香港科技大学（广州）
THE HONG KONG UNIVERSITY OF SCIENCE AND TECHNOLOGY (GUANGZHOU)

NB-Walker: Non-Blocking Page Table Walker to Enhance Address Translation in NUMA GPUs

Zihang Chen[1], Lieven Eeckhout[2], Hongyuan Liu[3]

Tianao Ge[1], Xinkai Wang[4], Jiayi Huang[1]

[1] The Hong Kong University of Science and Technology (Guangzhou)

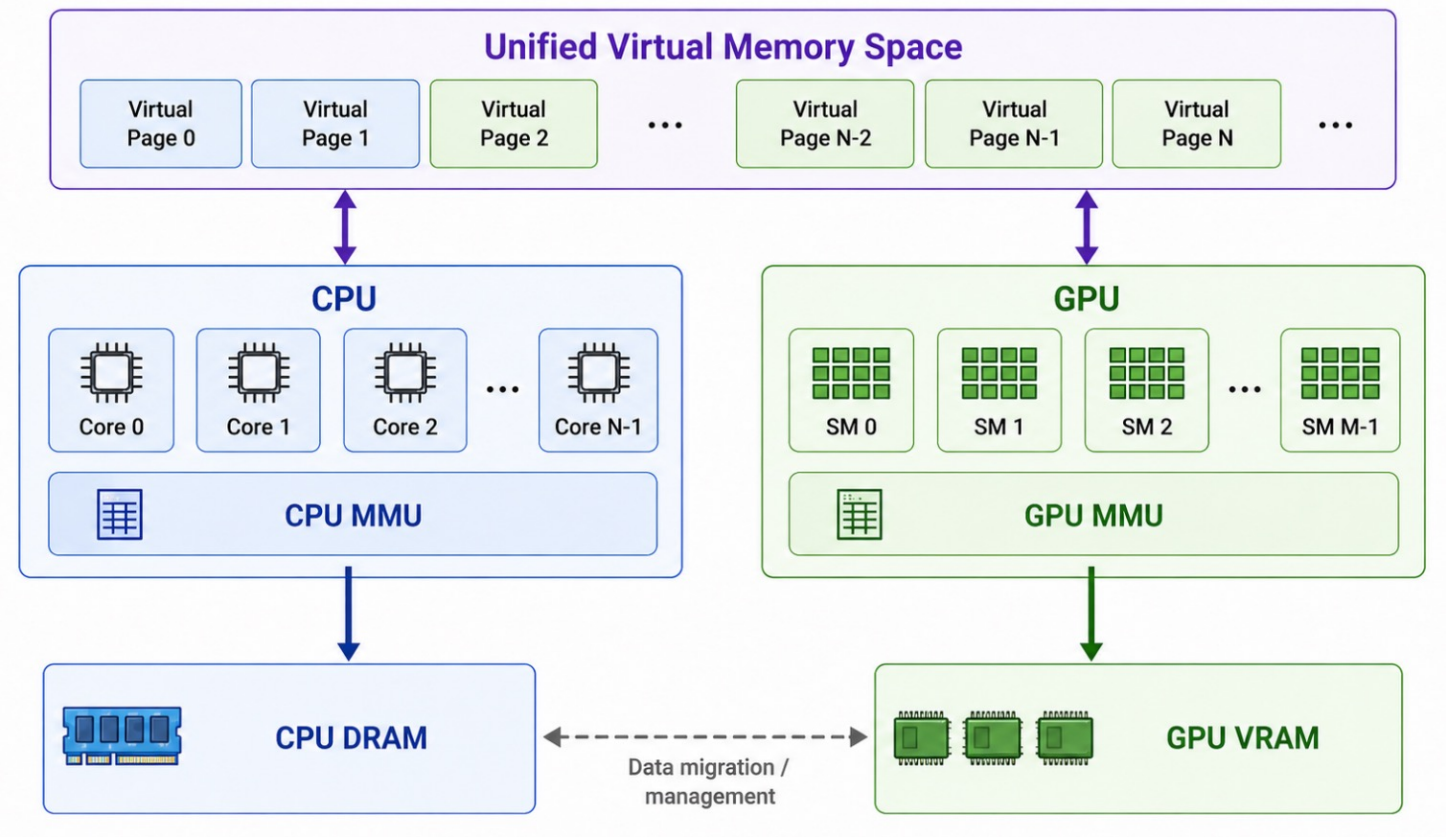
[2] Ghent University

[3] Stevens Institute of Technology

[4] Shanghai Jiao Tong University

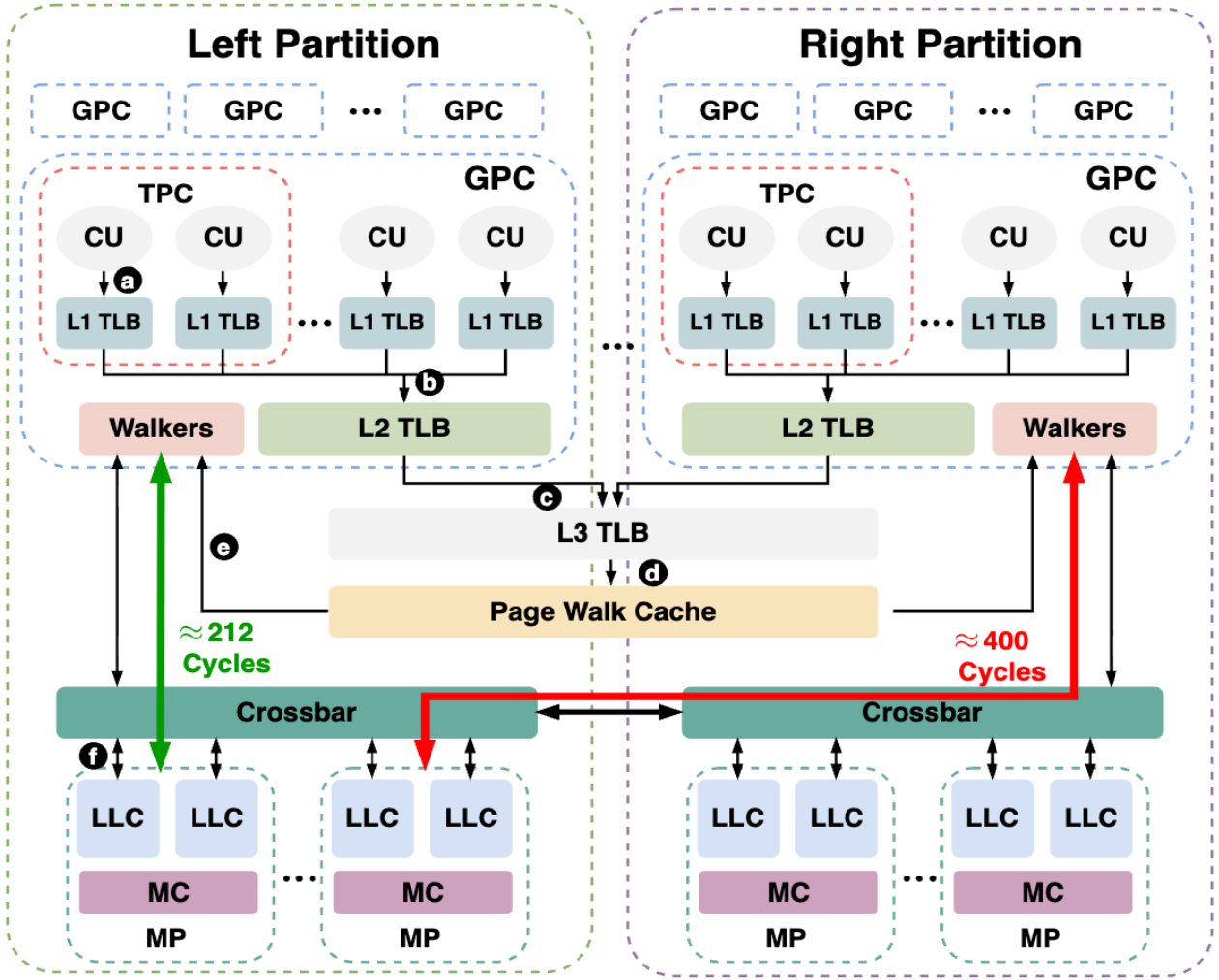


Why Address Translation is Important in GPUs



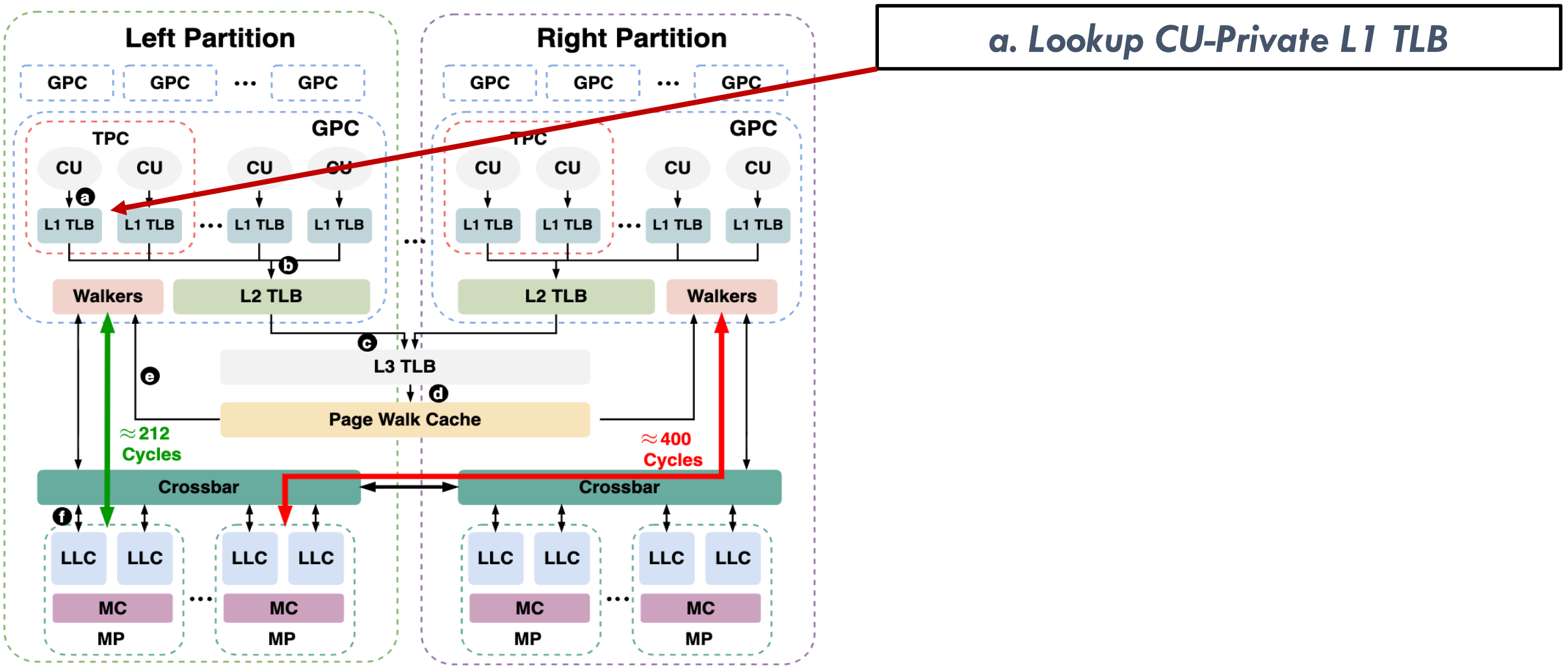
- Unified Virtual Memory Space
  - Simplify programming
  - i.e., cudaMemManaged()
- Memory Oversubscription
- Protection & Isolation
  - Prevent unauthorized memory accesses

