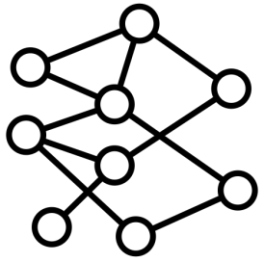
IDYLL: Enhancing Page Translation in Multi-GPUs via Light Weight PTE Invalidations

Bingyao Li[1], Yanan Guo[1], Yueqi Wang[1], Aamer Jaleel[2], Jun Yang[1], Xulong Tang[1]

[1]University of Pittsburgh, [2]NVIDIA

Multi-GPU is Popular

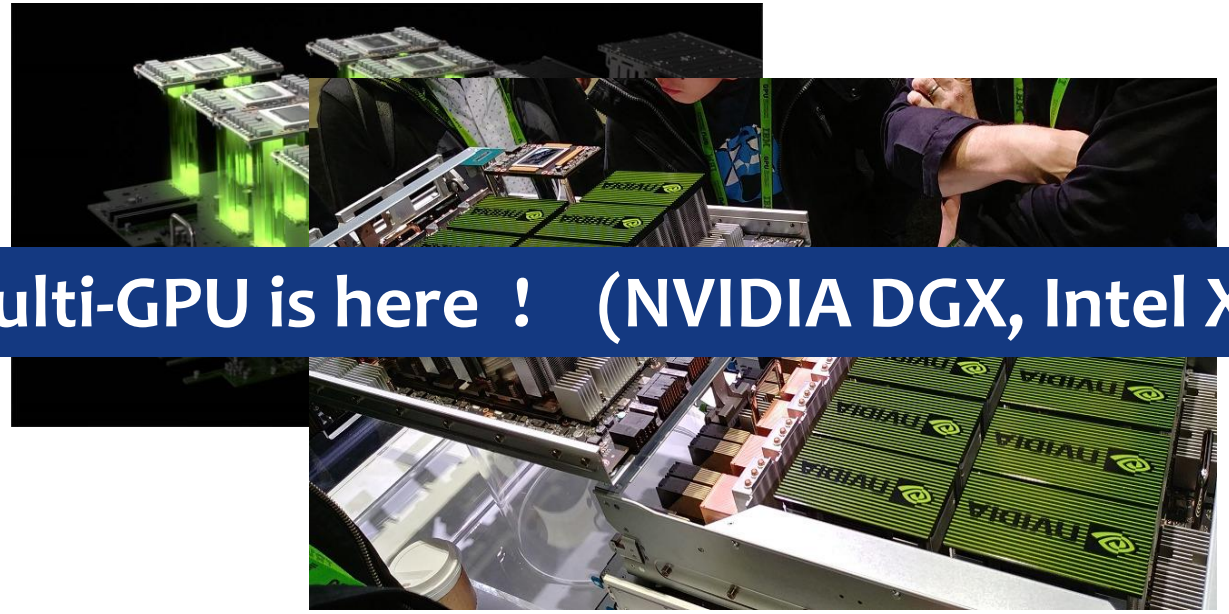
Graph Processing



DNN

ChatGPT

Datacenter Workloads



Multi-GPU is here ！ (NVIDIA DGX, Intel Xe)

