Filtering Translation Bandwidth with Virtual Caching

Hongil Yoon Jason Lowe-Power* Gurindar S. Sohi

University of Wisconsin-Madison
* University of California, Davis

Executive Summary

Problem: **High translation bandwidth demand** for GPUs

- High serialization overhead at the shared address translation HW
- High performance degradation

Idea: Use **virtual cache** hierarchy as a ***bandwidth filter***!

- TLB misses hit in the GPU cache hierarchy
- Synonyms are very rare

Results:

- Virtual cache is practical and shows near ideal performance

Outline

Motivation and key observations

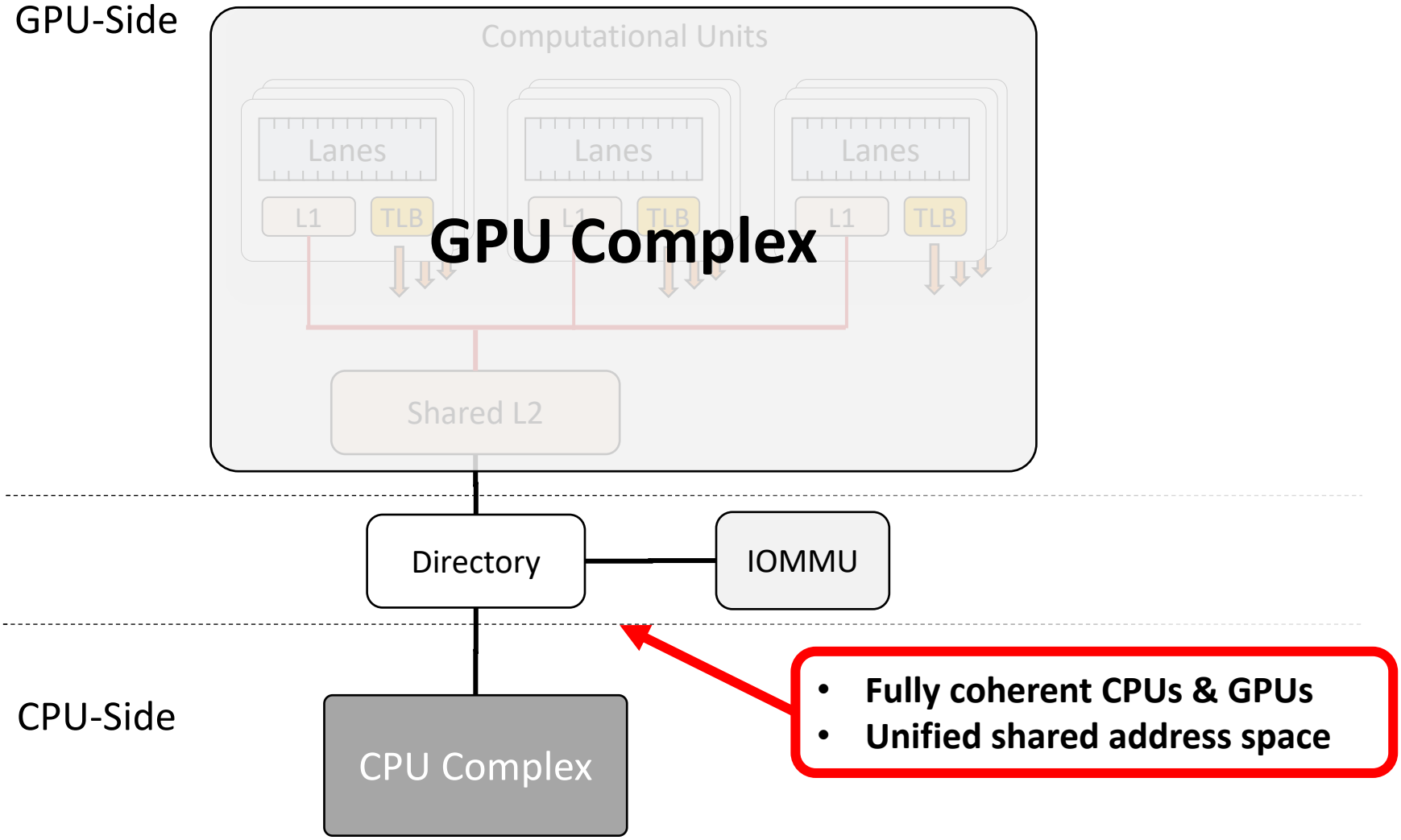
- GPU MMU design and its problem
- Filtering effect of GPU virtual caching