GPU Address Translation Workflow



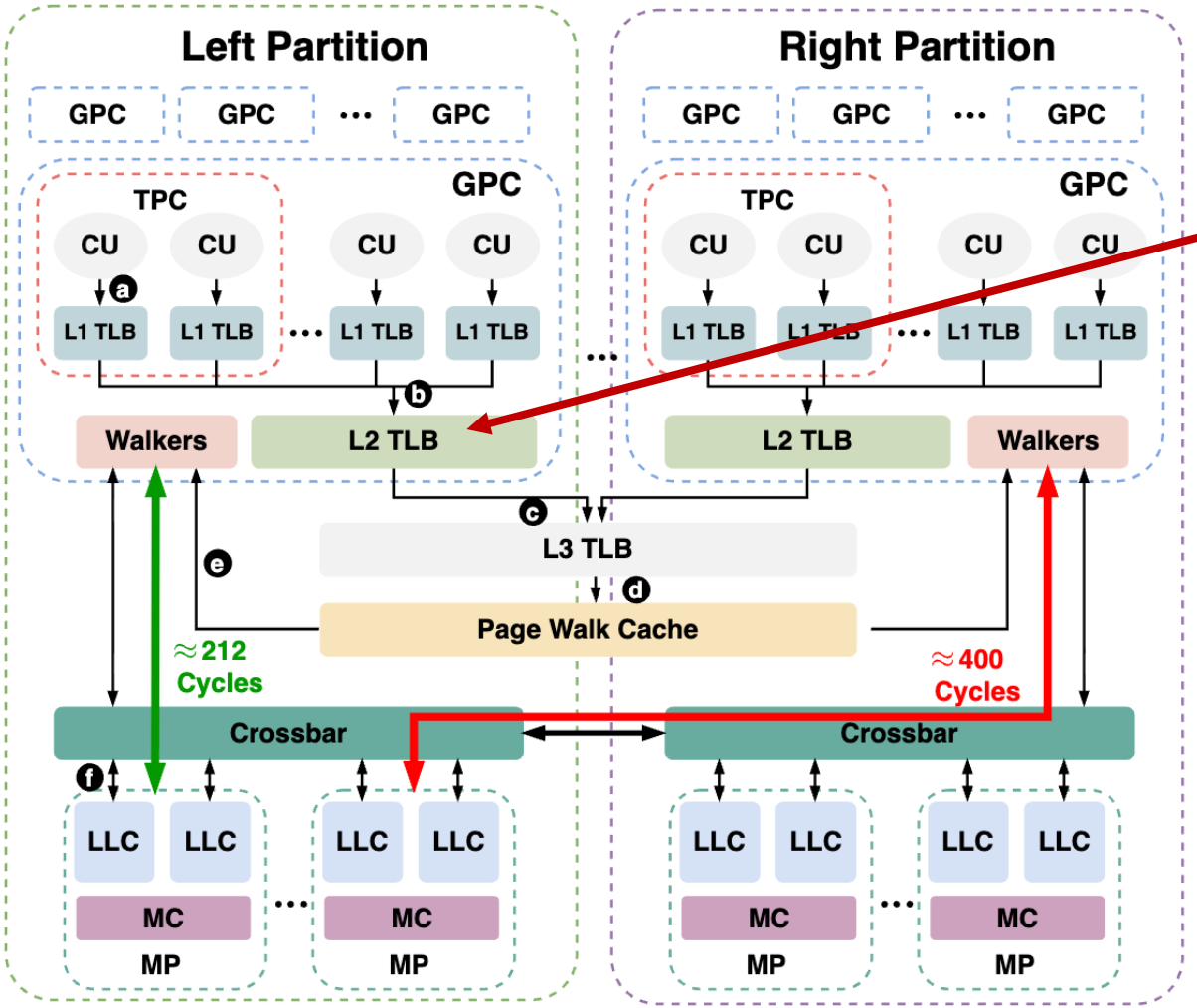
Jin, Zhixian, et al. "Uncovering real gpu noc characteristics: Implications on interconnect architecture." 2024 57th IEEE/ACM International Symposium on Microarchitecture (MICRO). IEEE, 2024.

GPU Address Translation Workflow



Jin, Zhixian, et al. "Uncovering real gpu noc characteristics: Implications on interconnect architecture." 2024 57th IEEE/ACM International Symposium on Microarchitecture (MICRO). IEEE, 2024.

GPU Address Translation Workflow

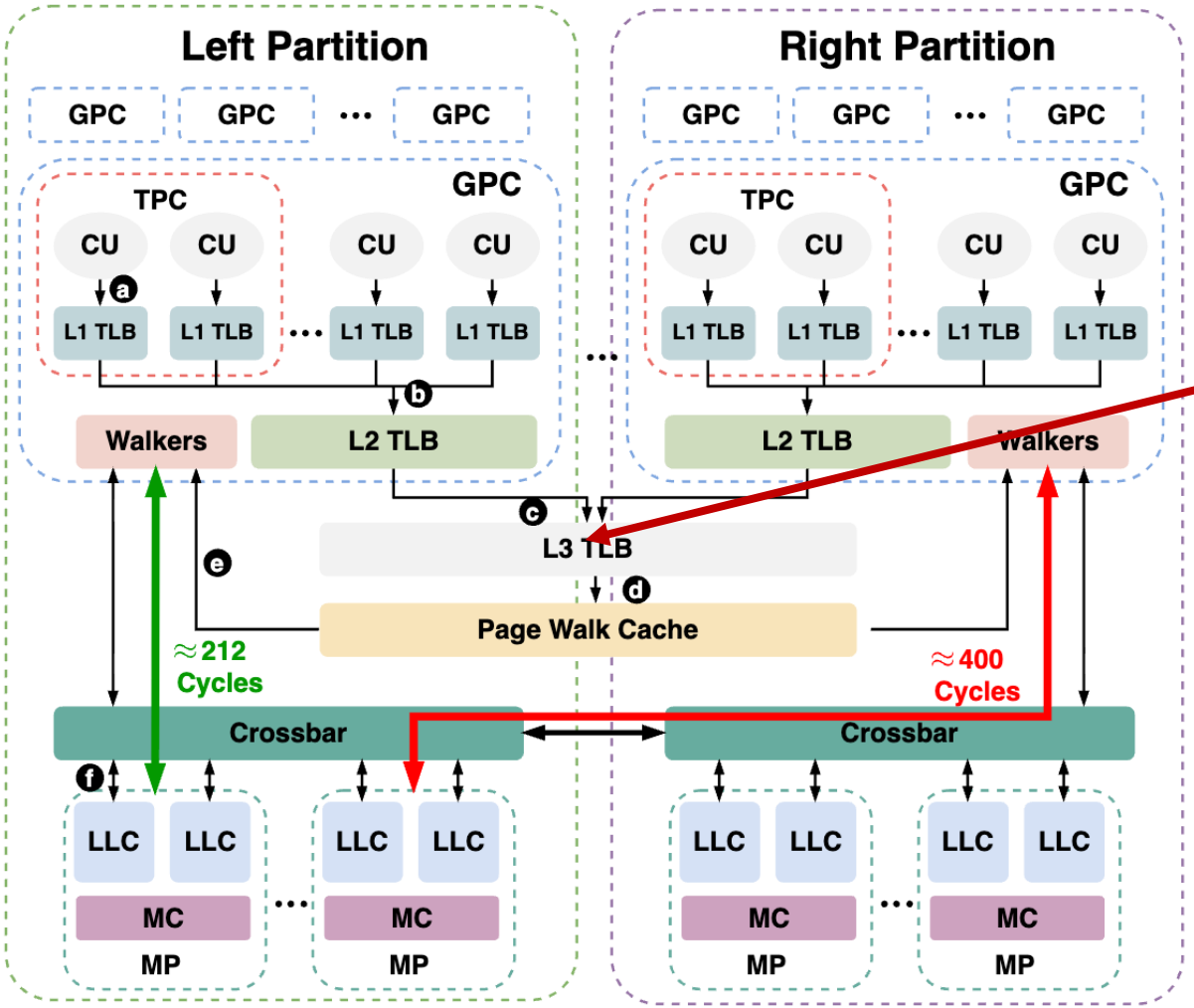


a. Lookup CU-Private L1 TLB

b. Lookup GPC-Shared L2 TLB

Jin, Zhixian, et al. "Uncovering real gpu noc characteristics: Implications on interconnect architecture." 2024 57th IEEE/ACM International Symposium on Microarchitecture (MICRO). IEEE, 2024.

