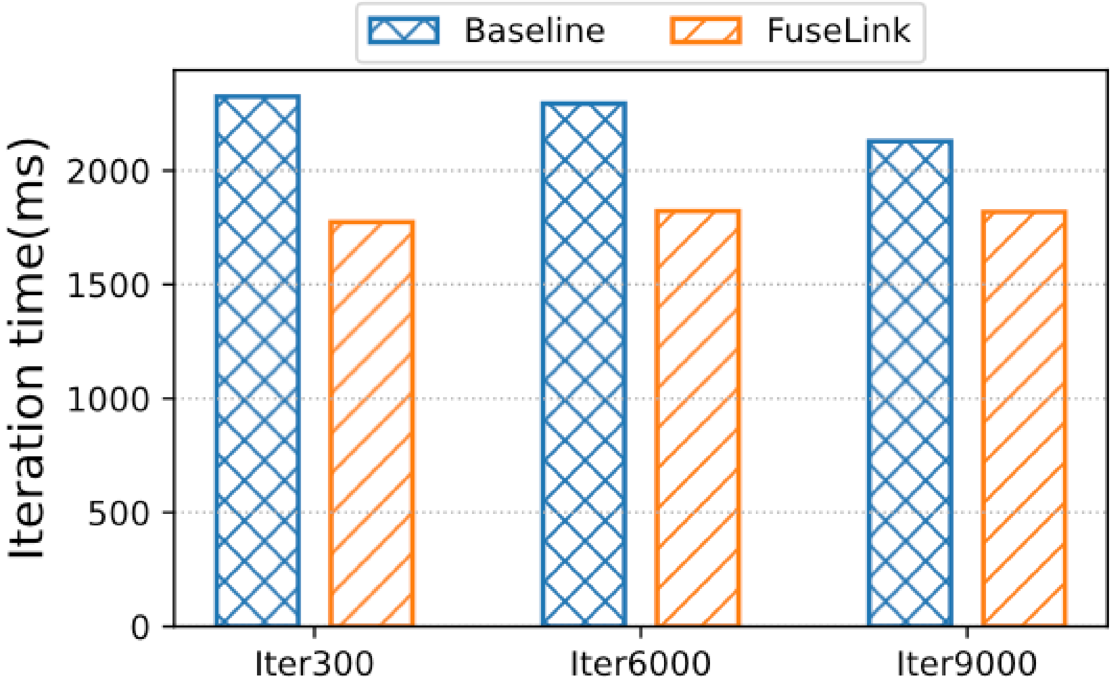
Eight serving instances

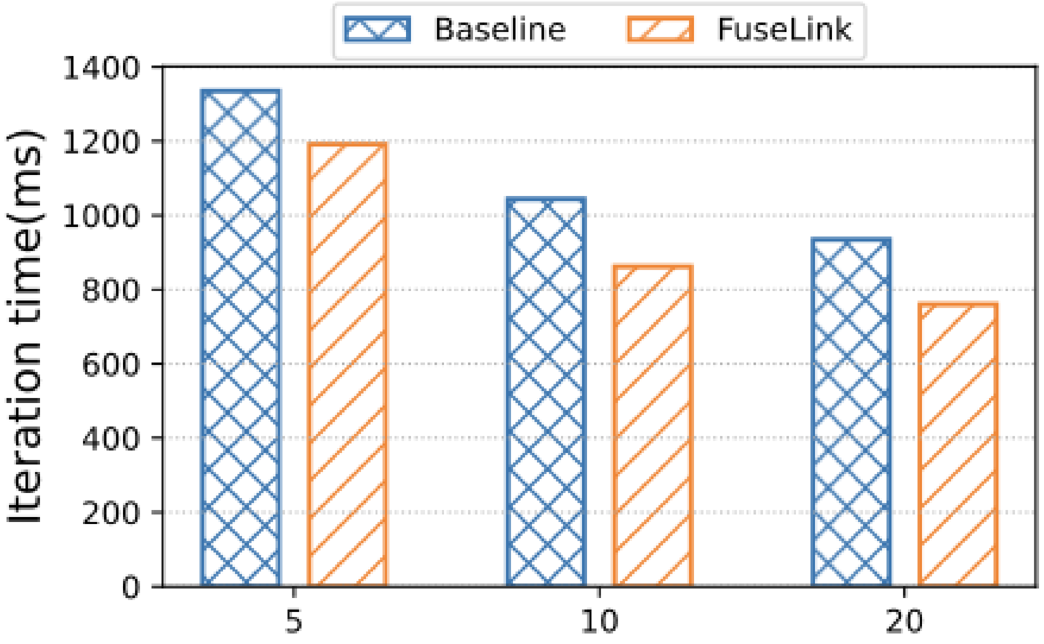


Two serving instances

Improvement: accelerated data transfer & reduced waiting time

Evaluation: expert-parallelism and imbalanced embedding transmission





Each server has two expert shards
Accelerating imbalanced all-to-all in Mixtral 8x22B expert-parallel training

Accelerating imbalanced embedding transmission when training DLRM

Limitations and future works

- Appliable to other GPUs?
  Yes, only need P2P memory access & virtual memory mapping feature.

- Fine-grained load balancing?
  Per-chunk load balancing to per-packet load balancing.

Thank You!

Code: https://github.com/axio-project/FuseLink
Contact: zrenak@cse.ust.hk