Ever-growing application complexity

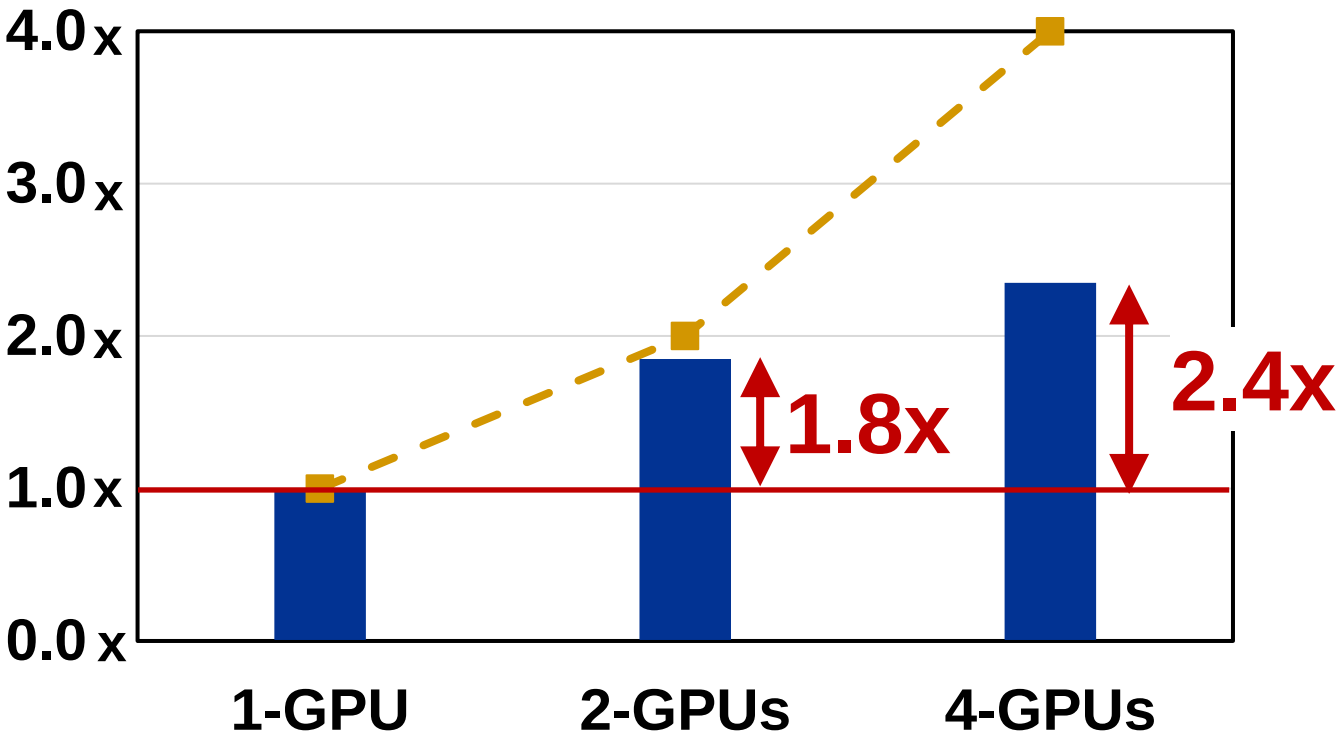
and input dataset sizes

Multi-GPUs provide aggregated memory capacity

Unified virtual memory simplifies application deployment

University of Pittsburgh

Multi-GPU Scalability



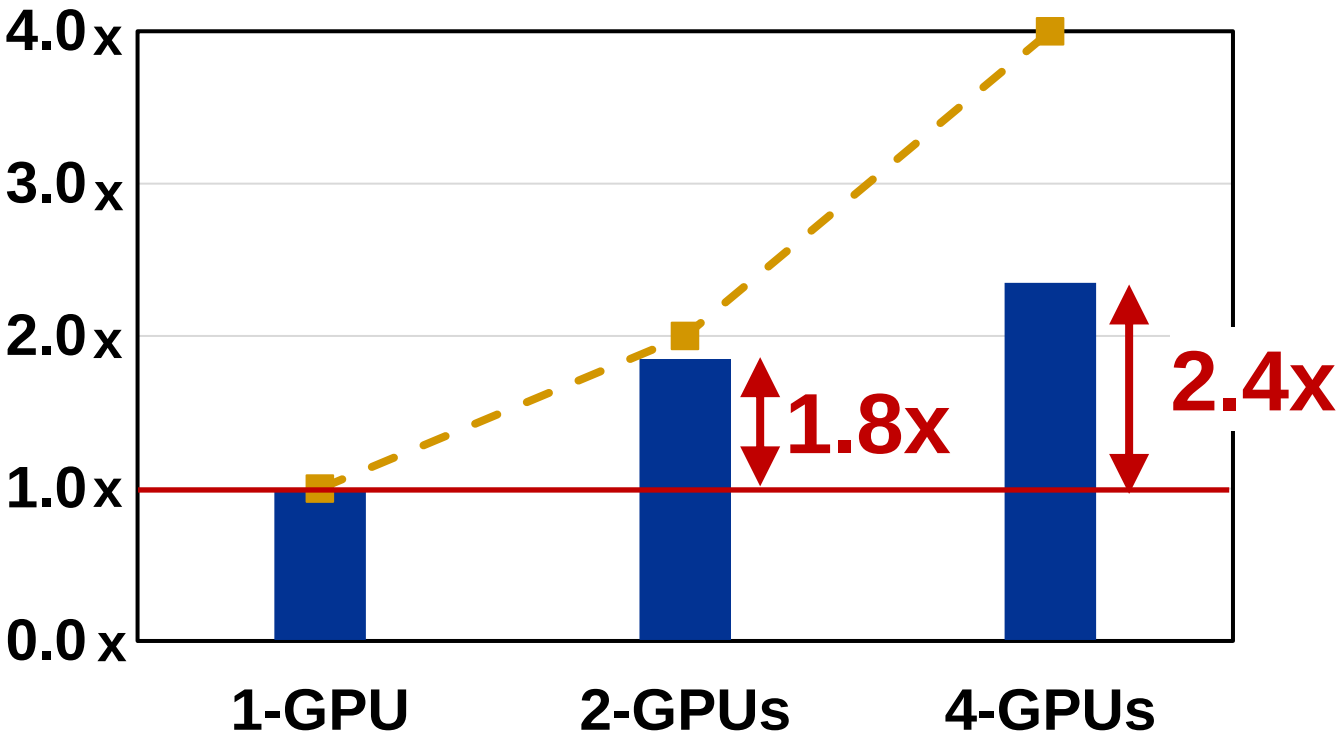
Why performance gap:

- **NUMA data access**
- **Data transfer**
- **Address translation**

……

[1] https://developer.nvidia.com/blog/easy-multi-gpu-deep-learning-digits-2/

Multi-GPU Scalability



Why performance gap:

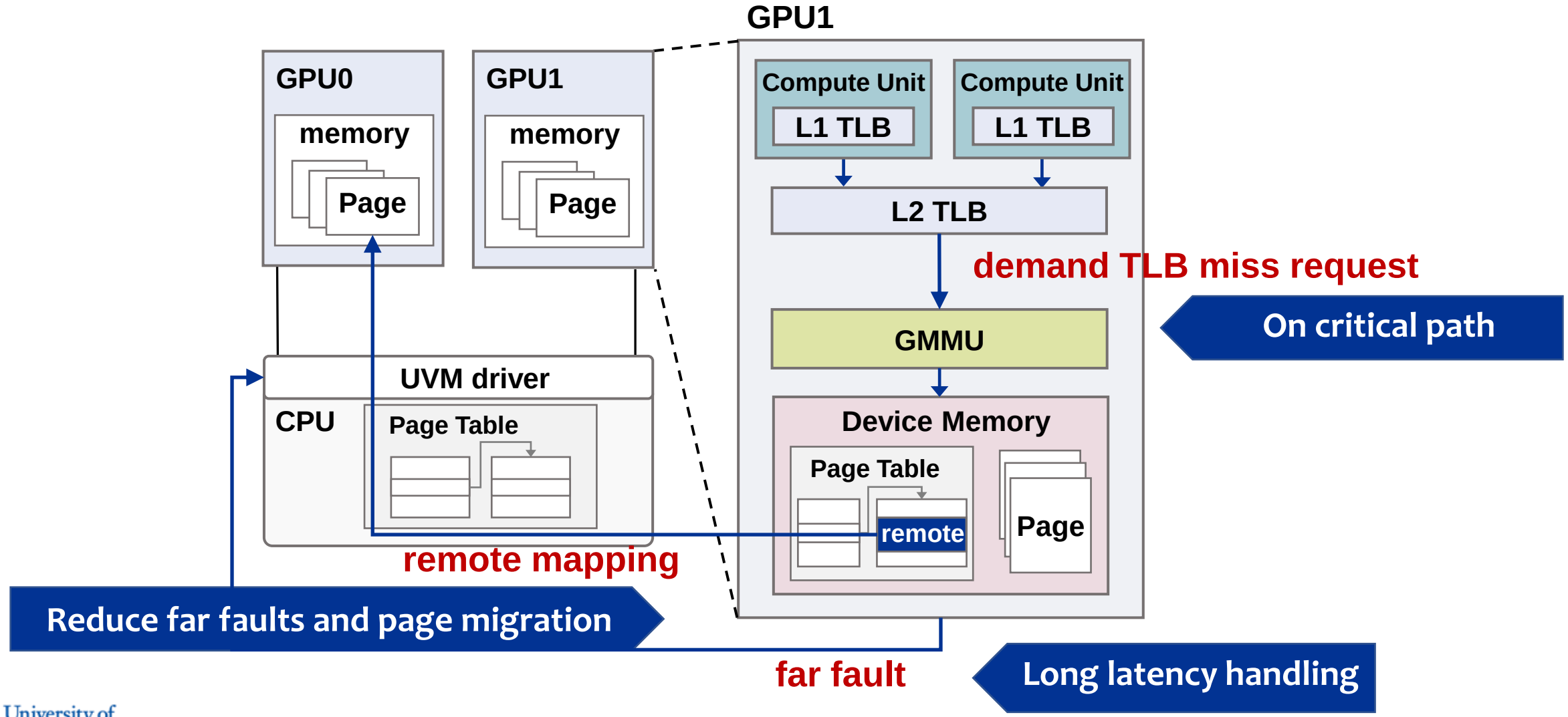
- NUMA data access
- Data transfer
- Address translation