GPU Address Translation Workflow



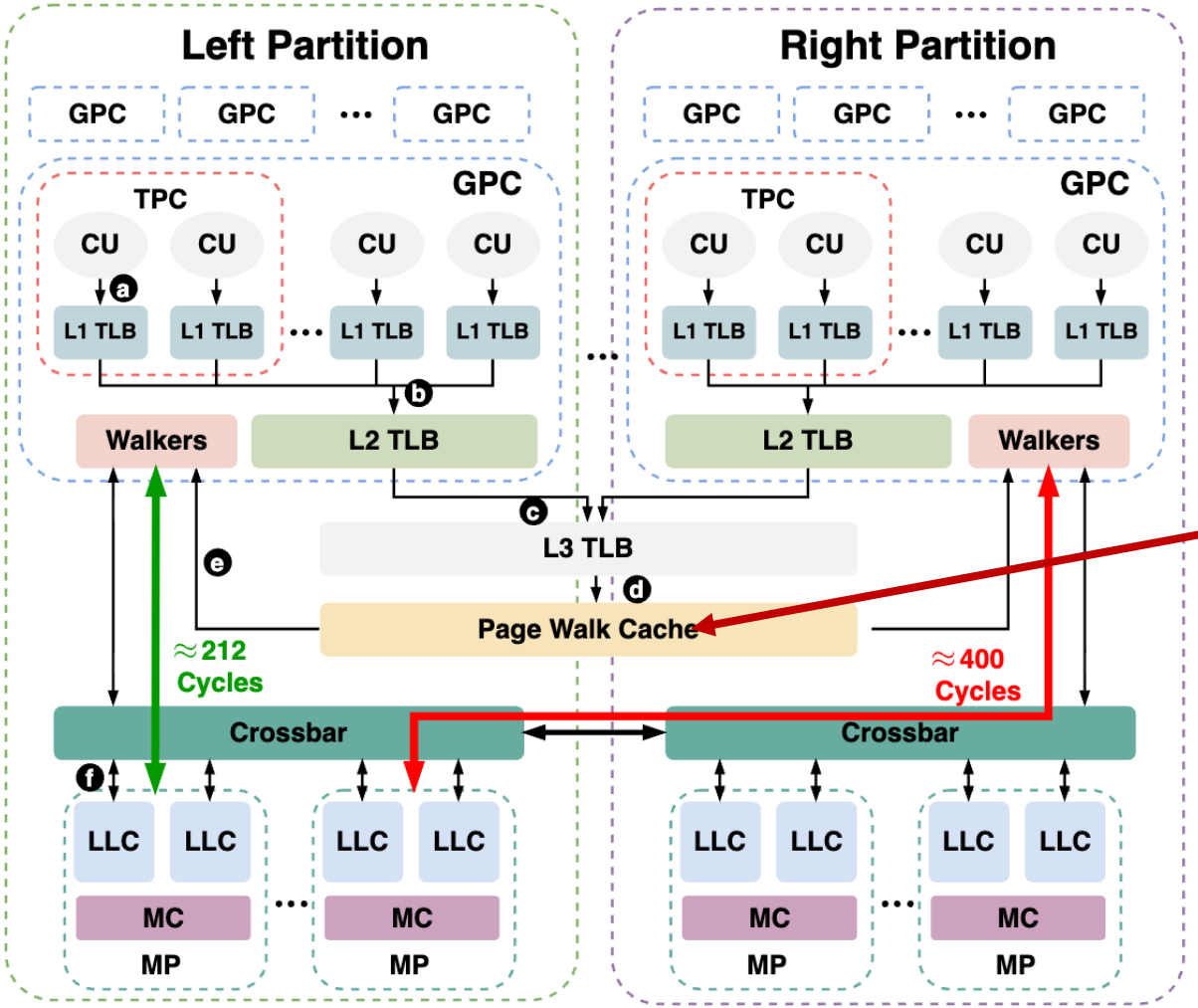
a. Lookup CU-Private L1 TLB

b. Lookup GPC-Shared L2 TLB

c. Lookup GPU-Shared L3 TLB

Jin, Zhixian, et al. "Uncovering real gpu noc characteristics: Implications on interconnect architecture." 2024 57th IEEE/ACM International Symposium on Microarchitecture (MICRO). IEEE, 2024.

GPU Address Translation Workflow



a. Lookup CU-Private L1 TLB

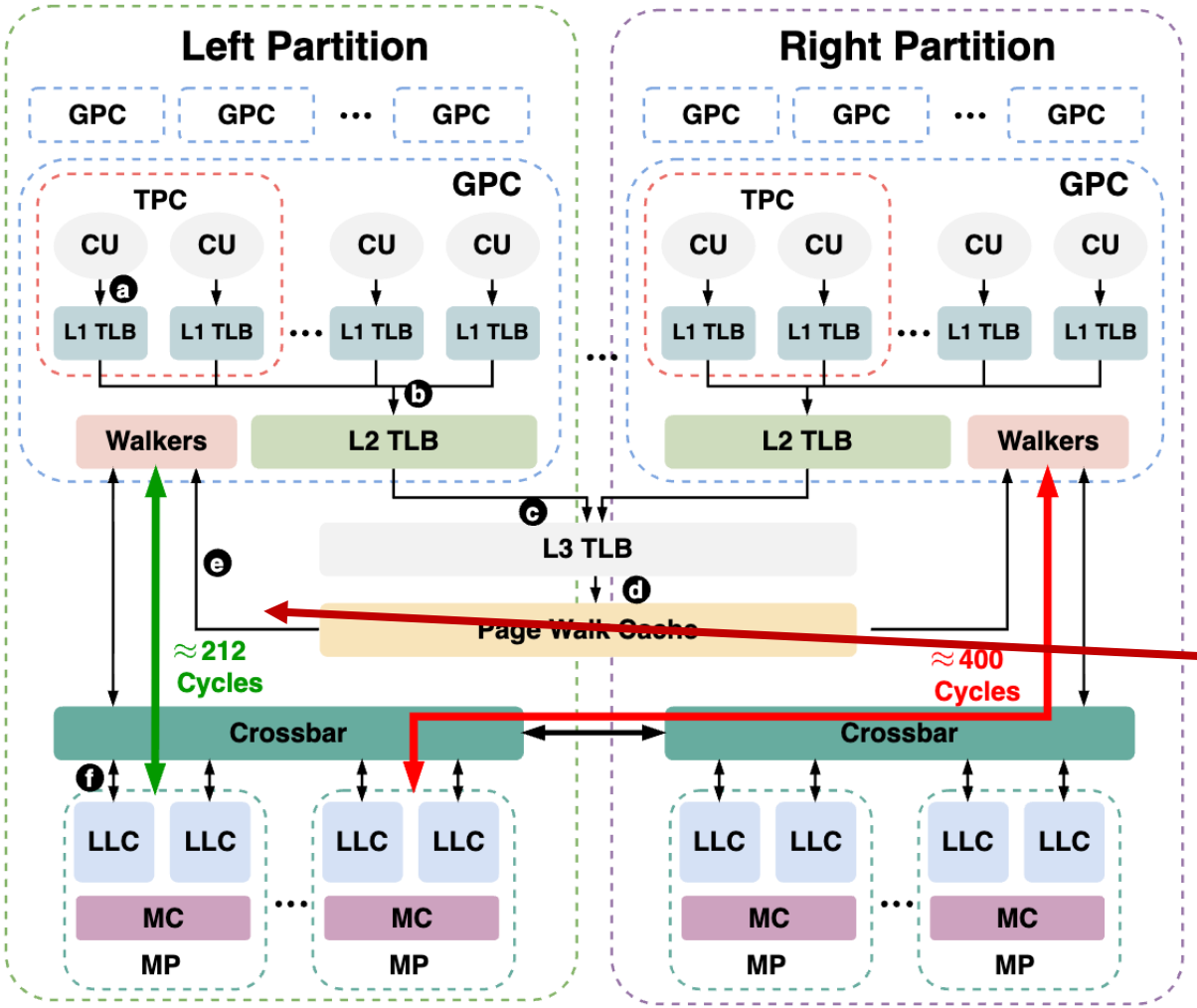
b. Lookup GPC-Shared L2 TLB

c. Lookup GPU-Shared L3 TLB

d. Lookup Page Walk Cache

Jin, Zhixian, et al. "Uncovering real gpu noc characteristics: Implications on interconnect architecture." 2024 57th IEEE/ACM International Symposium on Microarchitecture (MICRO). IEEE, 2024.

GPU Address Translation Workflow



a. Lookup CU-Private L1 TLB

b. Lookup GPC-Shared L2 TLB

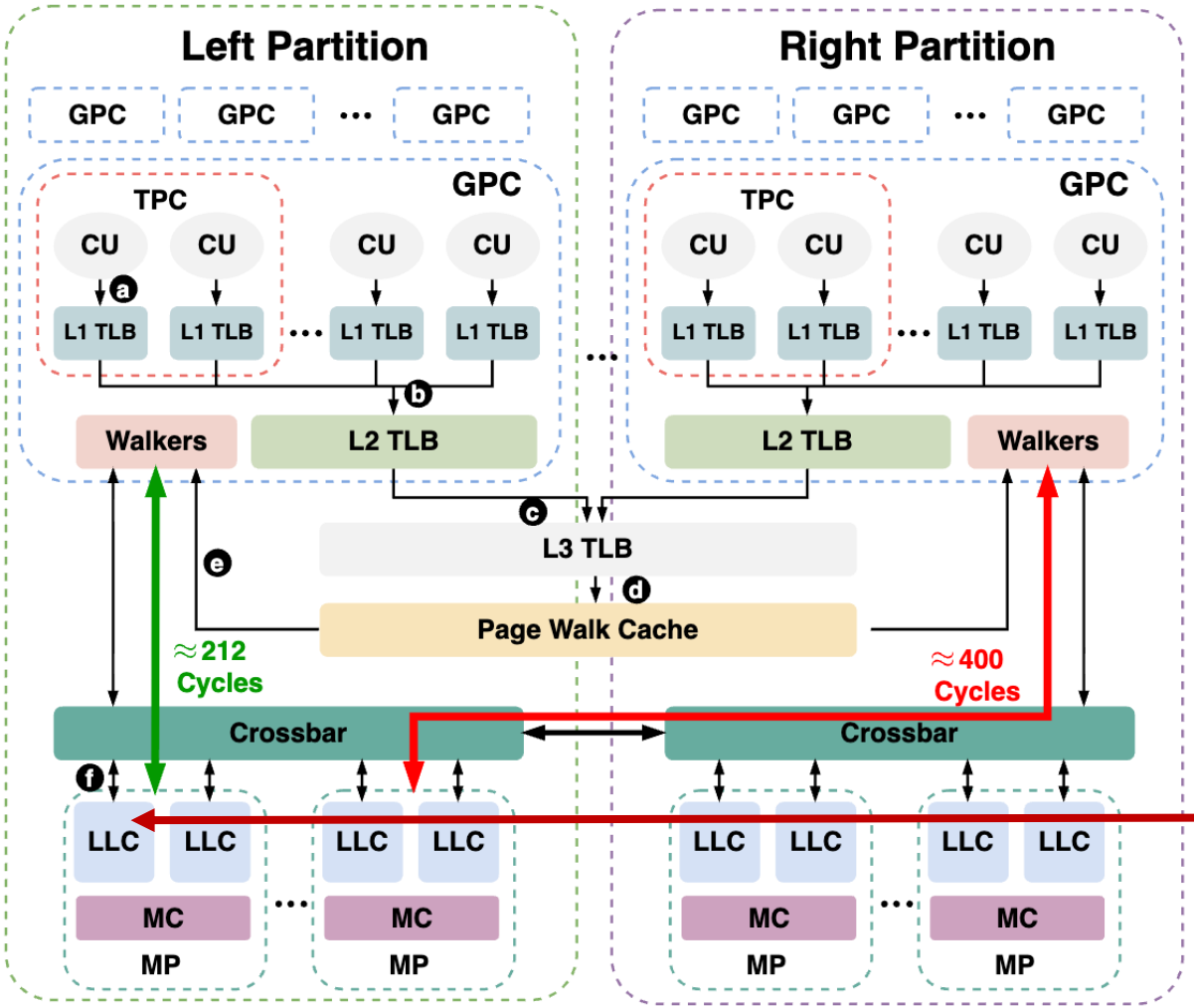
c. Lookup GPU-Shared L3 TLB

d. Lookup Page Walk Cache

e. Forward to Page Walkers

Jin, Zhixian, et al. "Uncovering real gpu noc characteristics: Implications on interconnect architecture." 2024 57th IEEE/ACM International Symposium on Microarchitecture (MICRO). IEEE, 2024.

