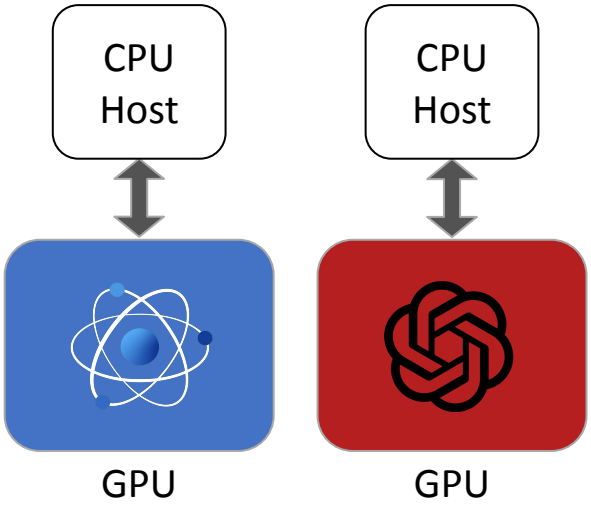
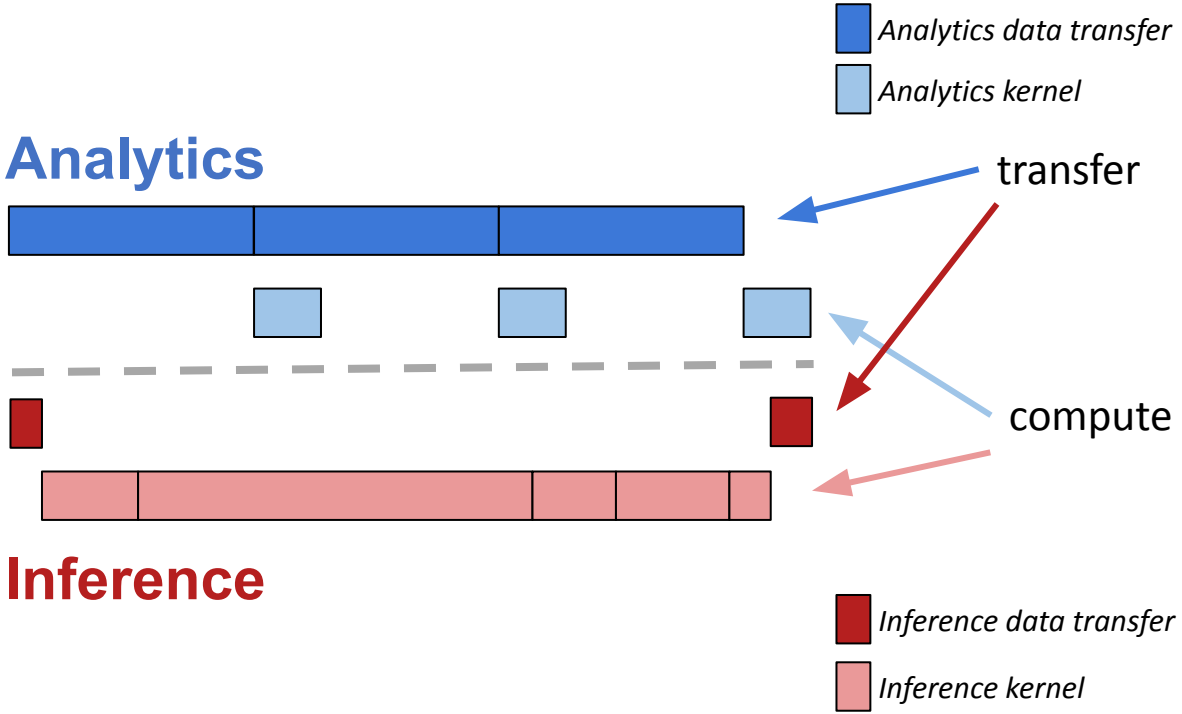
Data-Movement Aware GPU Sharing for Data-Intensive Systems