Occupies 30%-80% of execution time

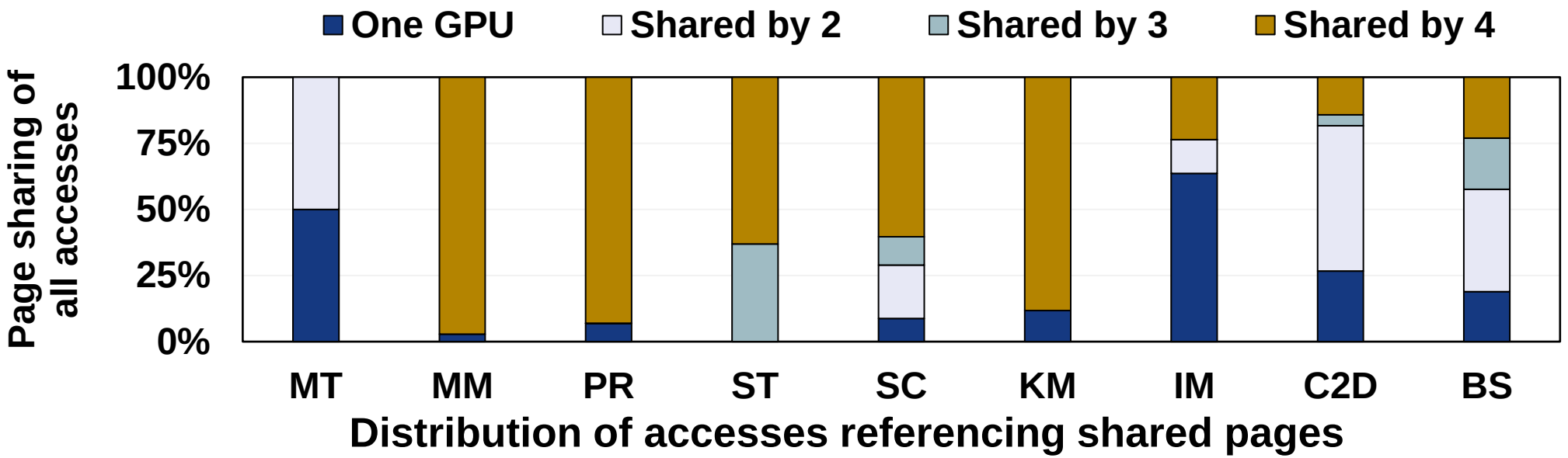
(T. Allen et al. [SC'21], B.Li et al. [MICRO'21], S. Shin et al. [MICRO'18])

[1] https://developer.nvidia.com/blog/easy-multi-gpu-deep-learning-digits-2/

Address Translation in Multi-GPU



Page Sharing Characterization



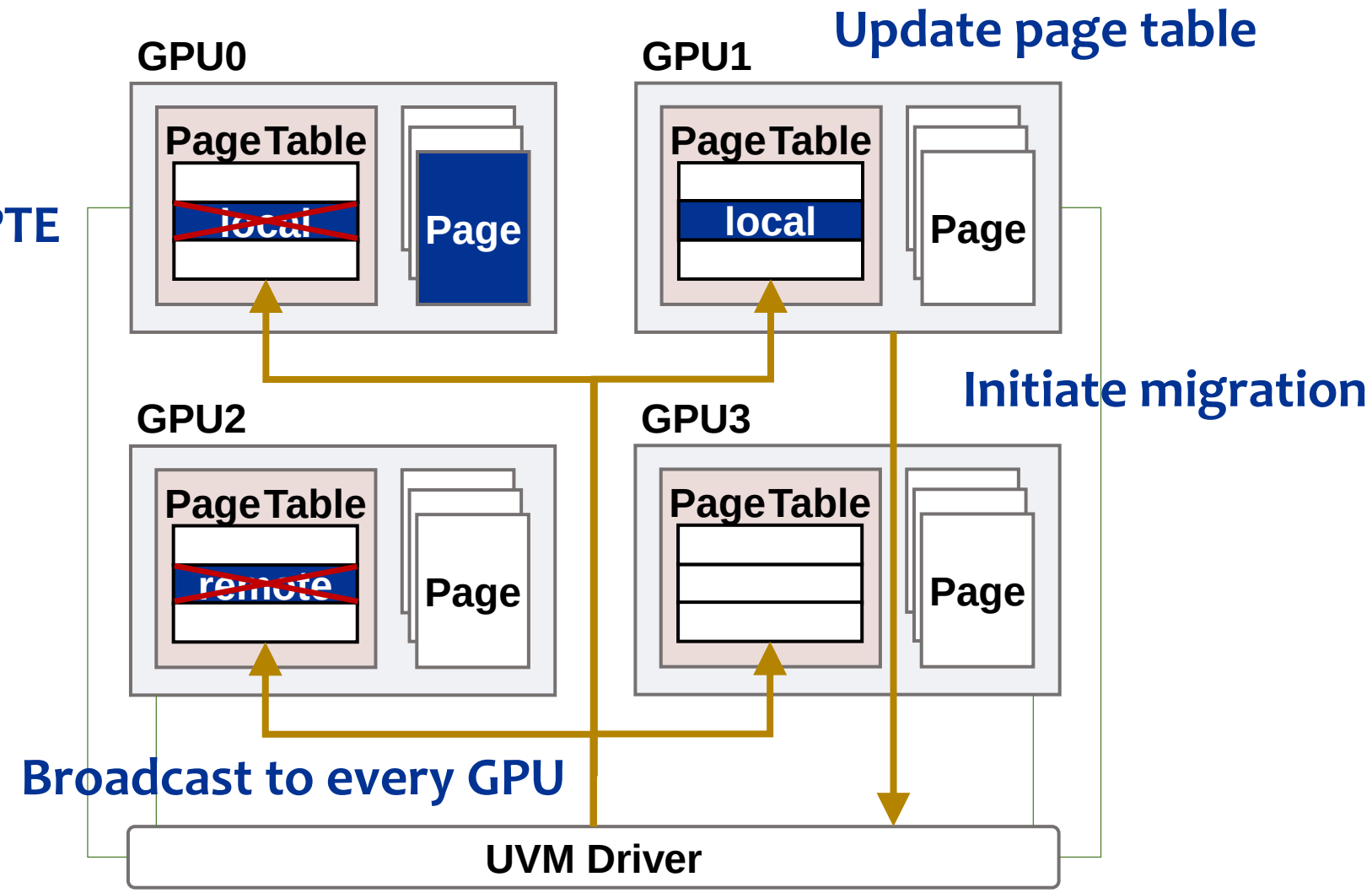
Observation:

Significant page sharing among multiple GPUs

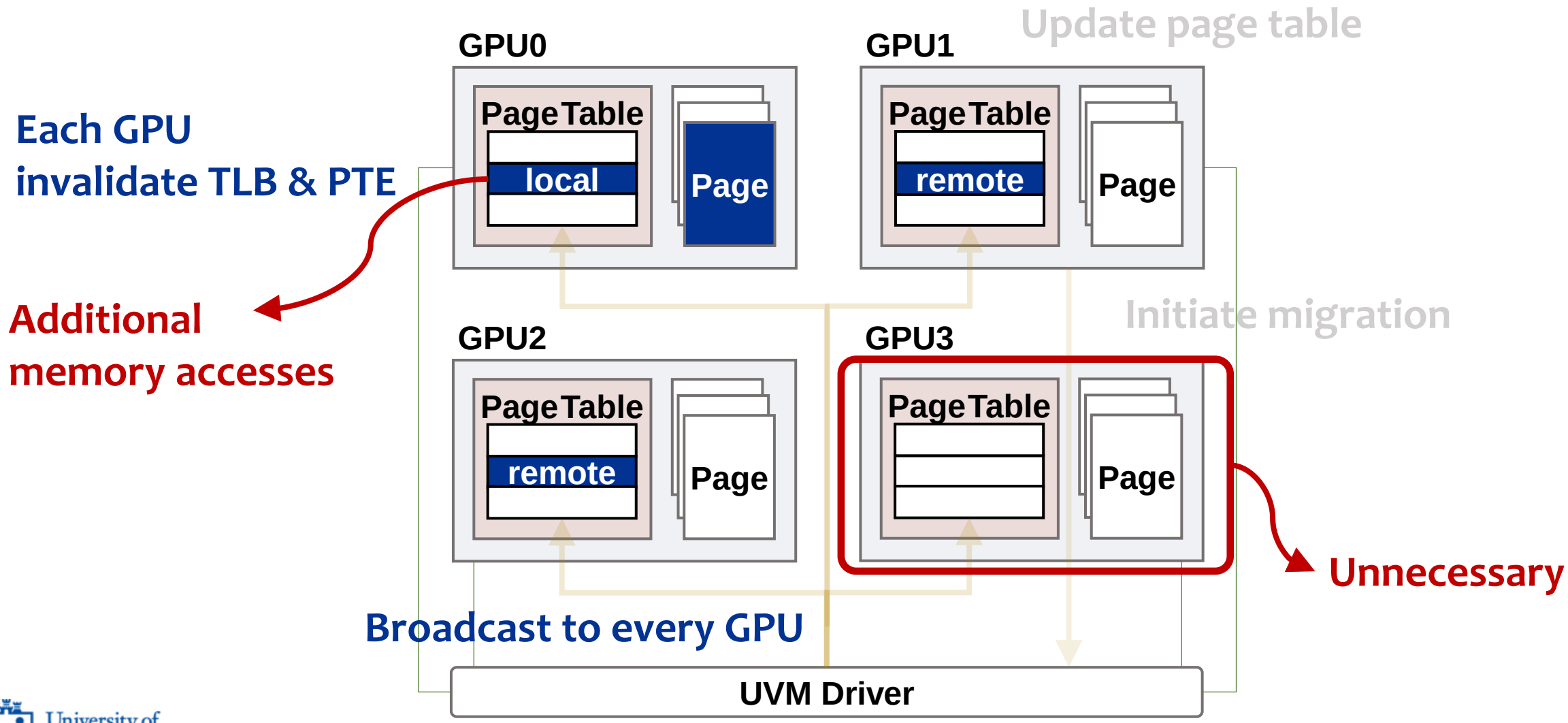
- Reduce NUMA data access -> **Frequent page migration**

Page Migration with Remote Mapping

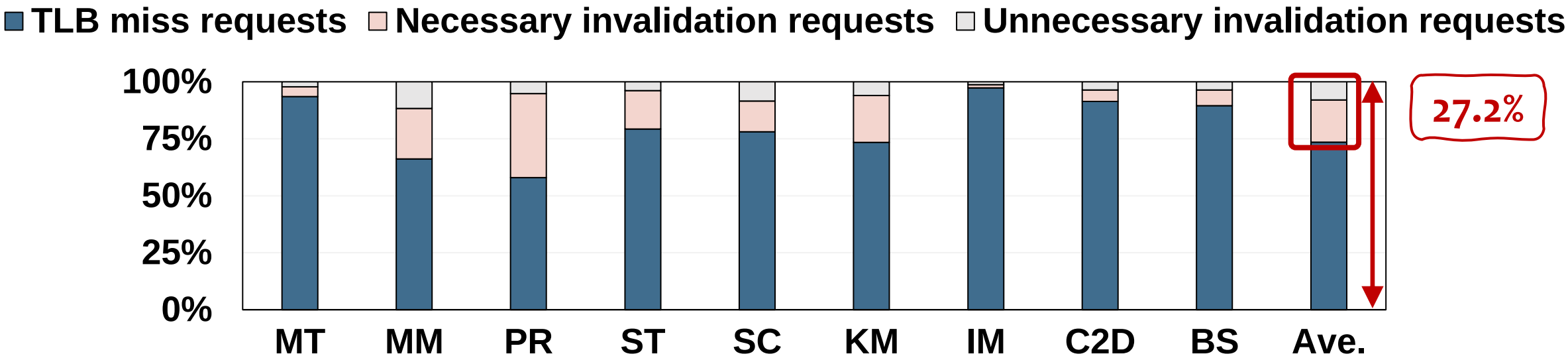
Each GPU
invalidate TLB & PTE



Problem of Existing Page Migration



PTE Invalidation Characterization

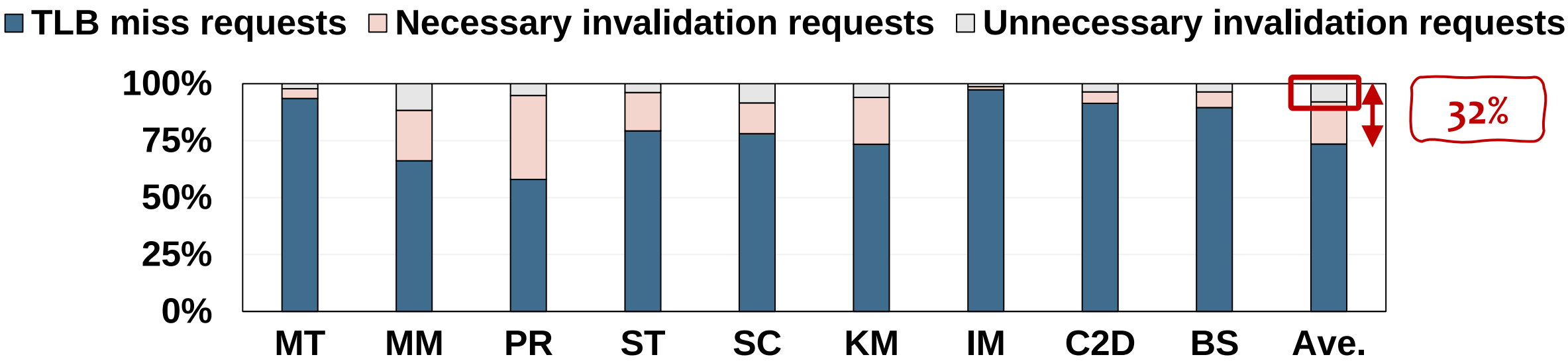


Observation:

Substantial PTE invalidation requests

- **PTE invalidations** accounts for **a quarter** of total requests to the page walker.