Design of GPU virtual cache hierarchy

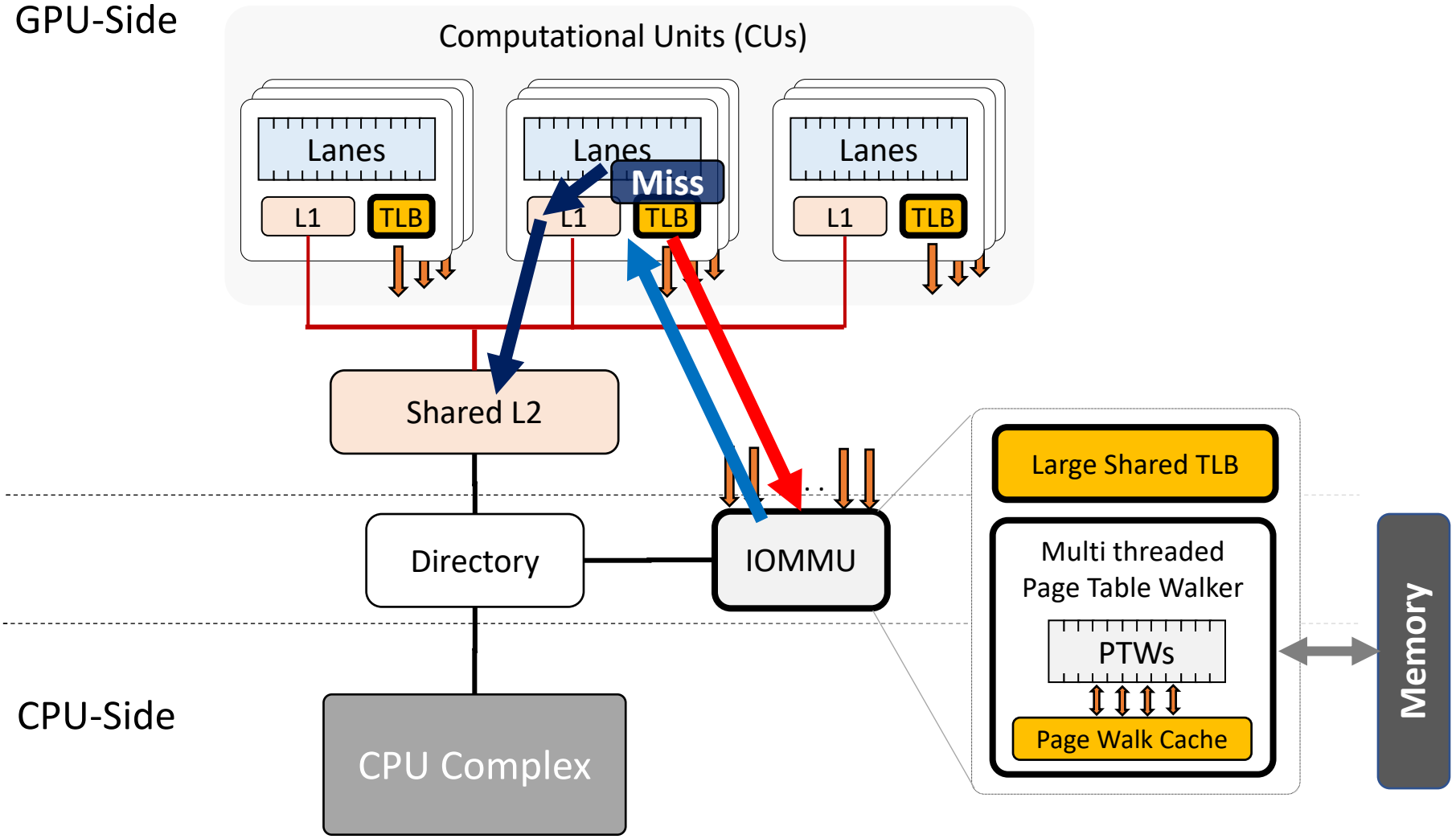
Evaluation

Summary & Conclusion

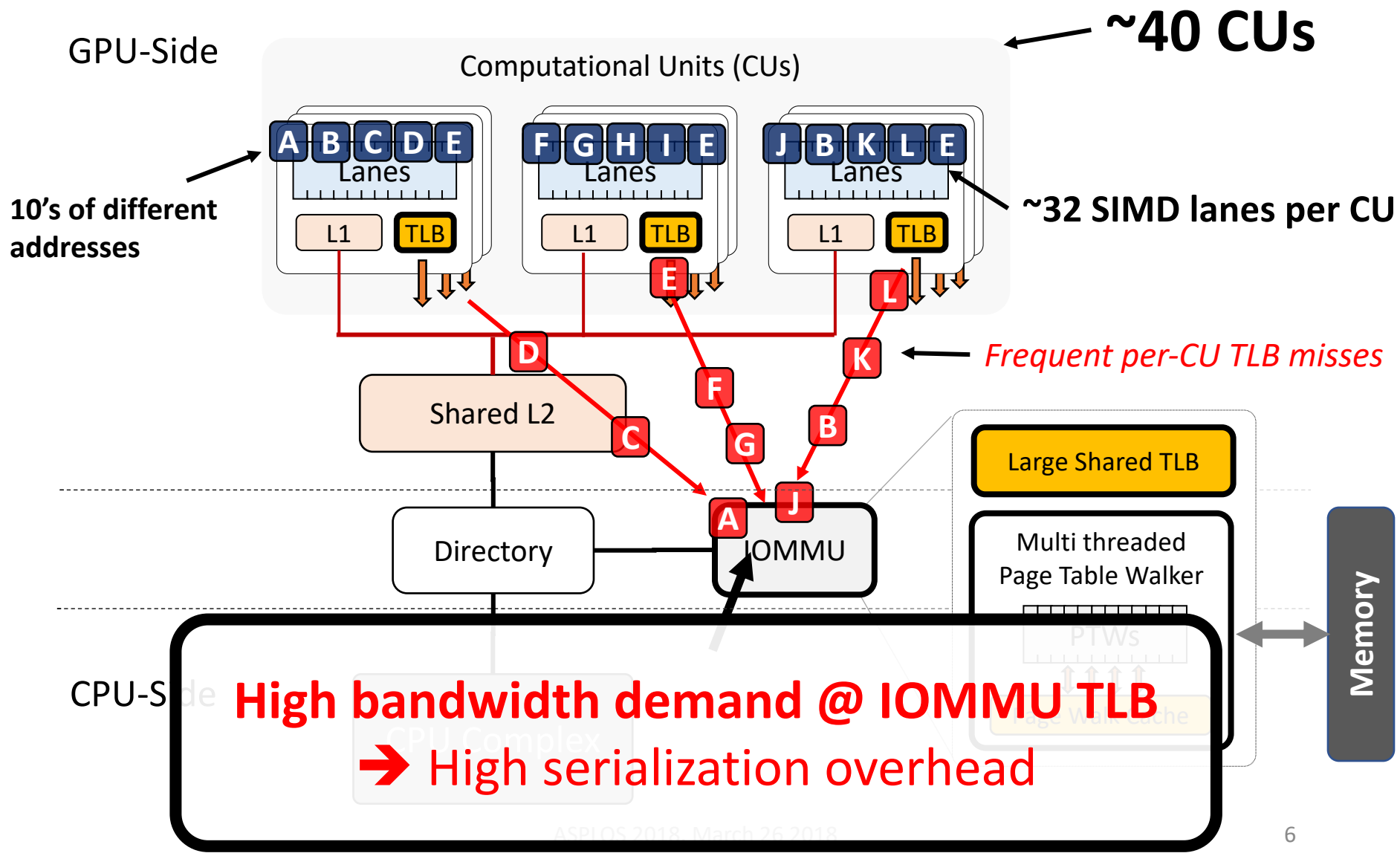
GPU Address Translation



GPU Address Translation



High translation bandwidth for GPUs



GPU address translation overhead

Observation 1: GPU workloads very frequently access the IOMMU TLB.

+1 access per cycle with high variation