Yi Jiang, Hamish Nicholson, Viktor Sanca, Anastasia Ailamaki

CIDR, Chaminade, 01/20/2026

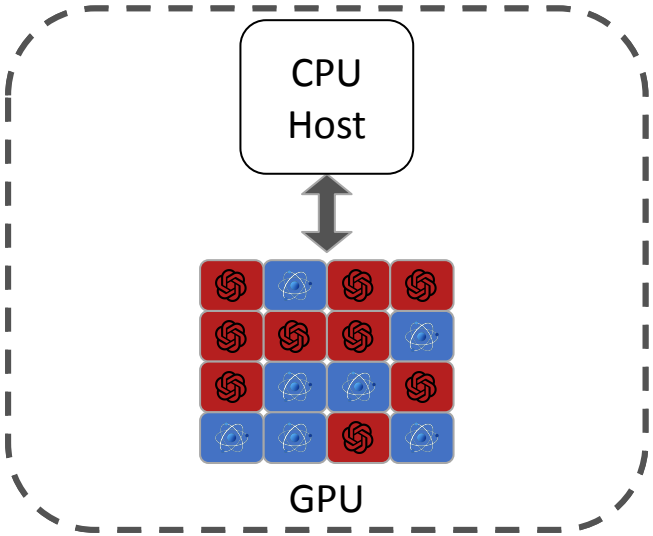
Dedicated GPUs waste compute and interconnect BW