GPU Address Translation Workflow



a. Lookup CU-Private L1 TLB

b. Lookup GPC-Shared L2 TLB

c. Lookup GPU-Shared L3 TLB

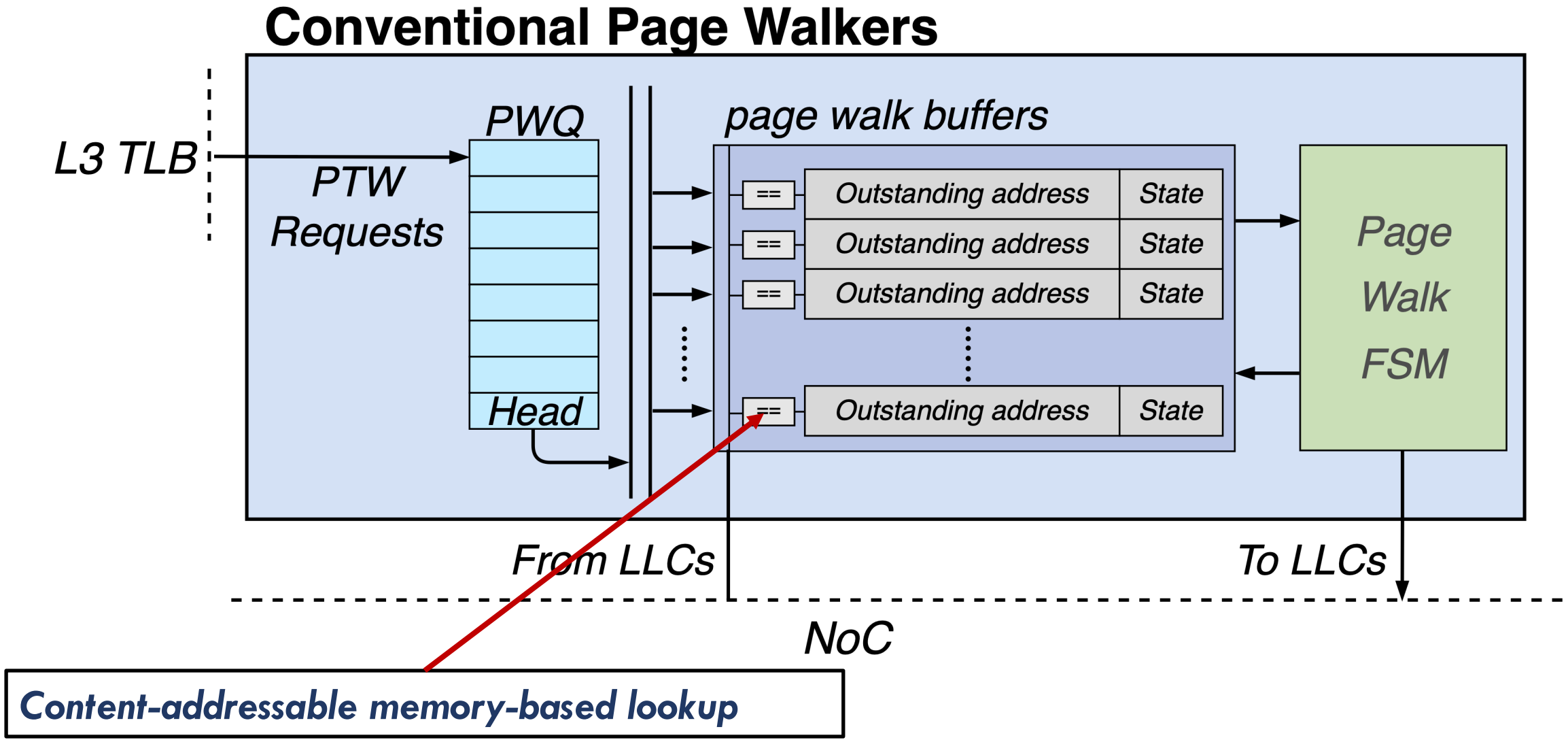
d. Lookup Page Walk Cache

e. Forward to Page Walkers

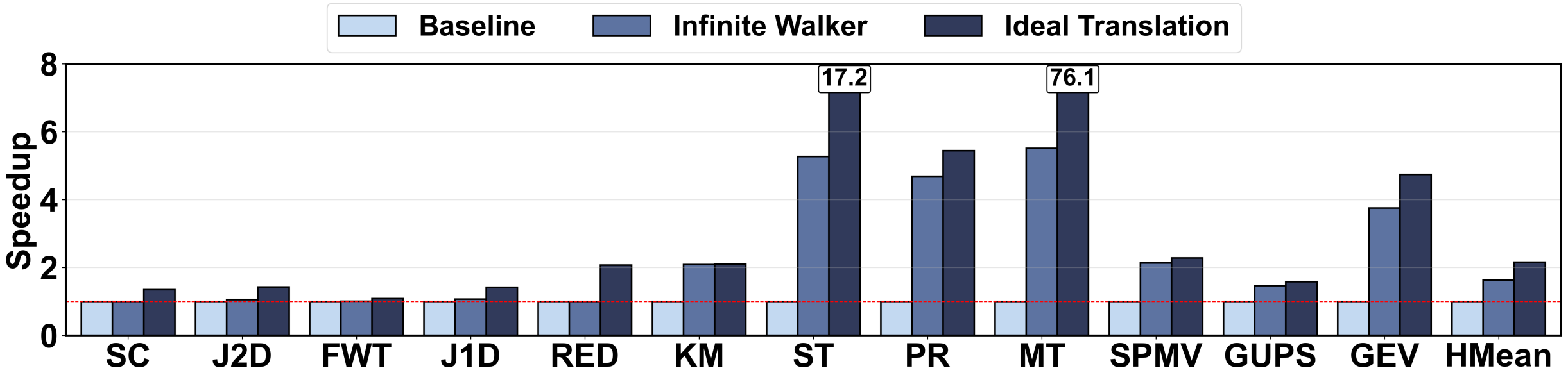
f. Fetch PTEs from Memory

Jin, Zhixian, et al. "Uncovering real gpu noc characteristics: Implications on interconnect architecture." 2024 57th IEEE/ACM International Symposium on Microarchitecture (MICRO). IEEE, 2024.

How Page Walkers Work



Motivation #1: Severe Translation Bottleneck

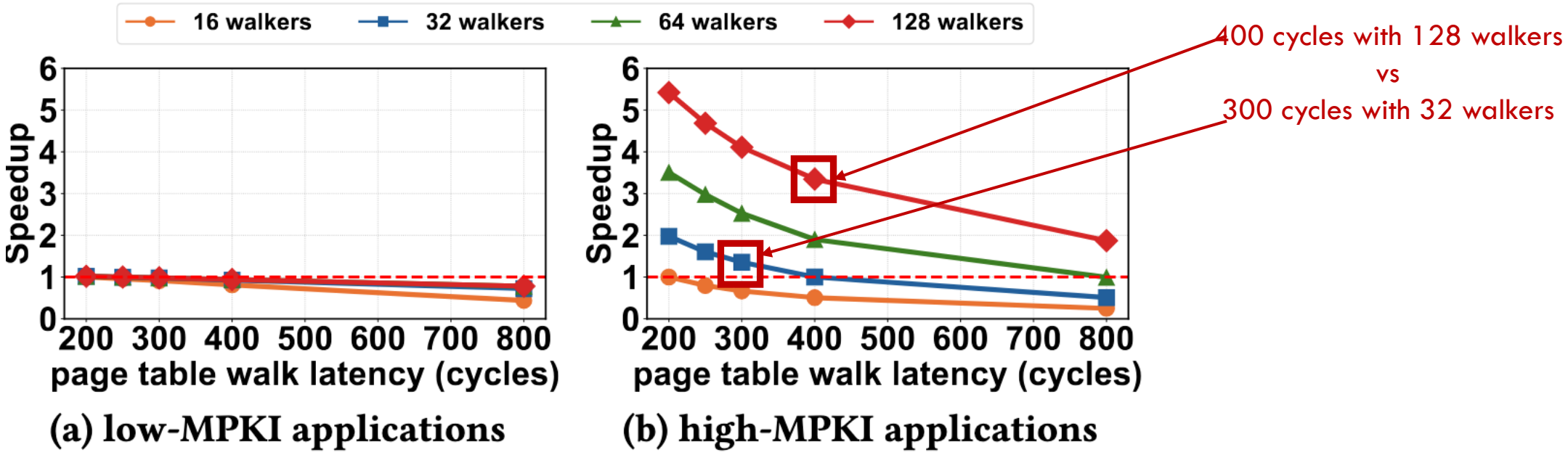


Speedup infinite page walkers versus ideal translation, normalized to the baseline GPU.

Takeaway: Page table walk throughput becomes a critical performance bottleneck for workloads exhibiting a high L3 TLB Miss-per-Kilo-Instructions (MPKI).

- Infinite Walker: Unlimited page-table walk throughput, but page-walk latency remains.
- Ideal Translation: Zero address-translation overhead (perfect translation).

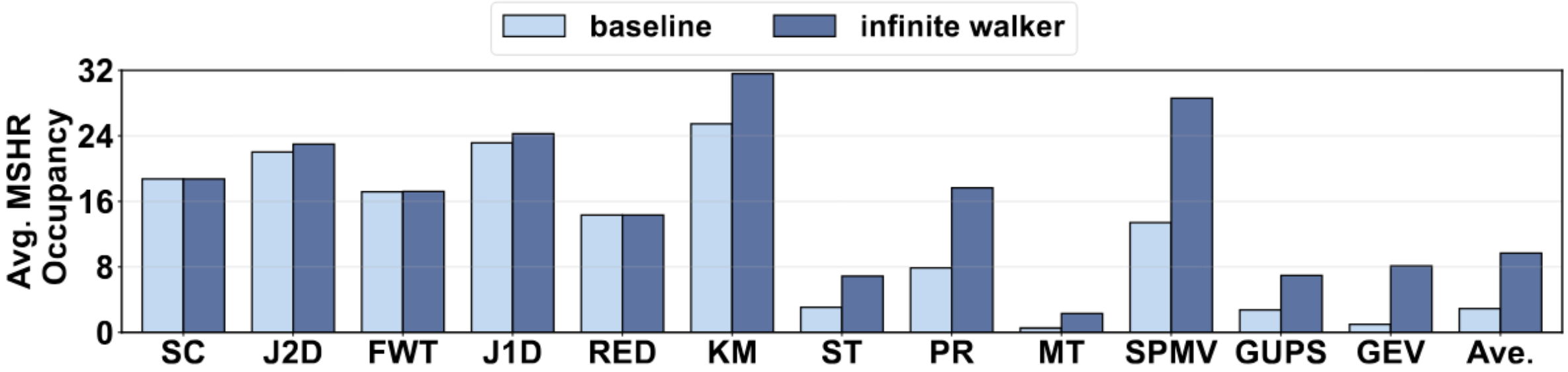
Motivation #2: Tradeoff between Throughput and Latency



Performance speedup across varying page table walker counts and walk latencies, normalized to a baseline of 16 walkers and 200-cycle latency.