Observation 2: The GPU shows high serialization overhead at the IOMMU TLB due to its limited bandwidth.

1.77x runtime overhead over an IDEAL MMU

Observation 3: GPU requires impractically high bandwidth to alleviate the serialization overhead at the IOMMU TLB.

More than **4 accesses per cycle**

GPU address translation overhead

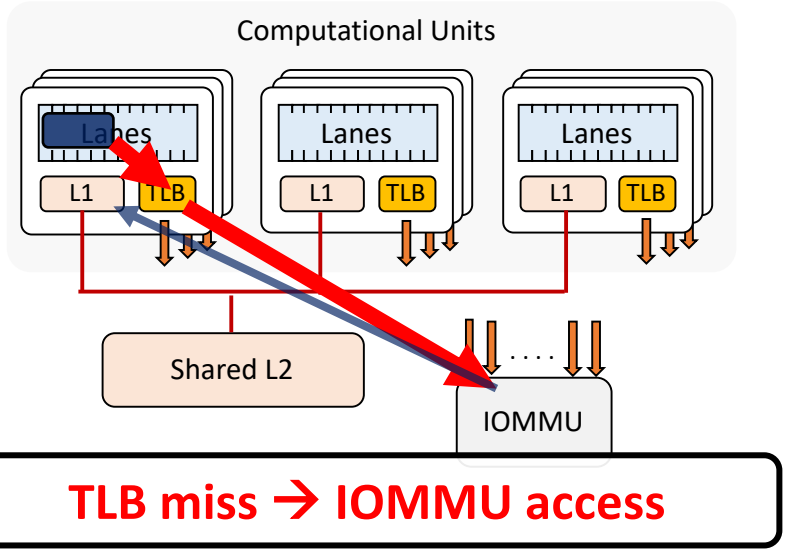
Challenge

How to **reduce the high bandwidth demand** at the **shared address translation** hardware