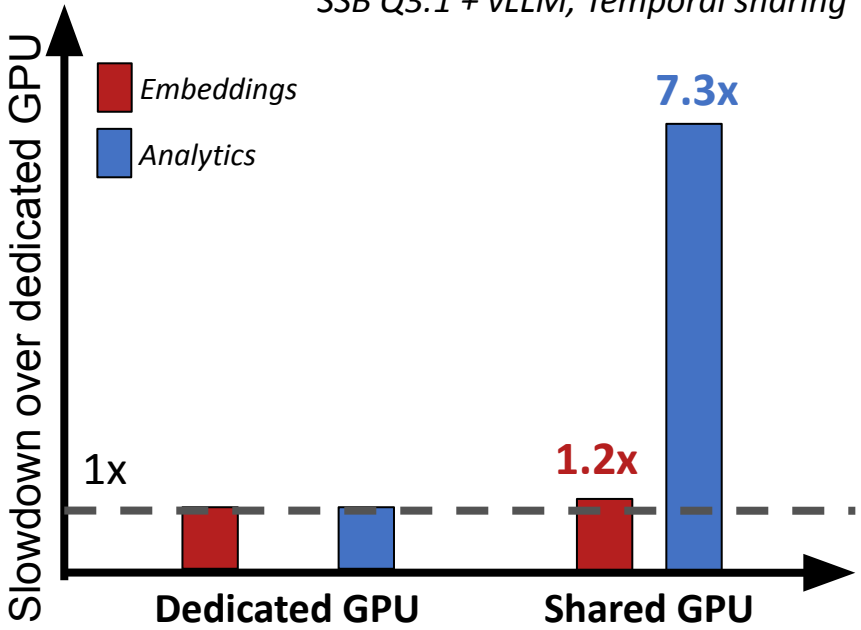


Idle interconnect during kernels; idle compute during transfers

GPU: sharing hurts performance

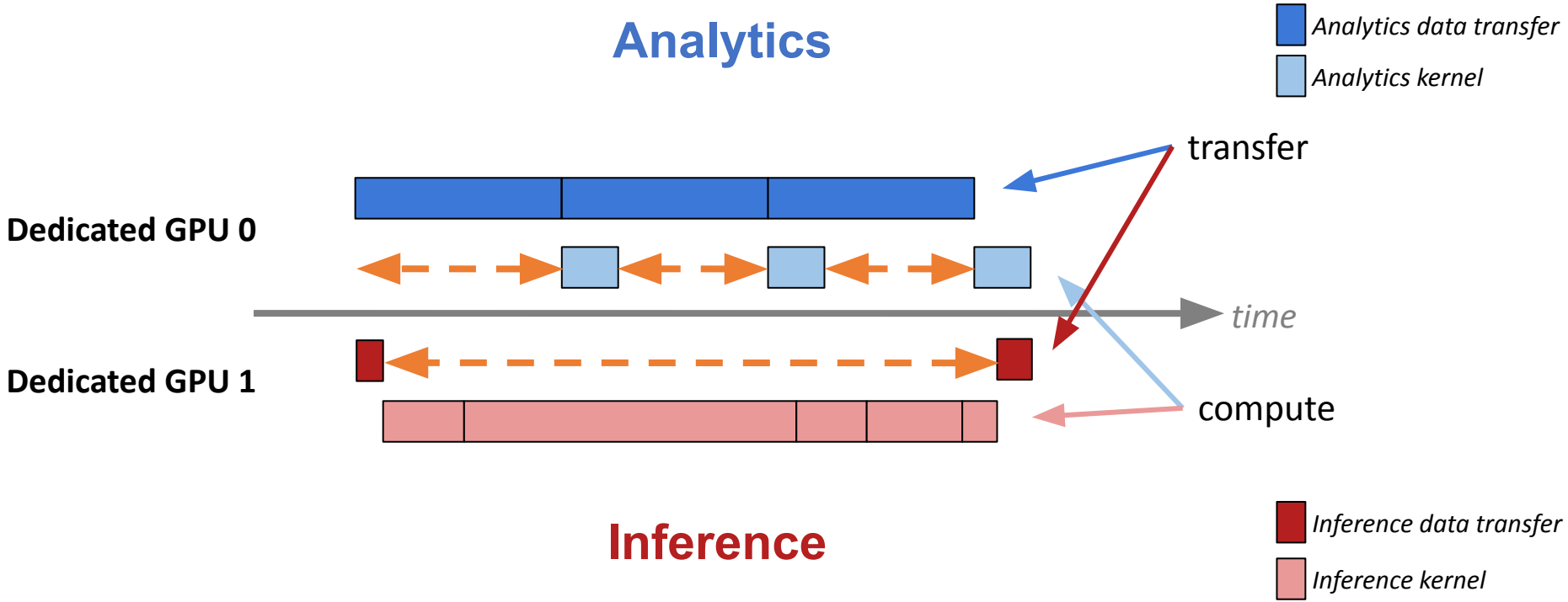