Takeaway: Increases in page table walk latency are tolerable if they are compensated for by a substantial gain in concurrency.

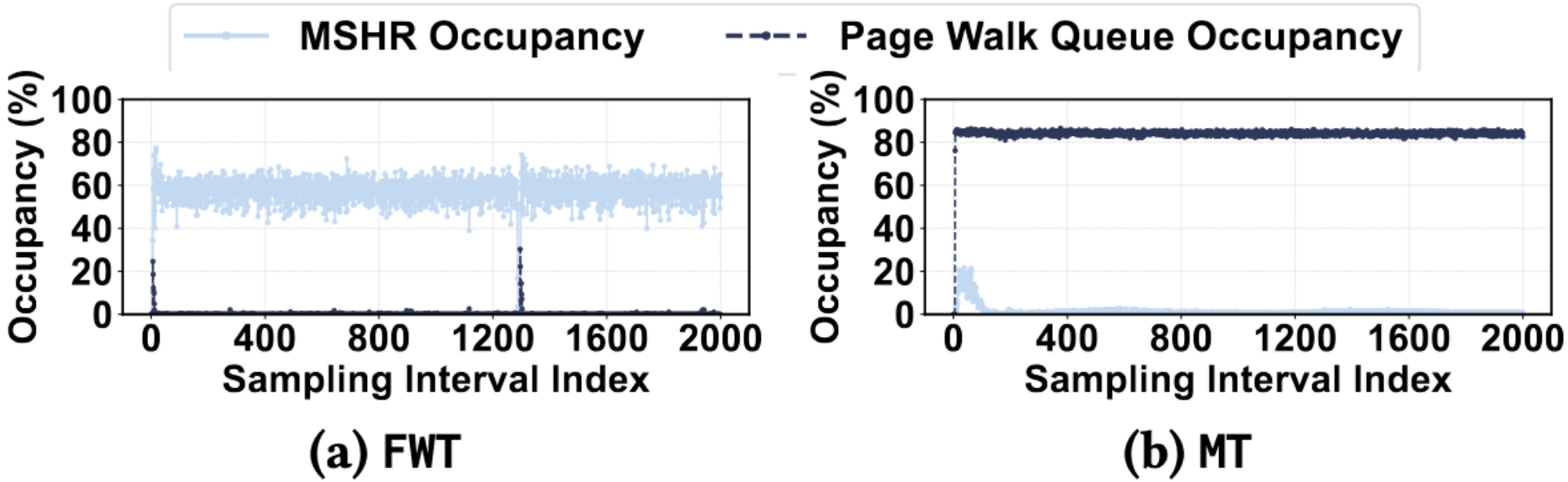
Motivation #3: Underutilized L1 Cache MSHRs



Average L1 MSHR occupancy across different benchmarks, assuming a 32-entry L1 MSHR configuration.

Takeaway: For applications with high MPKI, the address translation bottleneck often underutilizes the L1 data cache, as the compute units are frequently stalled awaiting TLB resolutions.

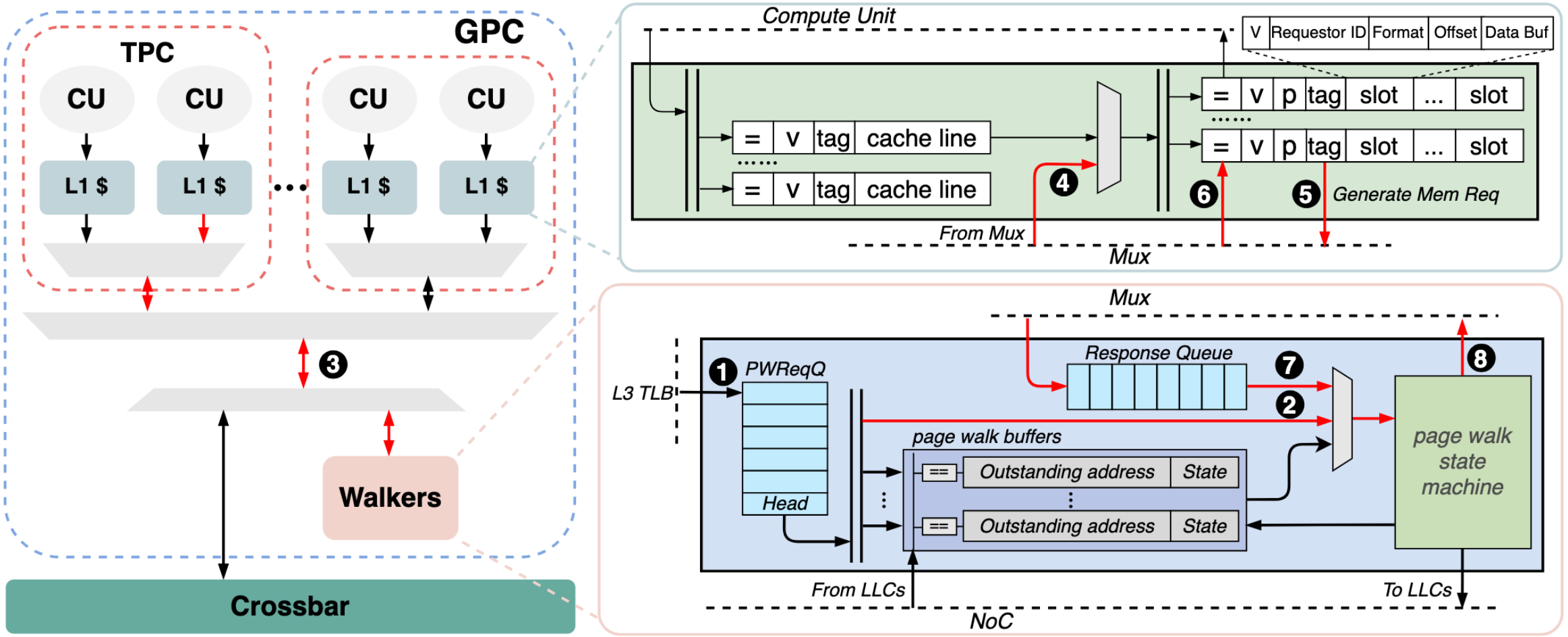
Motivation #3: Underutilized L1 Cache MSHRs



Dynamic resource occupancy of L1 MSHRs vs. PWQs. Sampling over 1,000-cycle intervals.

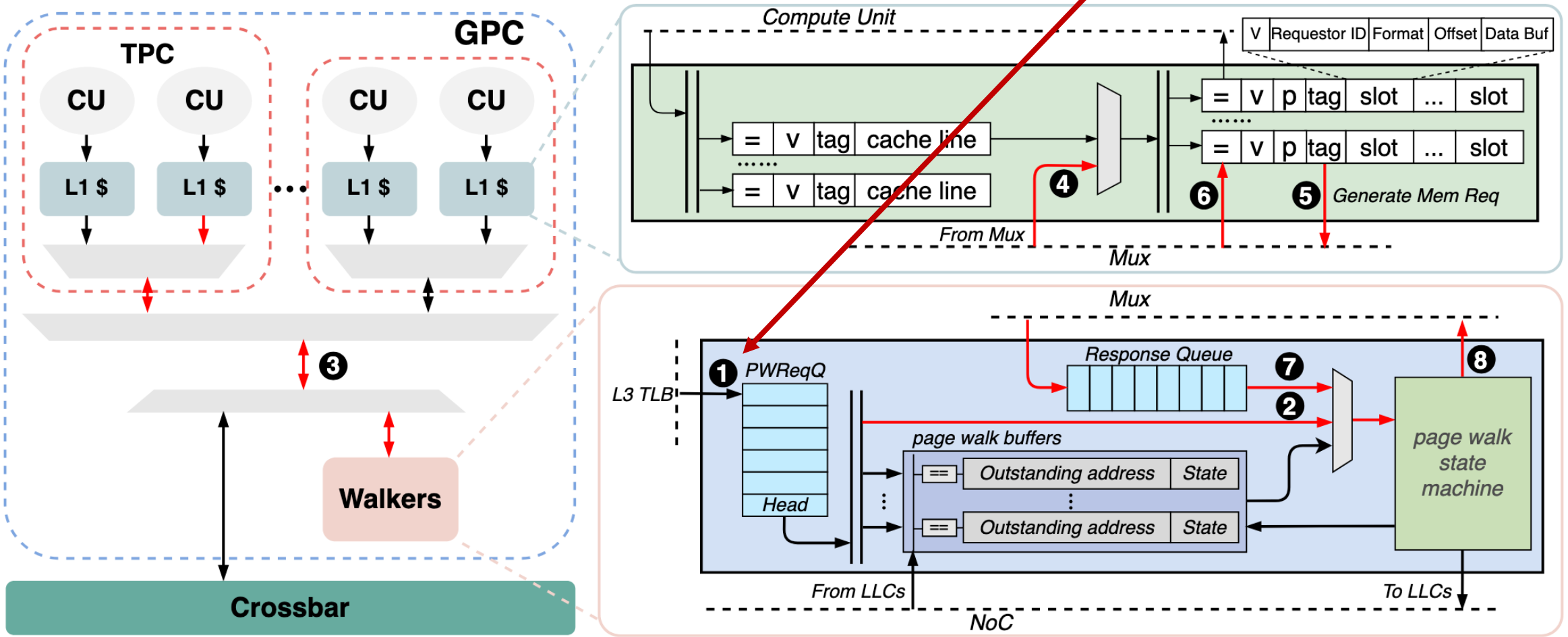
Takeaway: For applications with high MPKI, the address translation bottleneck often underutilizes the L1 data cache, as the compute units are frequently stalled awaiting TLB resolutions.