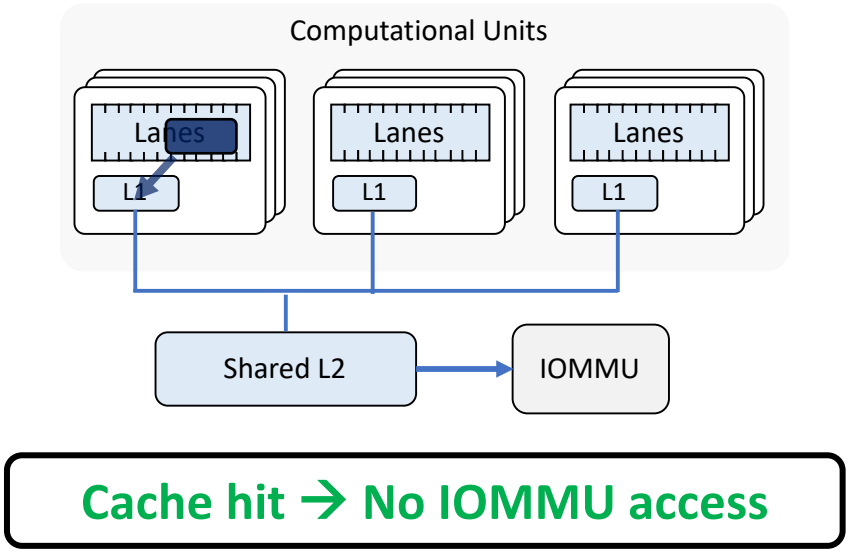
GPU Virtual Caching (aka VIVT cache)

- Use **a GPU VC hierarchy** as a **TLB miss bandwidth filter**
- **When?** Per-CU TLB miss, but hit in a cache hierarchy

Physically addressed cache



Virtual caching



GPU Virtual Caching (aka VIVT cache)

WHY ?

Blocks in cache likely to reside **longer than TLB entries**

Blocks in hierarchy from **many** pages

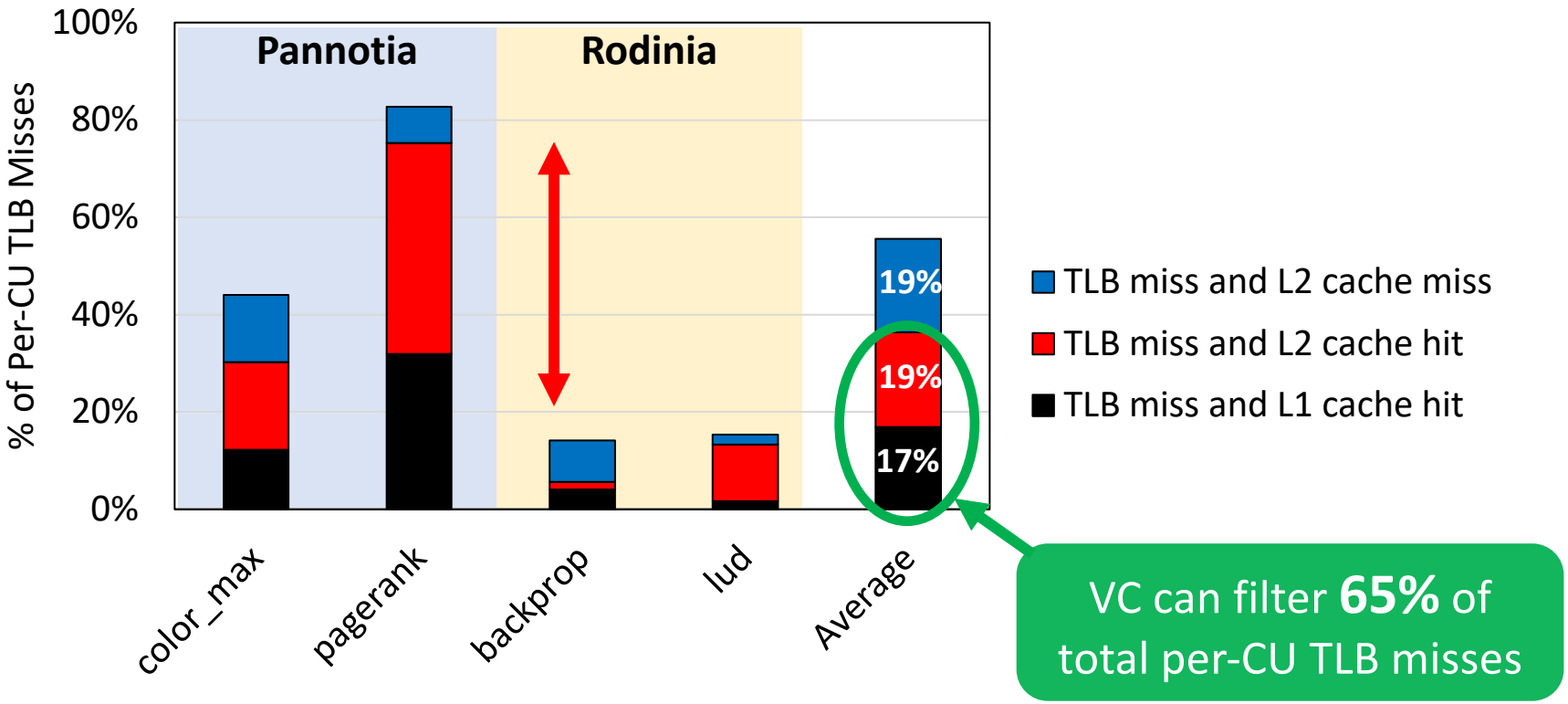
Cache increases address translation "**reach**"