Azure Standard_NC24ads_A100_v4 instance
NVIDIA A100, 16x PCIe 4.0
SSB Q3.1 + vLLM, Temporal sharing



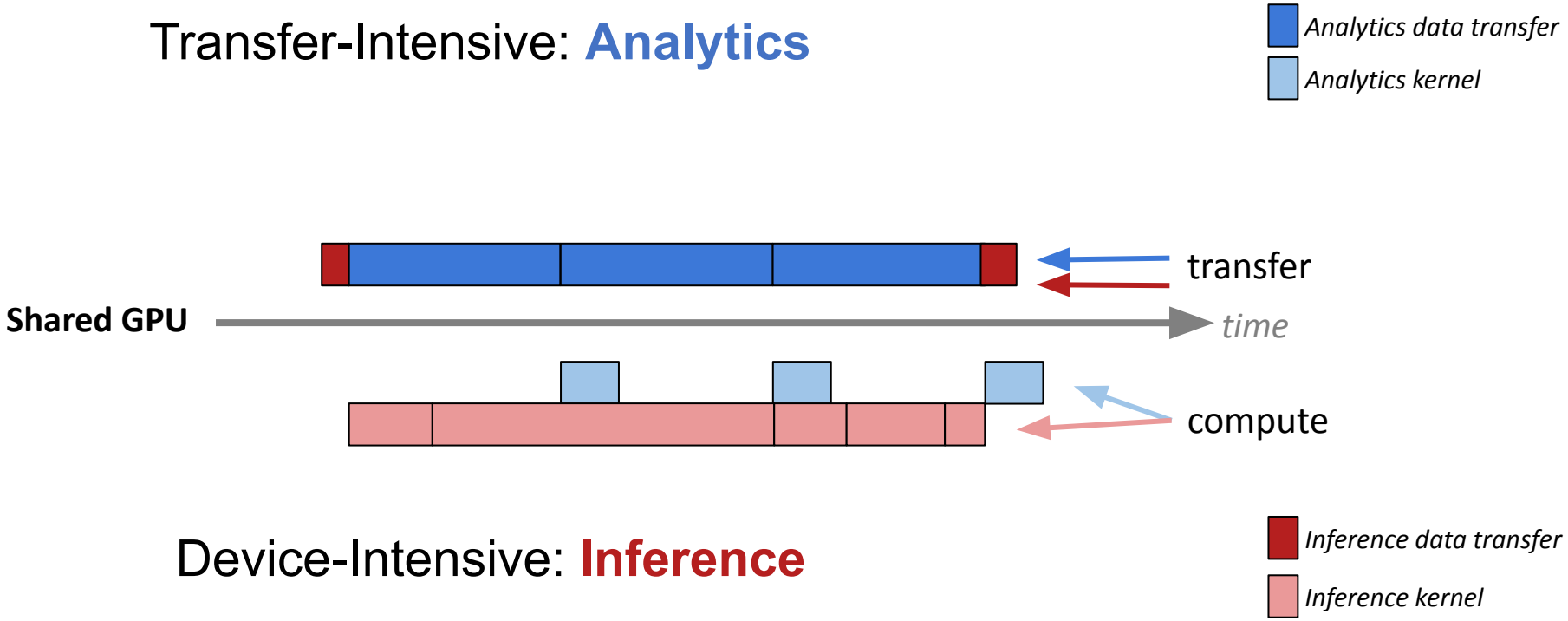
Minimize performance drops through resource-aware sharing