PTE Invalidation Characterization



Observation:

Substantial PTE invalidation requests

- **PTE invalidations** accounts for **a quarter** of total requests to the page walker.
- **One-third** of PTE invalidations broadcasted are **unnecessary**.

University of Pittsburgh

PTE Invalidation Characterization

How is performance impacted by PTE invalidations?

Observation:

Substantial PTE invalidation requests

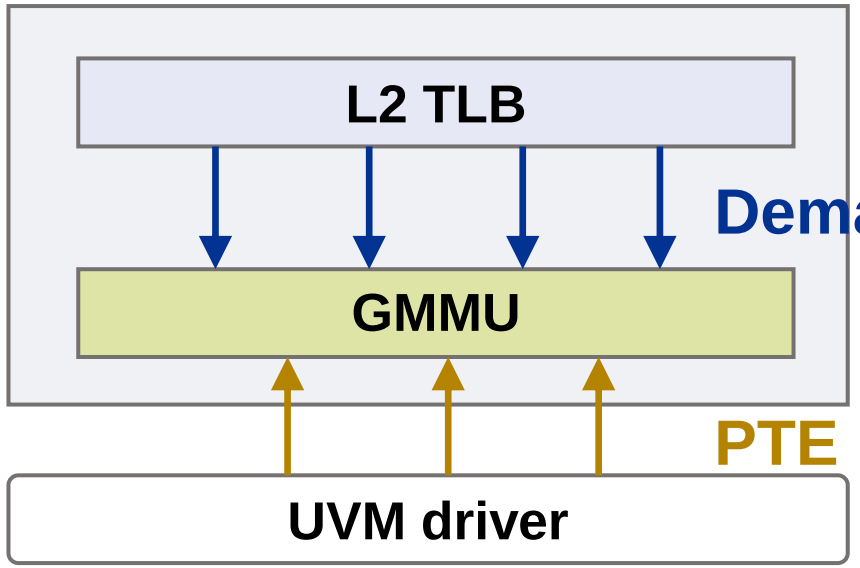
- **PTE invalidations** accounts for **a quarter** of total requests to the page walker.
- **One-third** of PTE invalidations broadcasted are **unnecessary**.

Invalidation Overhead

- **Contend with demand TLB miss request.**

Invalidation Overhead

- Contend with demand TLB miss request.



Demand TLB miss request

PTE invalidation request

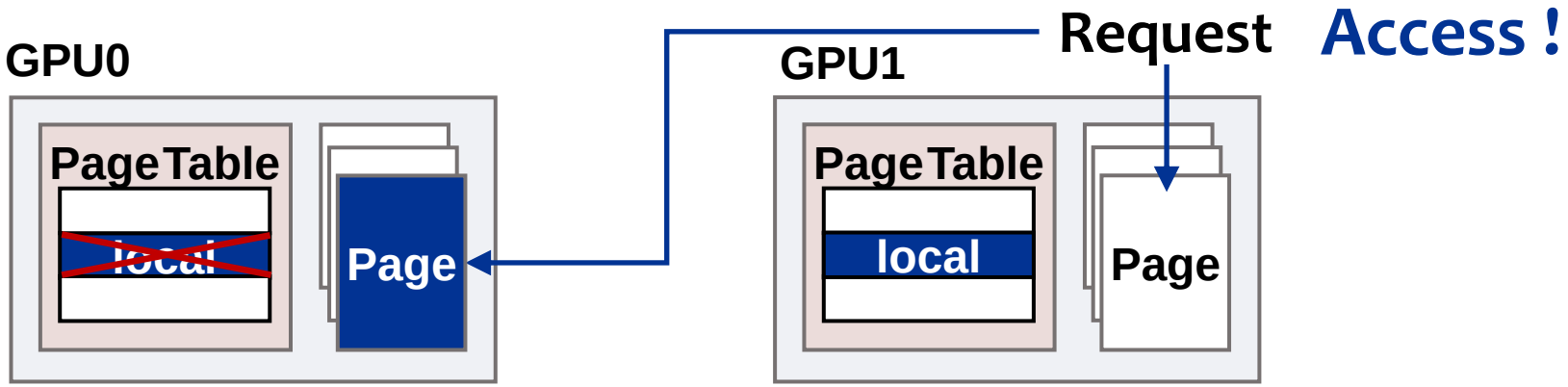
Without the interference of invalidation requests, the demand TLB miss request latency reduced by 55.8%.

Invalidation Overhead

- **Extra latency for waiting page to be migrated.**

Invalidation Overhead

- Extra latency for waiting page to be migrated.



30% of total page migration time is spent on waiting page to be migrated.

Problem Summary

Problem:

PTE invalidations

- **Unnecessary broadcast**
- Contend with the **demand TLB miss requests**

Address the overheads associated with PTE invalidations

In-PTE Directory Invalidation

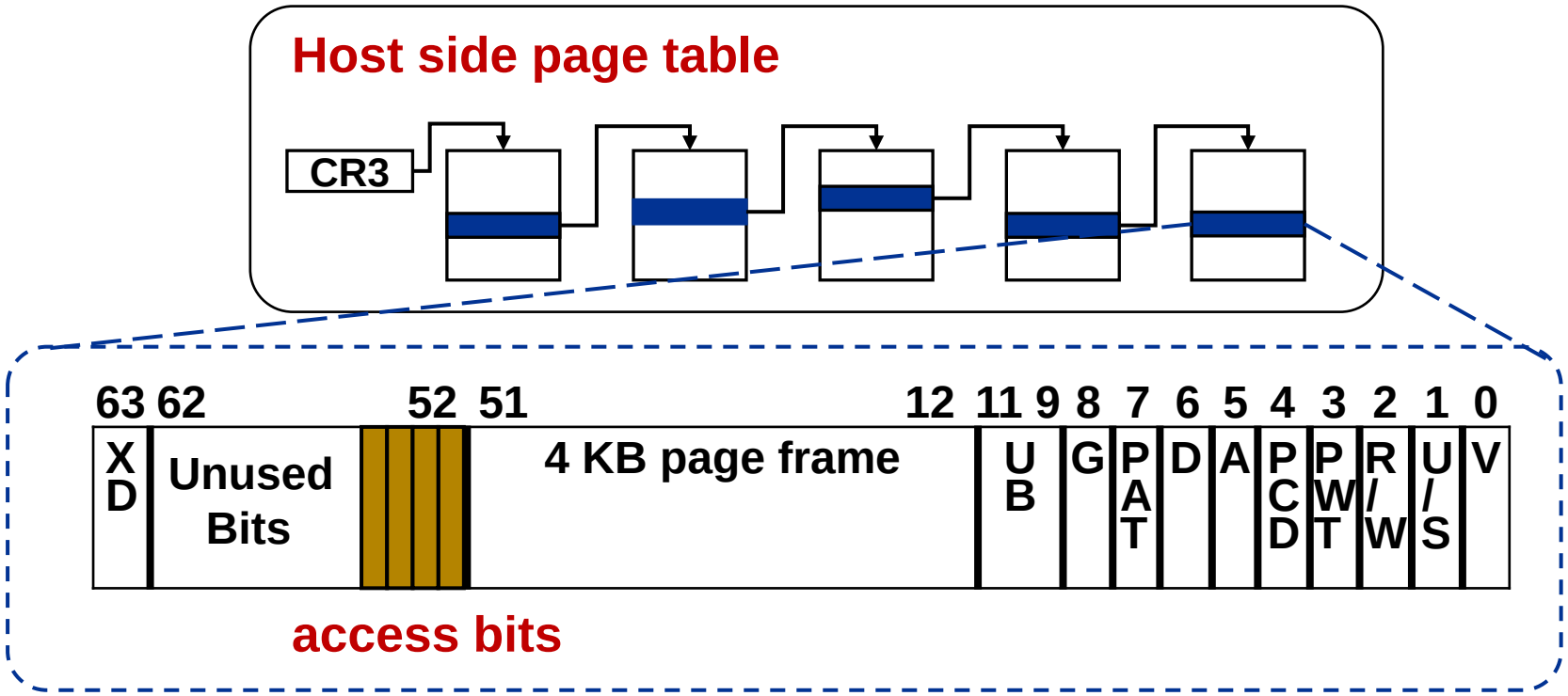
Lazy Invalidation

In-PTE **D**irector**Y** and **L**azy Inva**L**idation **(IDYLL)**

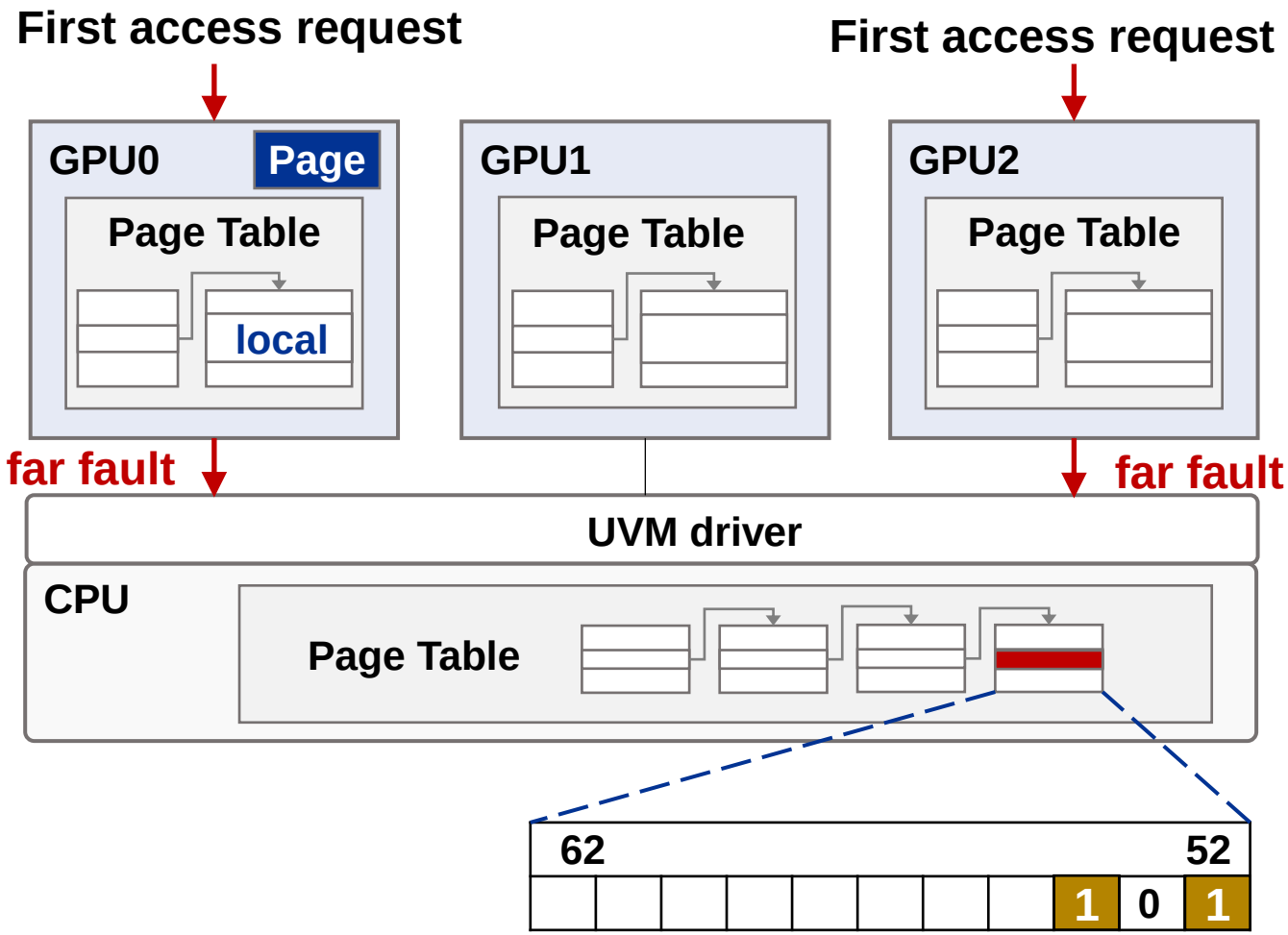
University of Pittsburgh

IDYLL : In-PTE Directory Invalidation

Host-side page table holds all up-to-date translations in all GPU.