Synonym are very rare

- **Virtual address synonyms** make virtual caches hard
  - In large CPU VCs: more “active” synonym accesses → more design complexity
- GPU is an accelerator: synonyms less likely.
  - Offload OS function calls to CPUs
  - Run a small number of applications at a time, etc
- Significant reduction in design complexities of GPU VC

Observation 4: Practical to use virtual caches throughout GPU hierarchy

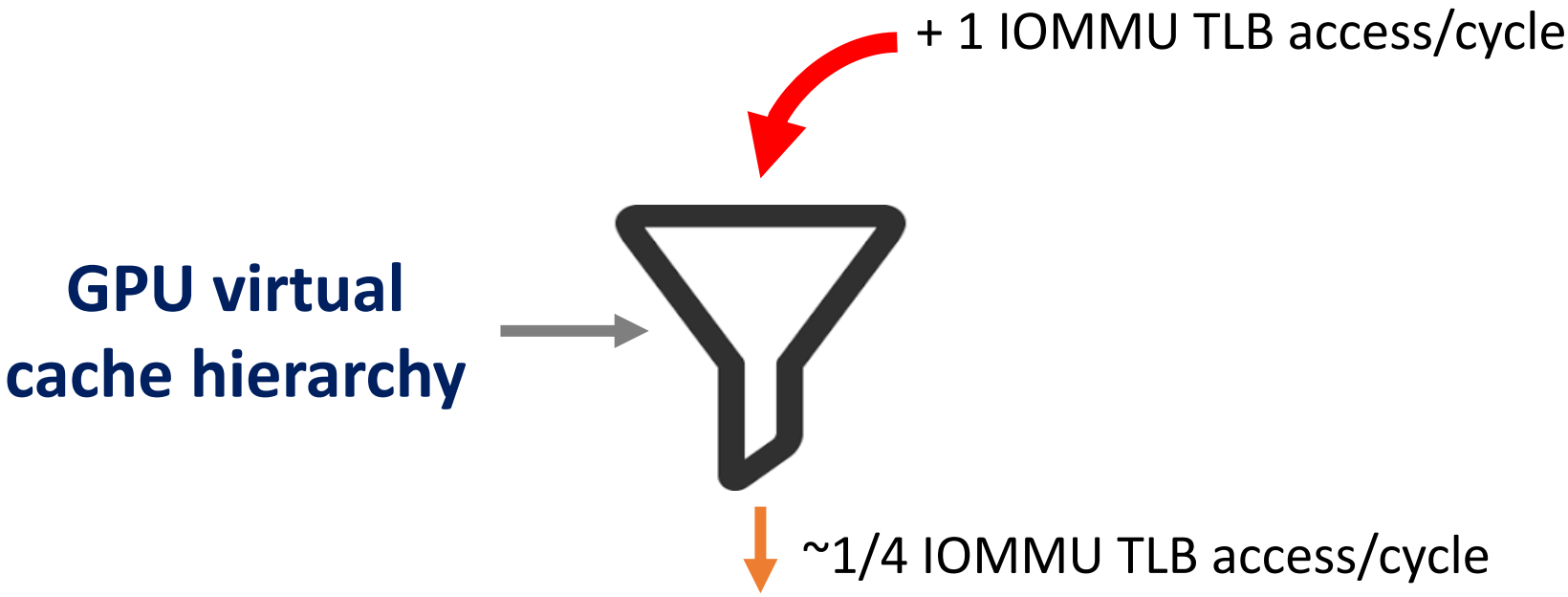
VC effective TLB miss filter



Breakdown of per-CU TLB misses (32-Entry TLB, 4KB Page)

Observation 5: A large fraction of TLB misses find data in GPU caches.