Hybrid workloads have complementary resource patterns



Hybrid workloads have complementary resource patterns

Transfer-Intensive: **Analytics**



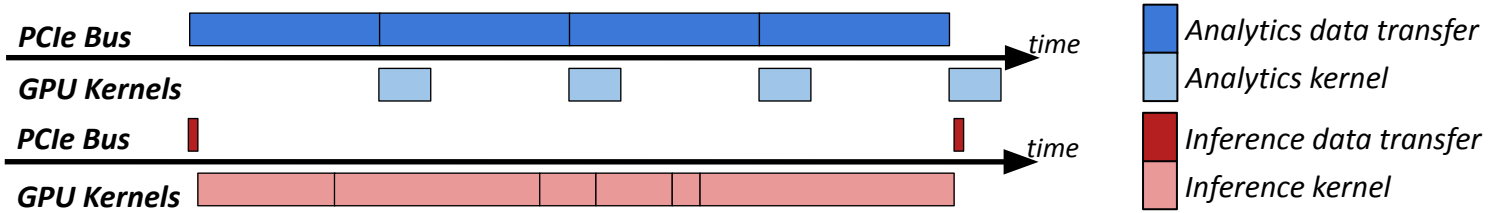
Device-Intensive: **Inference**

One workload's idle resource is another's opportunity

GPU sharing in practice

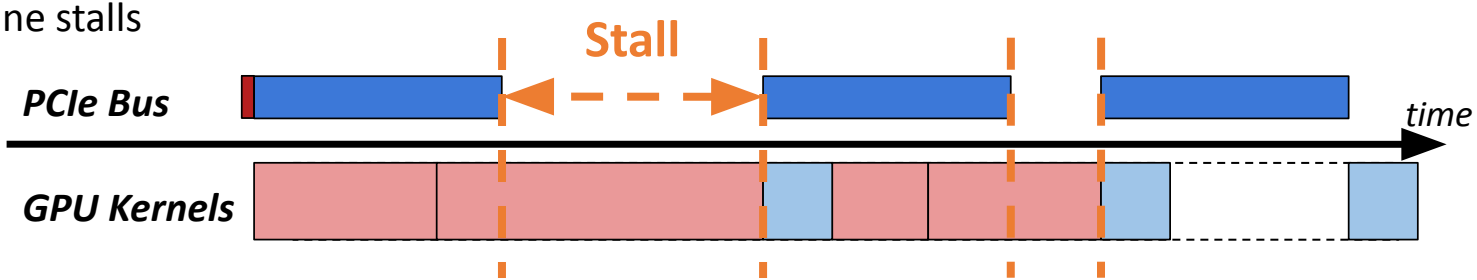
Analytics only

Inference only



Temporal sharing

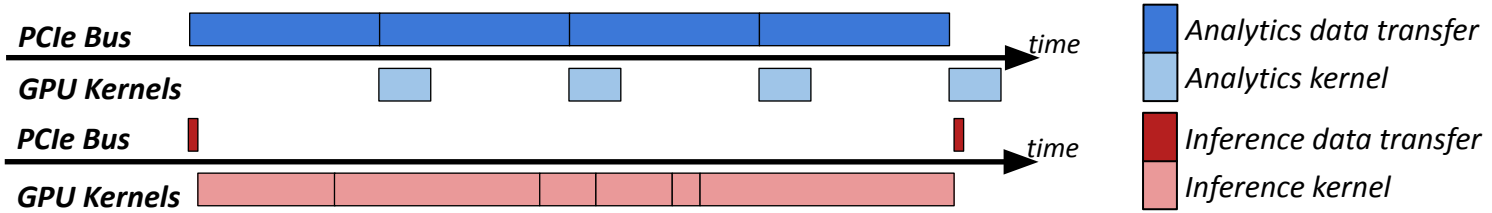
Pipeline stalls



GPU sharing in practice

Analytics only

Inference only

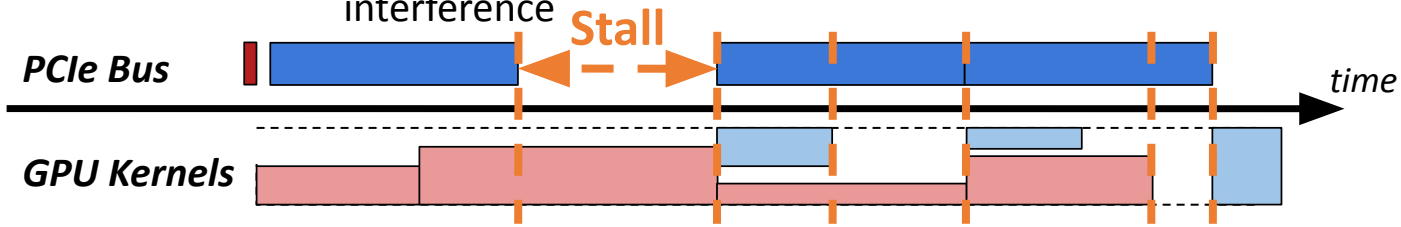


Temporal sharing

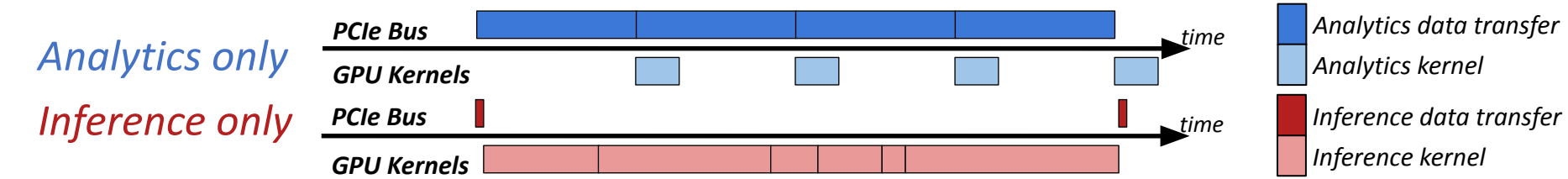
Pipeline stalls

Spatial sharing

Uncontrolled interference



GPU sharing in practice



Temporal sharing