NB-Walker Overview



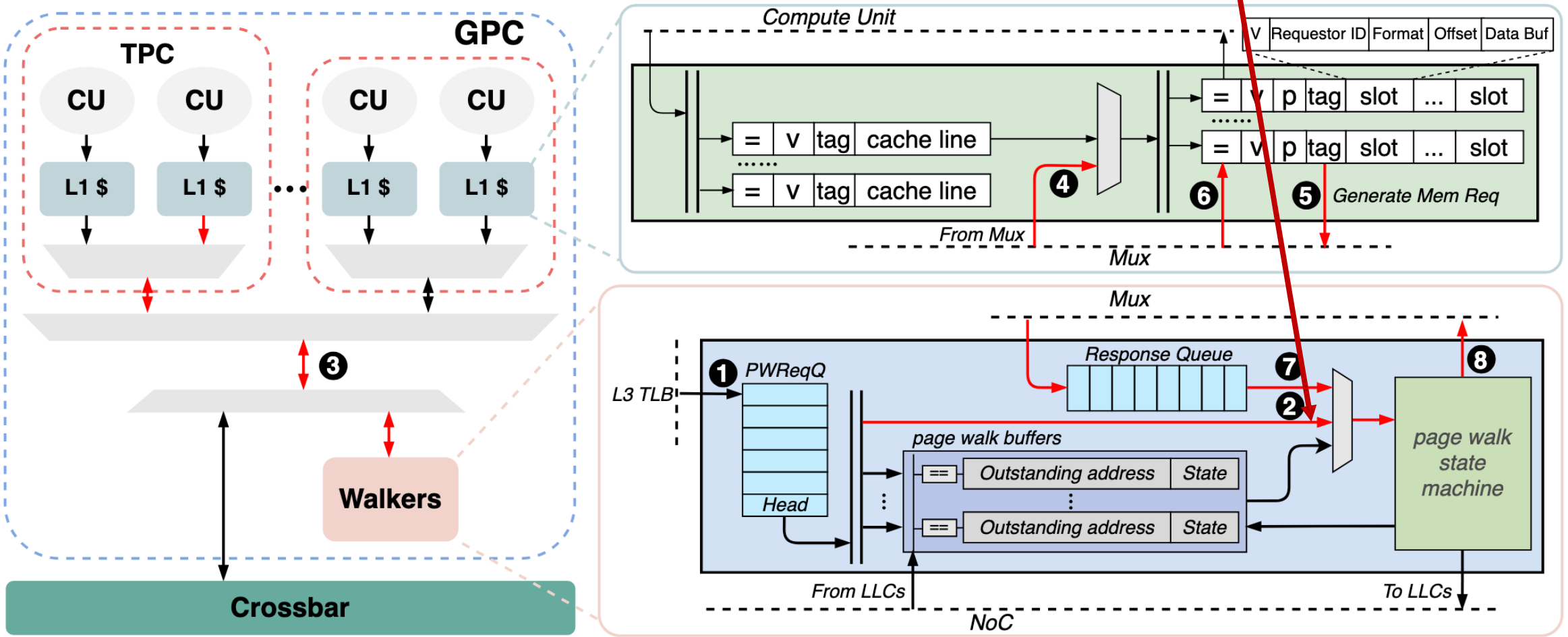
NB-Walker Workflow

1. Push Req into Page Walk Request Queue



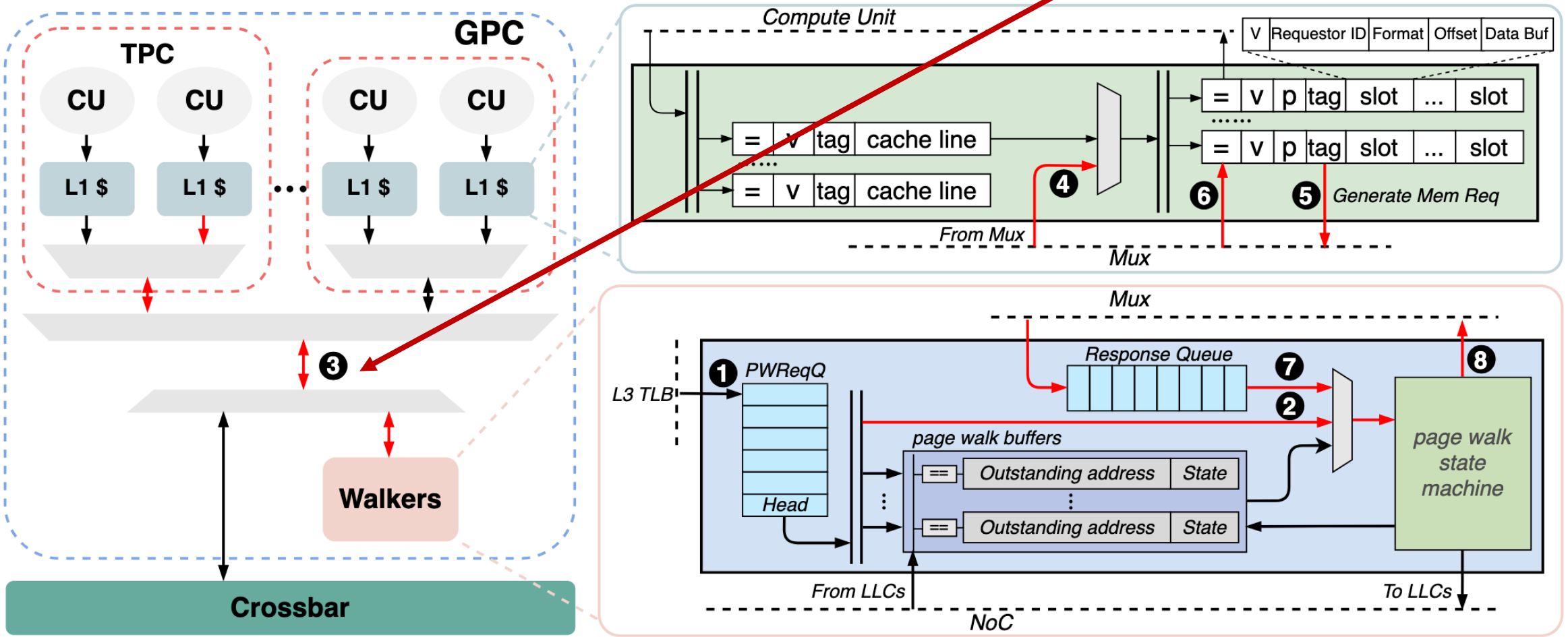
NB-Walker Workflow

2. Push Req into Finite State Machine



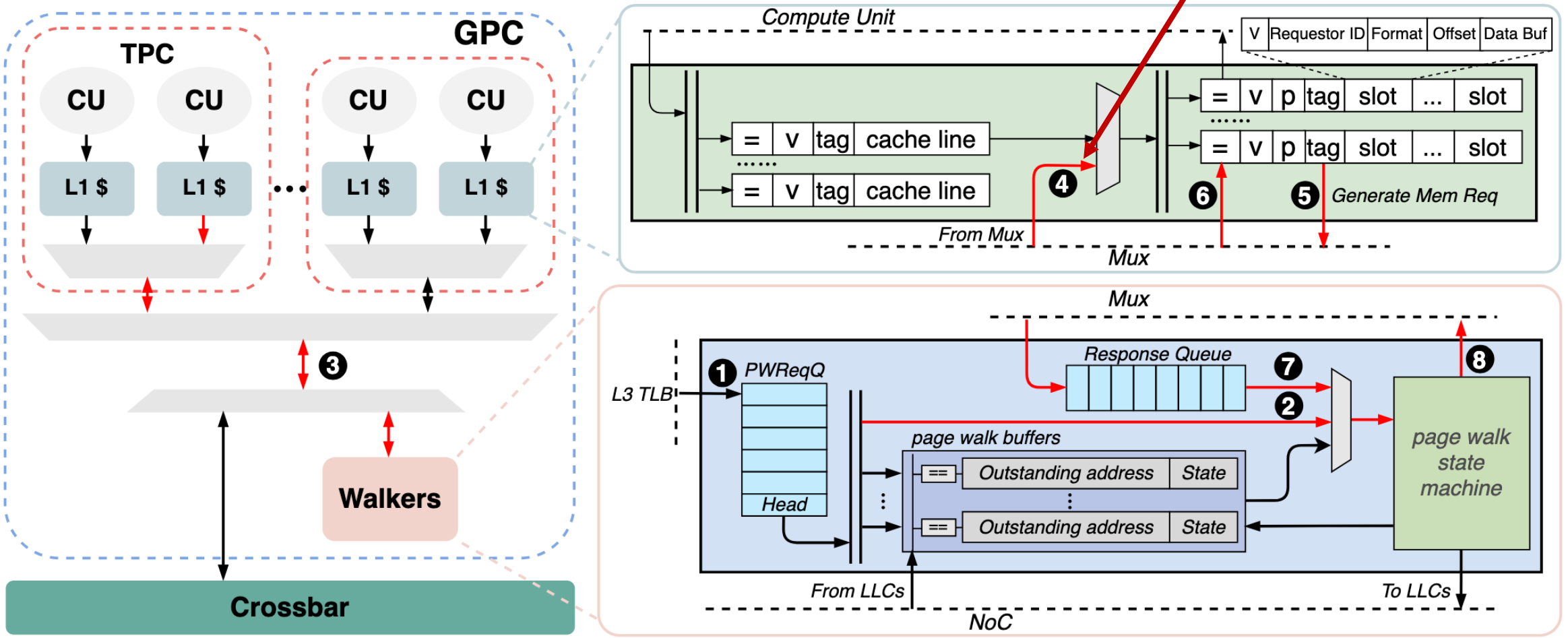
NB-Walker Workflow

3. Send Walk Metadata and Computed Physical Address to L1$ via Mux



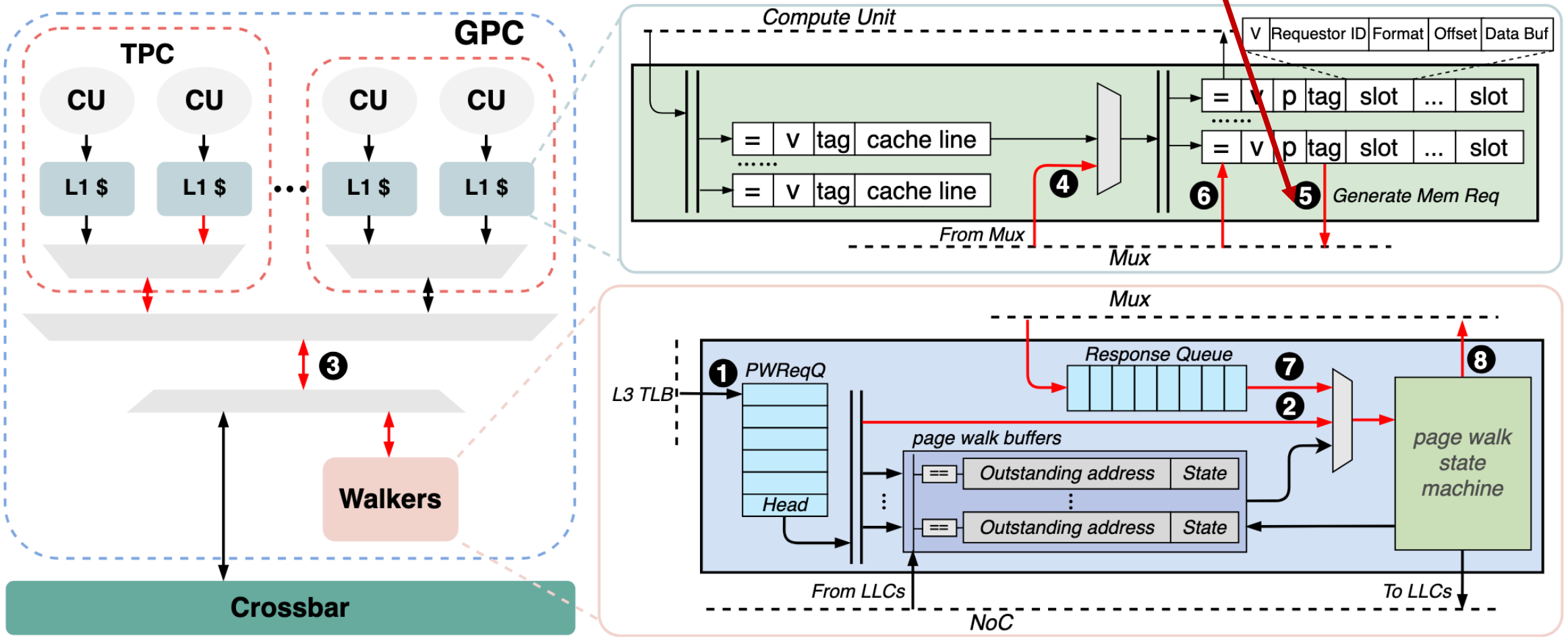
NB-Walker Workflow

4. Bypass Cache Block Lookup and Find Free MSHR Entry

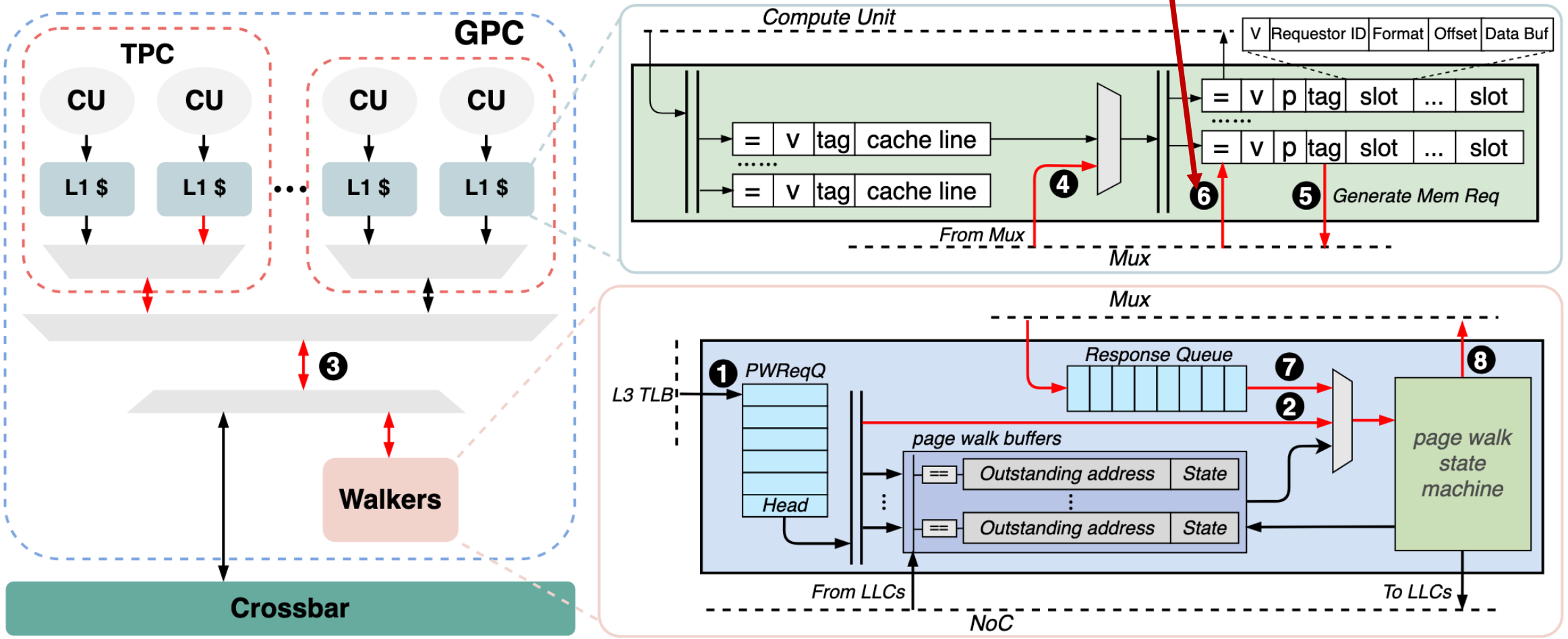


NB-Walker Workflow

5. Store Walk Metadata and Issue Memory Request

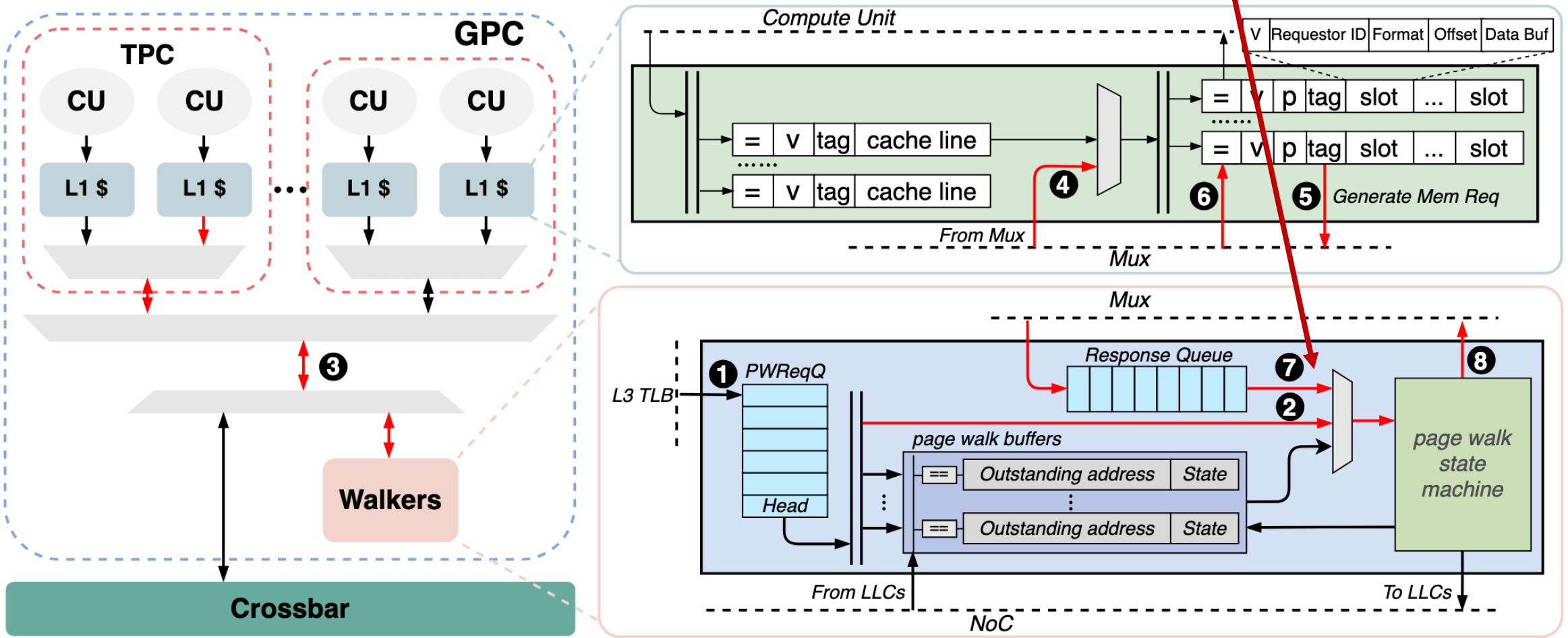


NB-Walker Workflow

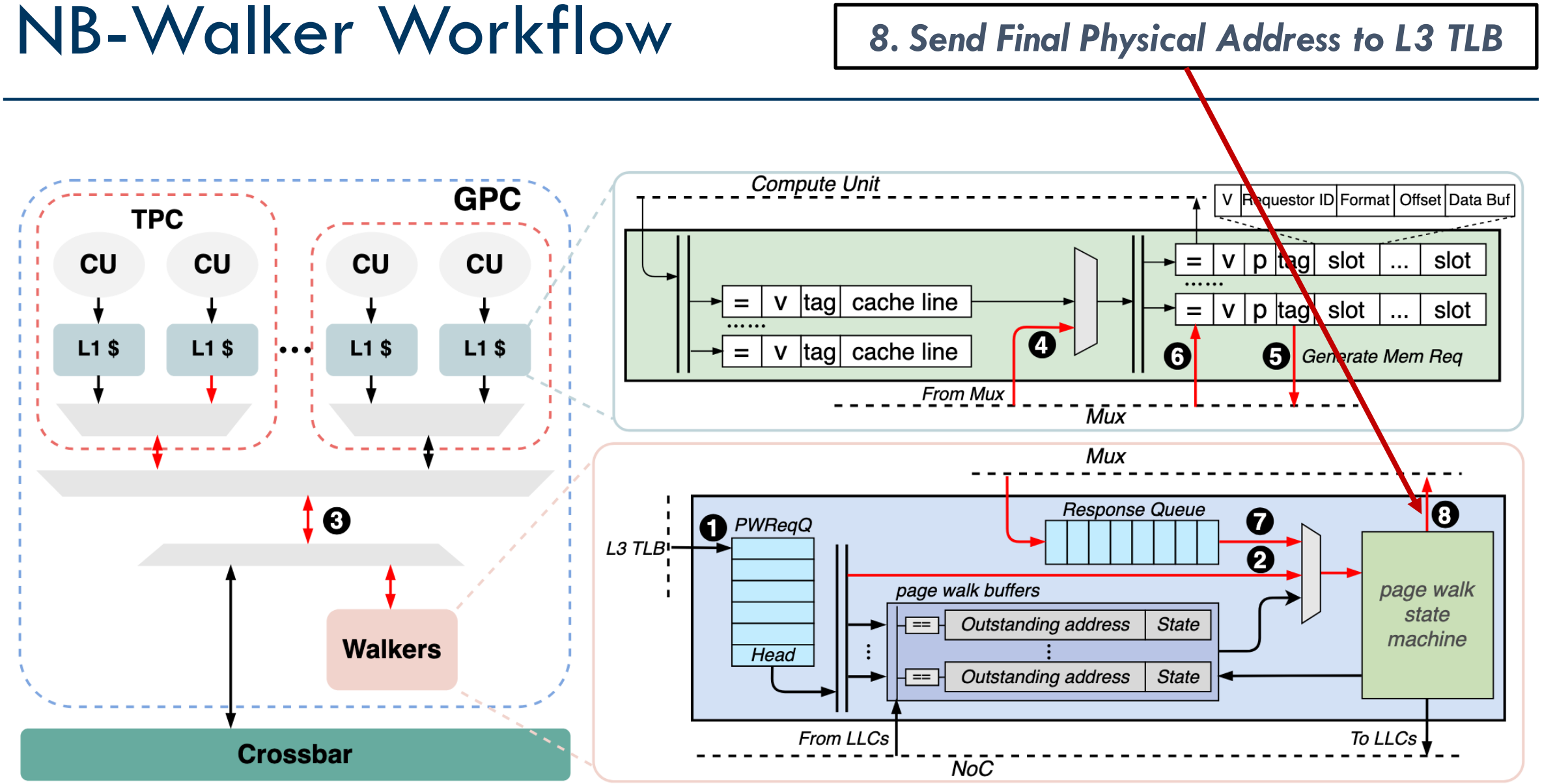


NB-Walker Workflow

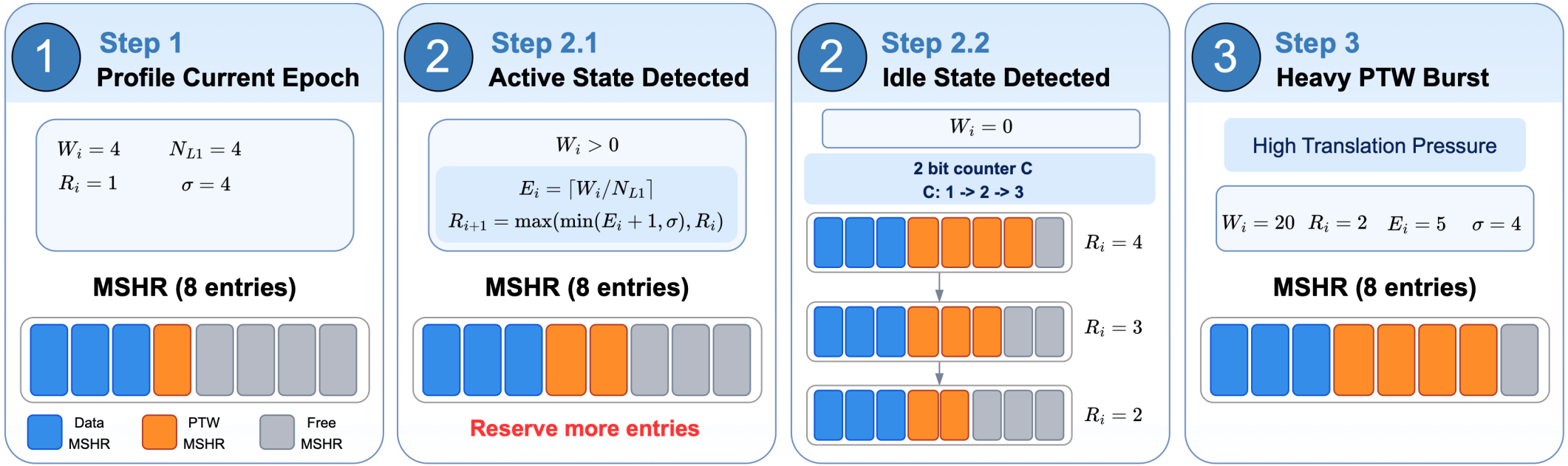
7. Push Response into Finite State Machine



NB-Walker Workflow



AMR: Adaptive MSHR Reservation



Key Idea: Reserve just enough MSHRs for PTWs based on recent demand and release them when walks are idle.

Benefit: Prevent data cache misses from starving translation, sustaining throughput under memory pressure.

Evaluation Methodology

Simulator: MGPUSim [ISCA 2019]

Module	Configuration
# of CU	1.0 GHz, 128 in total
CU	4 exec. units per CU, 64 threads per wavefront
DRAM	1 TB/s, 200 ns latency
L1 V-cache	64 KB, 16-way, 32 MSHR, 28 cycles, CU private
L1 S-cache	64 KB, 16-way, 32 MSHR, 28 cycles, shared b/w 4 CUs