Solution: GPU Virtual Caching



Huge opportunities

Performance and energy

Outline

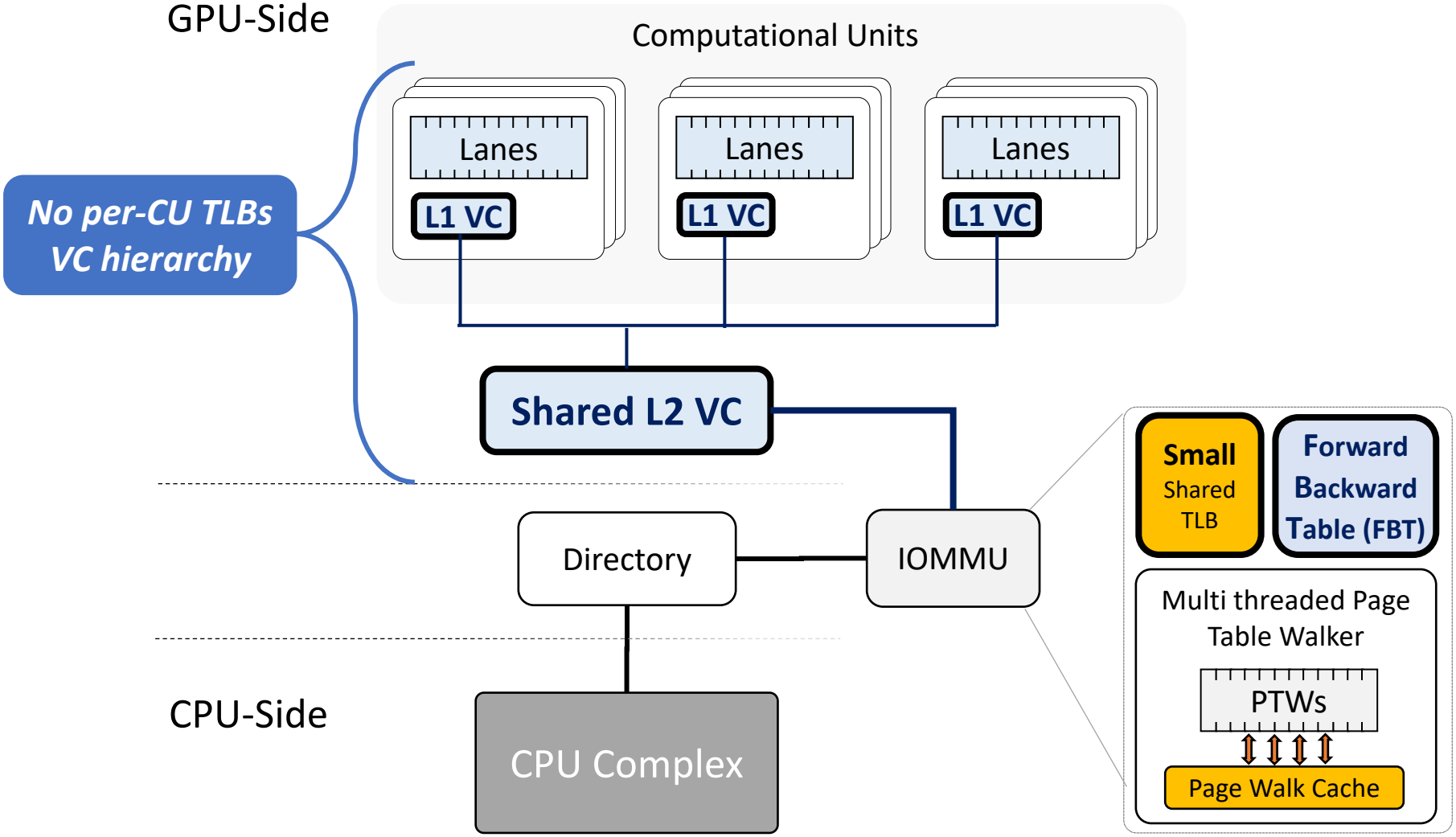
Motivation and key observations

Design of GPU virtual cache hierarchy

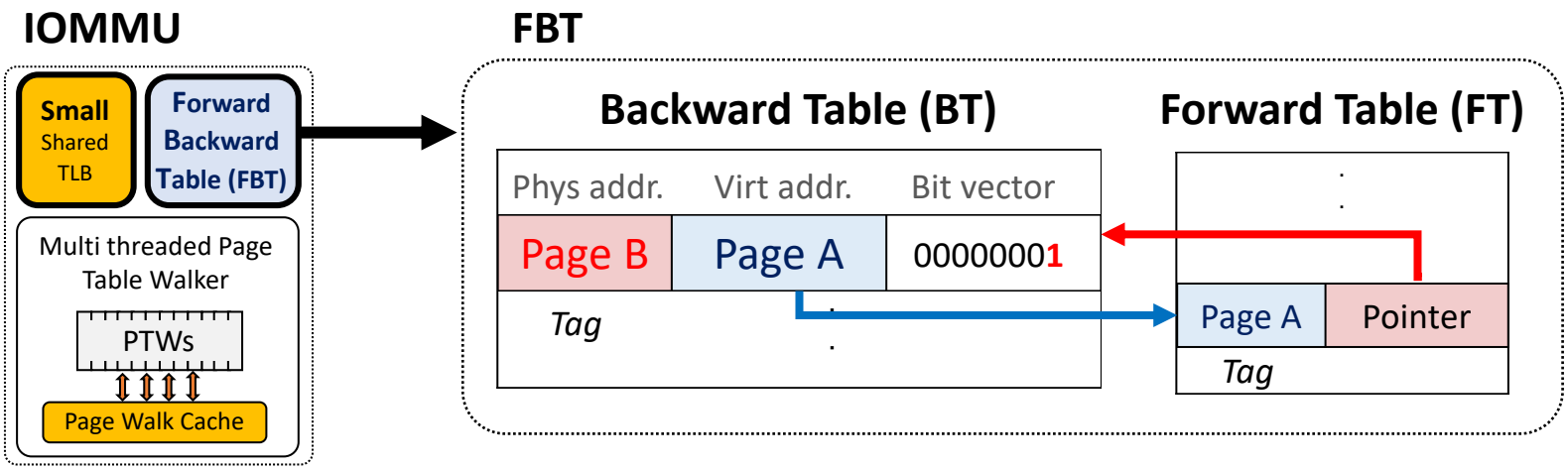
Evaluation

Summary & Conclusion

GPU Virtual Cache Hierarchy



Forward-Backward Table



Properties:

- Fully inclusive of current cache data in VCs
- Provide forward (V→P) and reverse (P→V) translations