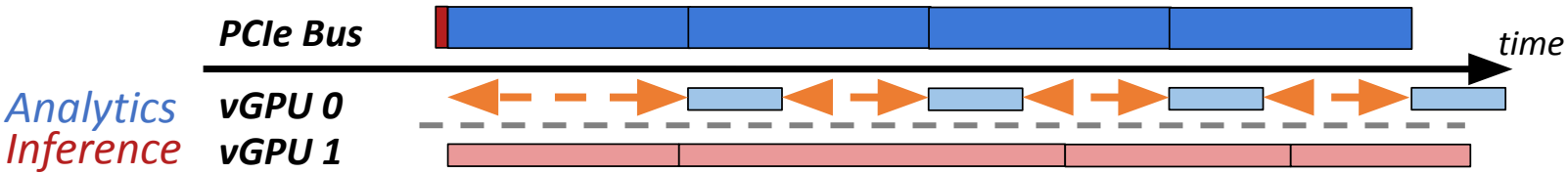
Pipeline stalls

Spatial sharing

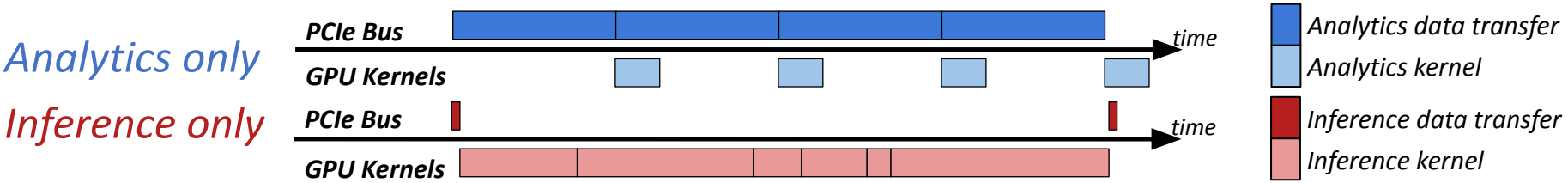
Uncontrolled interference

Physical partitioning

Static partitioning



GPU sharing in practice

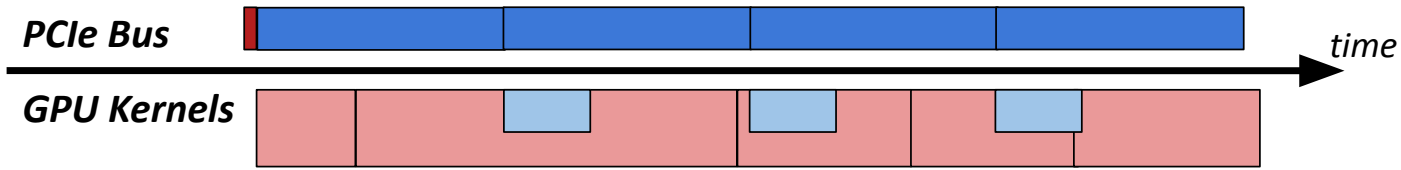


Temporal sharing
Pipeline stalls

Spatial sharing
Uncontrolled interference

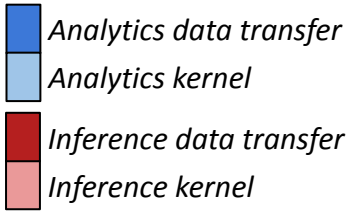
Physical partitioning
Static partitioning

Ideal sharing
Dynamic & concurrent
Minimize stalls



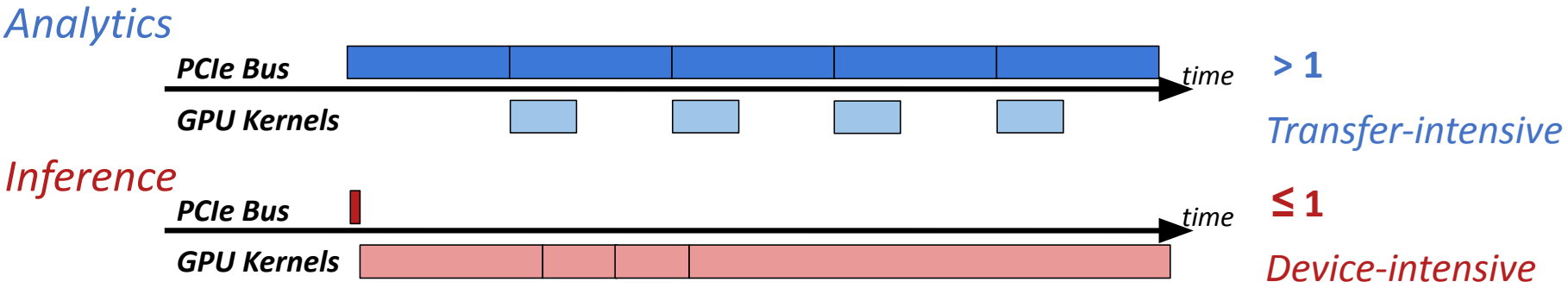
Isolating stalls for on-device *vs* interconnect resources

GPU Sharing: the interconnect as a first-class resource



- Online workload characterization

$$\text{Interconnect intensity} = \frac{T_{\text{data\_transfer}}}{T_{\text{kernel}}}$$