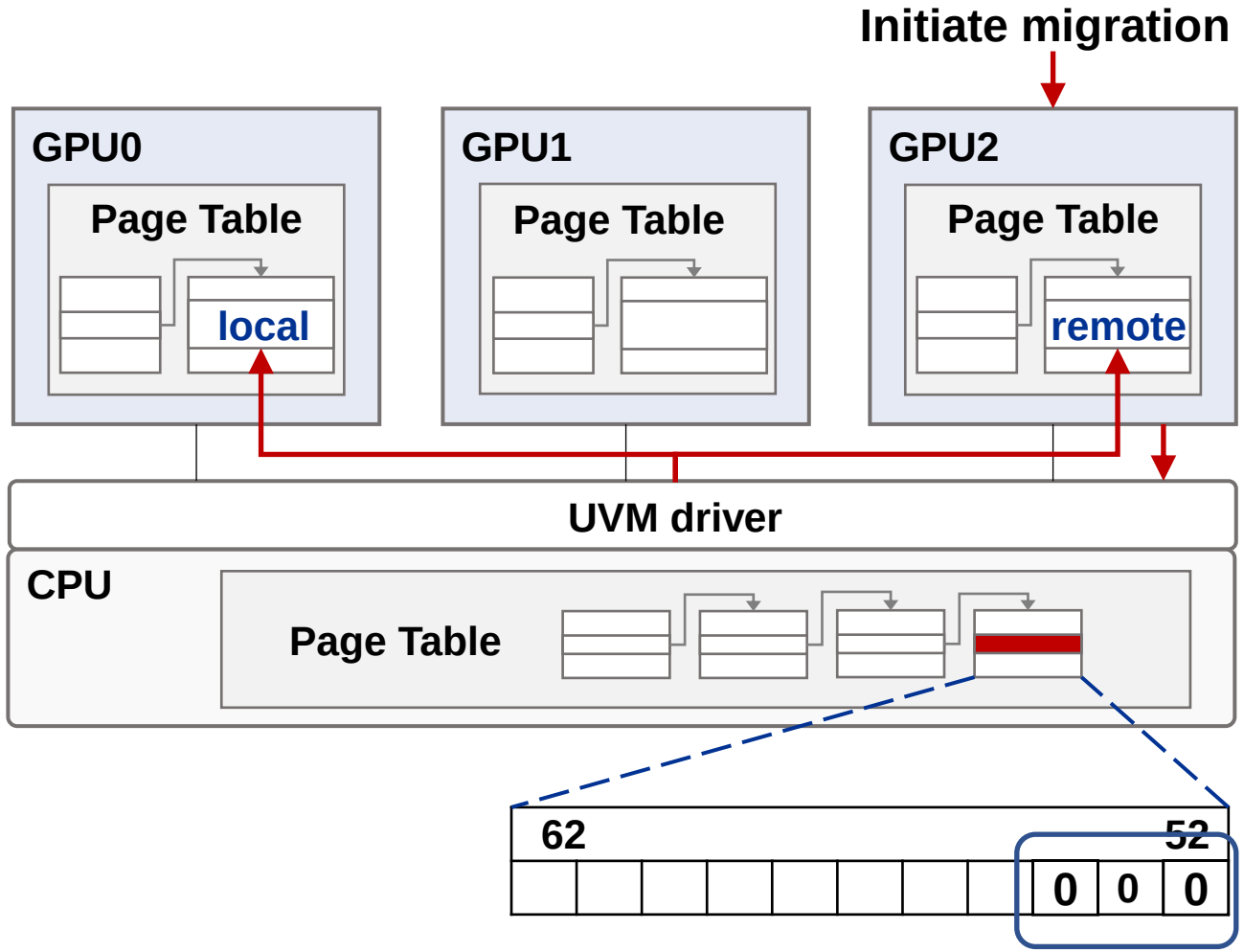


IDYLL : In-PTE Directory Invalidation



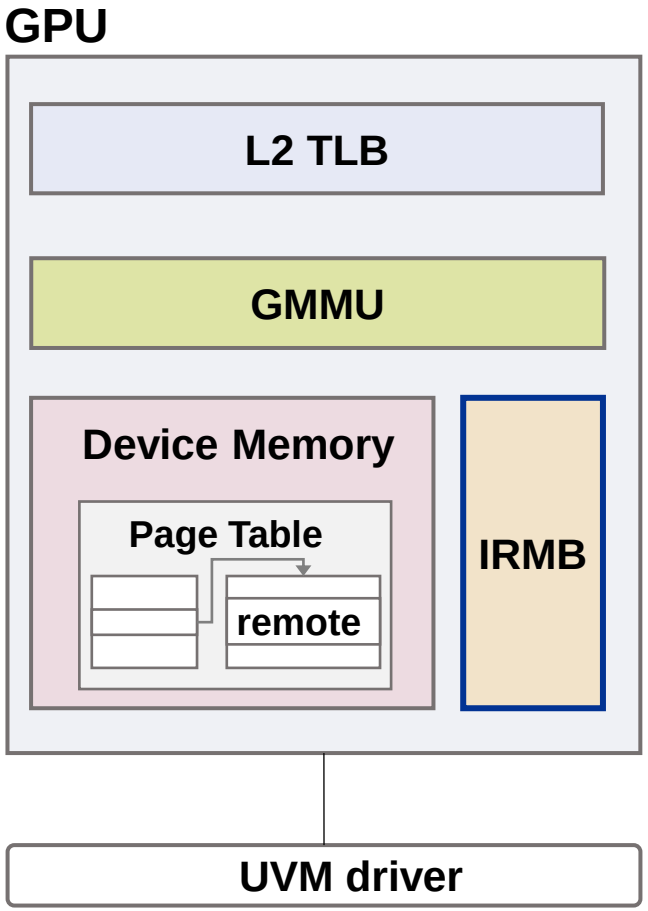
At the time of **first far fault**, the access bit is **set to 1**.

IDYLL : In-PTE Directory Invalidation



Send invalidations **only to GPUs with access bits set to 1.**

IDYLL : Lazy Invalidation



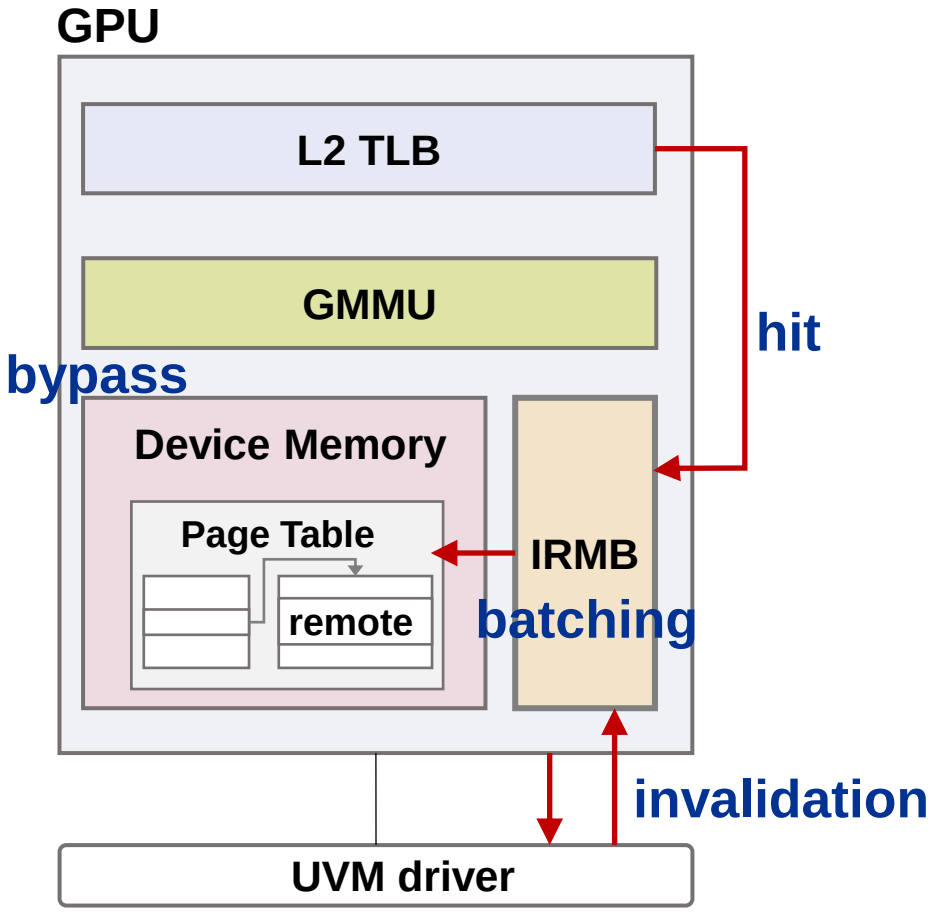
In-PTE Directory Invalidation -> unnecessary invalidations

Still substantial valid translation mappings

Invalidation Request Merging Buffer (IRMB)

temporally buffer and batch invalidation requests and **lazily update** the local page table.