Usages:

- Cache coherence, synonym detection, TLB shootdown, coherence msg. from GPUs etc
- Optimization: a second level shared IOMMU TLB

Outline

Motivation and key observations

Design of GPU virtual cache hierarchy

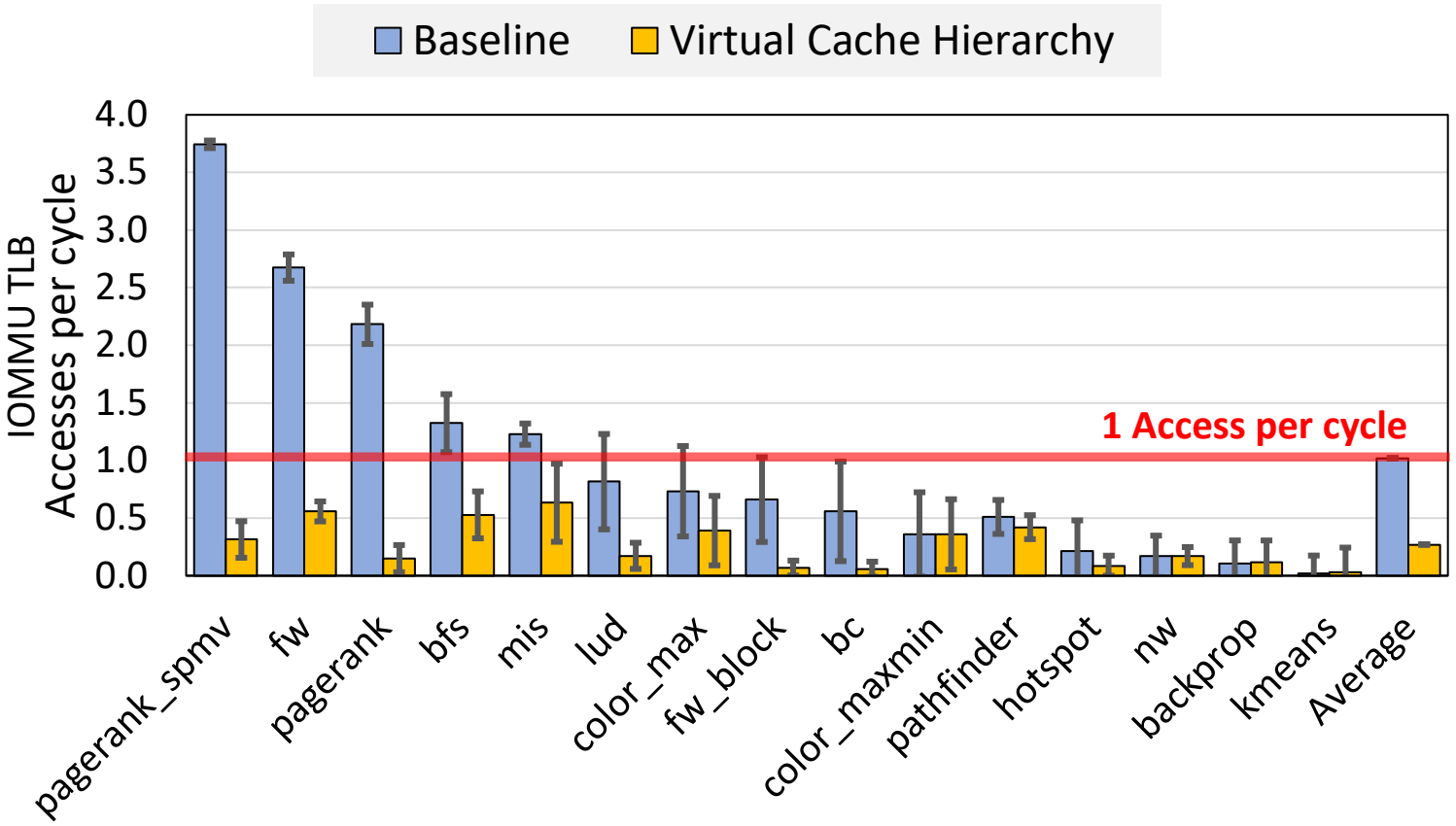
Evaluation

Summary & Conclusion

Simulation Methodology

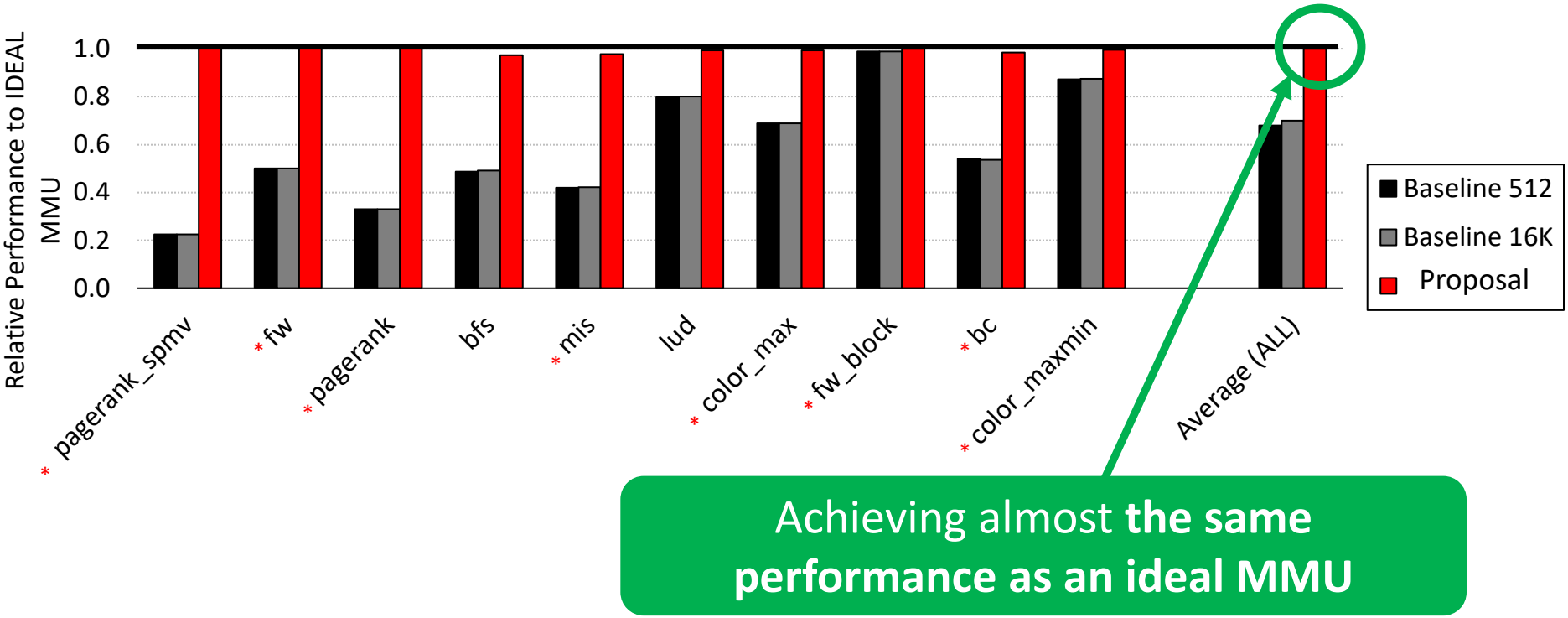
- Used gem5-gpu, **full system** simulation
- 16 CUs (medium-sized integrated GPU)
  - 32 lanes per each CU
  - 32/64/128 entry per-CU TLBs (4KB pages)
- IOMMU:
  - Previous work as baseline:
    16 concurrent page table walkers and 8KB page-walk cache