Interconnect intensity is a runtime measure of workload bottleneck

Dynamic GPU sharing at runtime

Monitor data transfers and GPU kernel execution

- Dynamic adaptation

  Transfer-intensive workloads: prioritize to saturate interconnect bandwidth

  Device-intensive workloads: fill in available GPU cycles

CUDA API call interception

CUDA events and CUPTI:

memory management operations

kernel launch operations

Prototype scheduler: LD_PRELOAD

No need for workload level changes

Experimental setup

Software

Proteus: GPU-accelerated query engine
vLLM v0.8.5: inference framework

Workloads

Analytics: Star Schema Benchmark (SF 100)
Inference - embedding generation models
- *small*: multilingual-e5-large (560M parameters, 1GB GPU memory)
- *large*: BGE-Multilingual-Gemma2 (9.2B parameters, 17.2GB GPU memory)

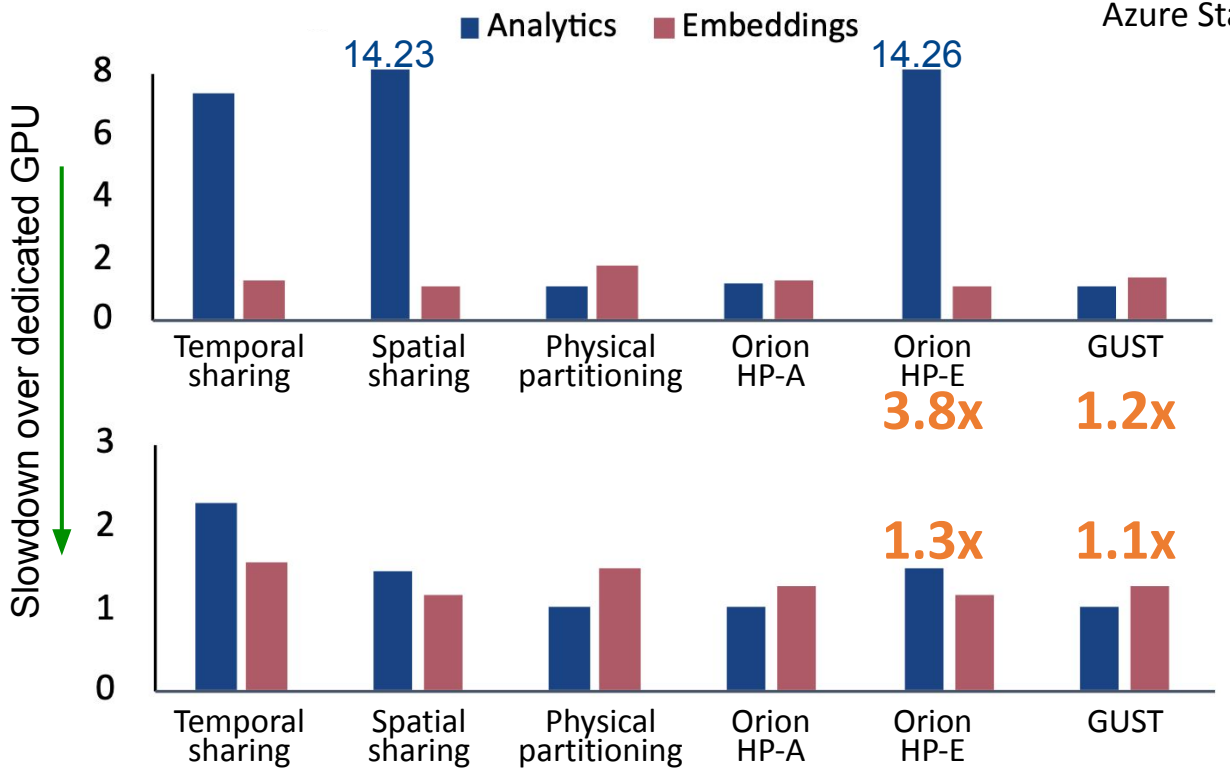
Baselines

Default time slicing	MPS	MIG	Orion
Temporal sharing	Spatial sharing	Physical partitioning	Interference-aware spatial sharing

Nvidia, F Strati EuroSys'24

GUST: GPU Unified Sharing with Transfer-awareness

Azure Standard_NC24ads_A100_v4 instance
NVIDIA A100, 16x PCIe 4.0
SSB Q3.1 + Embed