L1 I-cache	64 KB, 16-way, 8 MSHR, 28 cycles, shared b/w 4 CUs
L2 cache	8 MB, 16-way, 160 cycles, GPU shared
L1 TLB	32 entries, fully associative, 20 cycles, CU private
L2 TLB	512 entries, 8-way, 40 cycles, GPC shared
L3 TLB	2048 entries, 8-way, 80 cycles, GPU shared
Page table walk	8 page table walkers, GPC shared [15], [17], [22]
Page walk cache	128 entries, 10 cycles, GPU shared [24], [46]

Evaluation Methodology

Workloads classified by L3 TLB MPKI into ***L**ow* and ***H**igh*

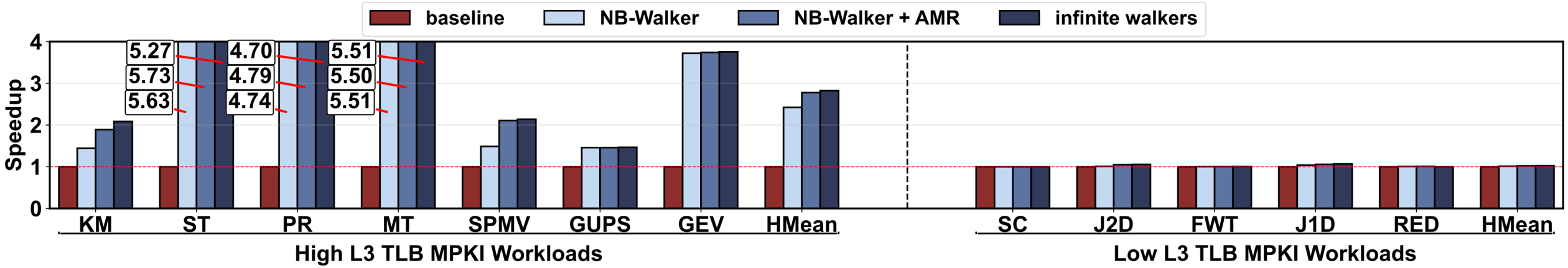
Abbr.	Application	Footprint (MiB)	MPKI L1 $	MPKI L3 TLB
SC	Simple Convolution on Matrix	2048	60.1	0.4
J2D	2-D Jacobi Solver	2048	183.3	2.2
FWT	Fast Warshall Transform	256	145.5	2.3
J1D	1-D Jacobi Solver	2048	211.5	3.2
RED	Reduction Kernel	2048	380.7	6.0
KM	K-Means Clustering	264	335.7	15.6
ST	Stencil Operation on 2-D Matrix	512	189.6	66.2
PR	Page Rank Algorithm	1024	439.5	93.5
MT	Matrix Transpose	512	219.2	191.8
SPMV	Sparse Matrix Vector Mult.	1390	5645.8	377.1
GUPS	Multi-threaded Random Access	256	1071.7	1122.5
GEV	Scalar, Vector and Matrix Multiplication	512	5906.0	5807.6

Evaluation Methodology

Configuration

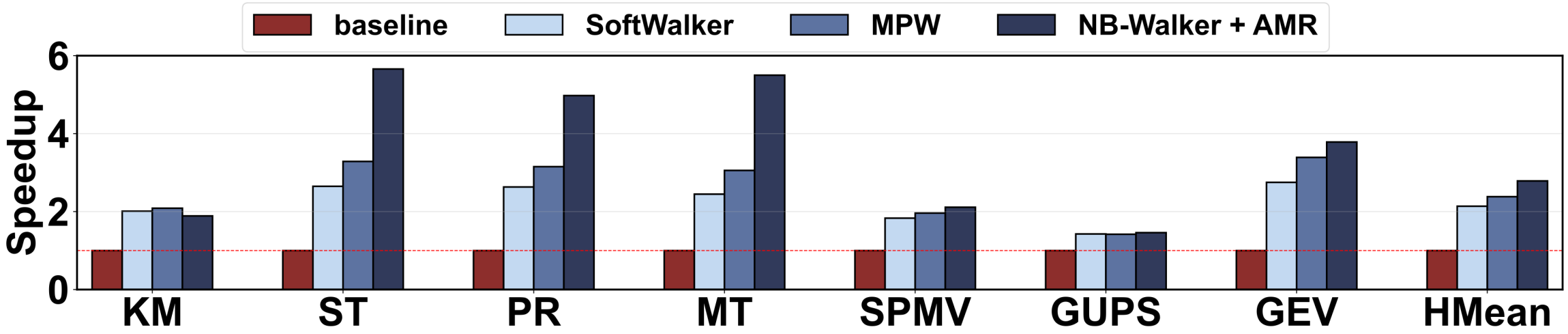
- **Baseline:** A conventional GPU that performs page table walks
- **NB-Walker:**
  - **NB-Walker:** basic proposal
  - **NB-Walker + AMR:** augments NB-Walker with Adaptive MSHR Reservation