IDYLL : Lazy Invalidation



IRMB :

The invalidation request **inserted into IRMB** instead of **performing invalidation immediately.**

Nearby invalidations are **merged** into one entry and **batched write back** to page table

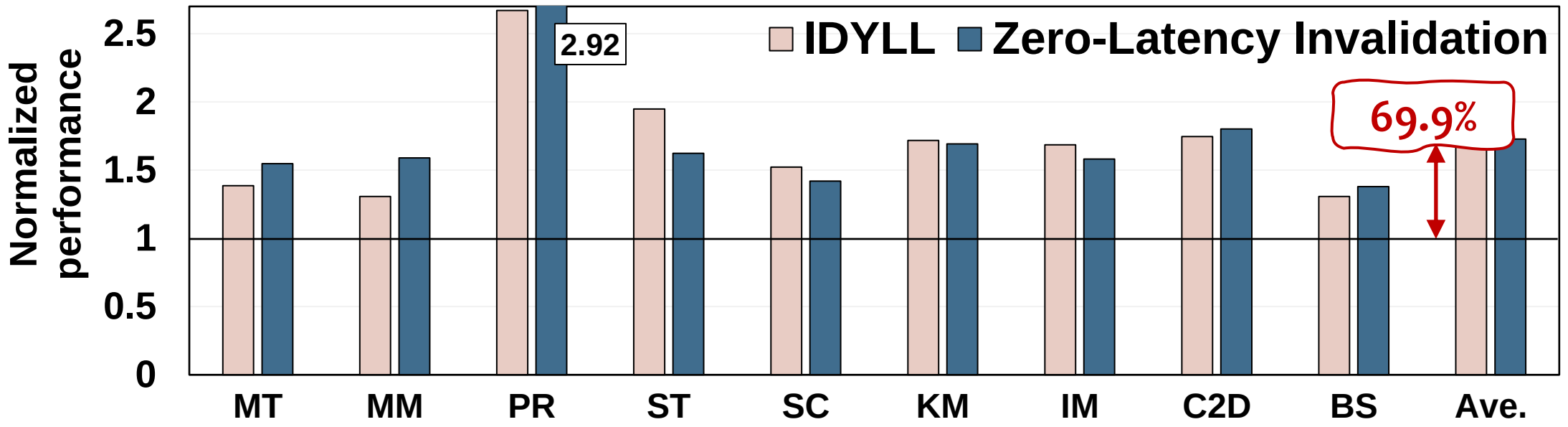
Serve as **indicator of invalid PTE** that, if hit in IRMB, can **bypass** local page table walk

Methodology

- Simulator: MGPUSim [ISCA 19']
- Workloads: 9 applications from Hetero-Mark, AMDAPPSDK , SHOC, and DNN Mark benchmark suites.
- Hardware setup:

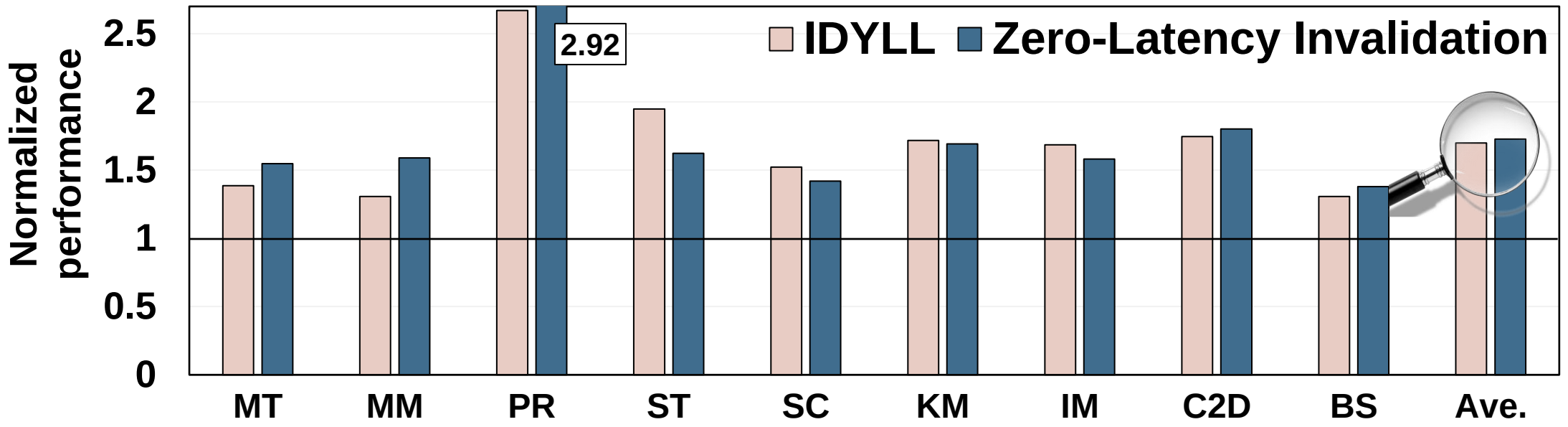
	Configuration
L2 TLB	512 entries, 16 way, CUs shared
Page table walk	GMMU 8 shared page table walker, 100-cycle latency per level
Page table cache	128 entries shared across page table walker
Access counter threshold	256

Evaluation



IDYLL achieves an average of 69.9% performance improvement.

Evaluation



IDYLL nears zero-latency invalidation performance.

More in the Paper

- **Detailed lookup process of IDYLL**
- **Experiment configurations**
- **Performance improvement breakdown**
- **Performance sensitivity to :**
  - IRMB size
  - Number of page table walk threads
  - L2 TLB size
  - Number of GPUs
  - Number of unused bits
  - Access counter threshold
- **Comparison with Large Page, Page Replication, State-of-the-art**

Summary

Problem: PTE invalidation overheads in multi-GPU systems.

- *Unnecessary PTE invalidation broadcast*
- *Contend with demand TLB miss request*

IDYLL:

A. In-PTE Directory **reduces unnecessary PTE invalidations**
B. Lazy Invalidation **amortizes and minimizes PTE invalidation overheads**

Improves **performance** by **69.9%** on average.

University of Pittsburgh

Thanks! Q&A

IDYLL: Enhancing Page Translation in Multi-GPUs

via Light Weight PTE Invalidations

***Bingyao Li*[1], Yanan Guo[1], Yueqi Wang[1], Aamer Jaleel[2], Jun Yang[1], Xulong Tang[1]**

[1]University of Pittsburgh, [2]NVIDIA

Email: bil35@pitt.edu