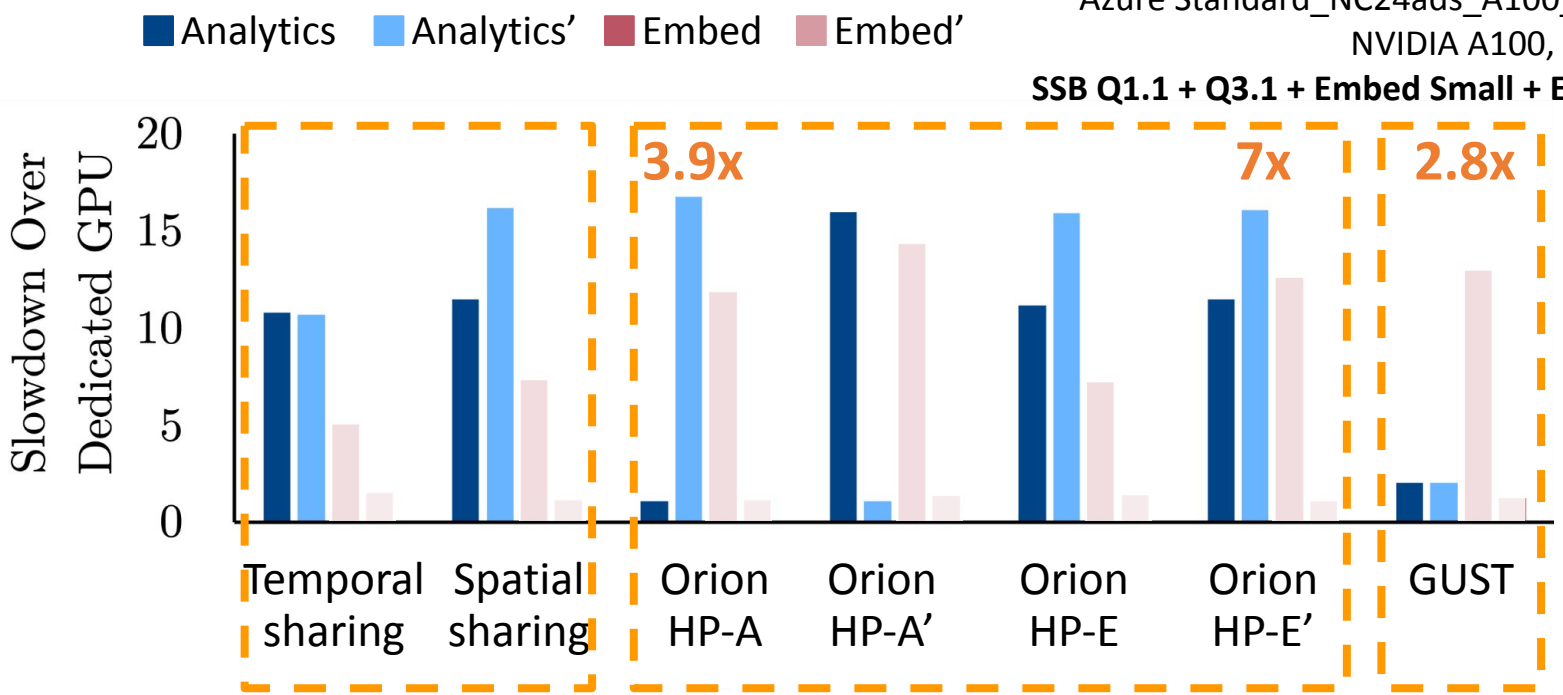
large

small

Effective GPU sharing for heterogeneous tasks requires transfer-awareness

GUST under high contention: 4-workload colocation

Azure Standard_NC24ads_A100_v4 instance
NVIDIA A100, 16x PCIe 4.0
SSB Q1.1 + Q3.1 + Embed Small + Embed Large



GUST enables efficient multiplexing of workloads with diverse resource requirements

GPU sharing must be data transfer aware

- **GPU sharing doesn't solve underutilization**
  - Idle compute during transfers; idle interconnect during kernels
  - Actively schedule data- & device-intensive work to fill gaps
- **Interconnect intensity enables dynamic adaptation**
  - Online characterization to classify workloads
- **Transfer-aware sharing reduces resource stalls**
  - Improves slowdown relative to dedicated GPUs from 3.9-7× to 2.8×

+ enable same level of sharing control as CPU

+ unified host-device resource sharing

Thank you!