- **Inf Walker:** Idealized Infinite Page Walkers

Performance Speedup



Key Result: NB-Walker and NBWalker+AMR achieve a 2.40x and 2.78x harmonic mean speedup over baseline GPU for high L3 TLB-MPKI workloads, respectively.

Comparison with State-of-the-Art



- **Baseline:** A conventional GPU that performs page table walks
- **MPW:** A SOTA hardware design that augments the PWQ to track walk metadata [HPCA 2024]
- **SoftWalker:** A SOTA hybrid design that offloads walk requests to compute units [MICRO 2025]

Key Result: NBWalker+AMR achieves an average speedup of 1.12x (up to 1.80x) and 1.30x (up to 2.25x) over MPW and SoftWalker, respectively.

Conclusion

- **Challenge:** Limited GPU page table walk throughput

- **Key Techniques:** Exploit idle L1 MSHRs to store walk metadata
  - Leverage underutilized L1 MSHRs to track walk requests
  - Dynamically reserve MSHR entries exclusively for PTW requests

- **Significant Performance Gains**
  - 2.78x, 1.30x and 1.12x harmonic mean speedup over baseline hardware page walker and two state-of-art page walker designs.

Thanks

NB-Walker: Non-Blocking Page Table Walker to Enhance Address Translation in NUMA GPUs

Zihang Chen <zchen097@connect.hkust-gz.edu.cn>,
Lieven Eeckhout, Hongyuan Liu, Tianao Ge, Xinkai Wang, Jiayi Huang

香港科技大学(广州)
THE HONG KONG UNIVERSITY OF SCIENCE AND TECHNOLOGY (GUANGZHOU)