  - Size: 512/16K entry IOMMU TLB
  - **BW limit:** 1 access per cycle
- Workloads:
  - Pannotia (emerging graph-based workloads)
  - Rodinia (conventional workloads)

Bandwidth reduction of IOMMU TLB



GPU virtual cache hierarchy is an **efficient TLB miss BW filter**

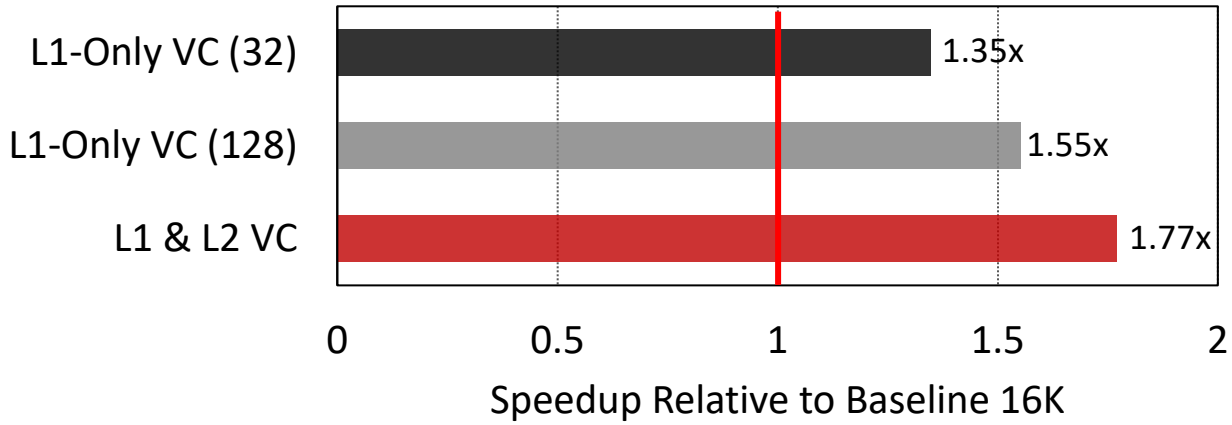
Performance Benefits



Achieving almost **the same performance as an ideal MMU**

Huge performance benefits by **filtering IOMMU TLB accesses**

Comparison with L1 VC design



L1-only VCs **<<<** **L1-L2 VC hierarchy** (additional speedup)

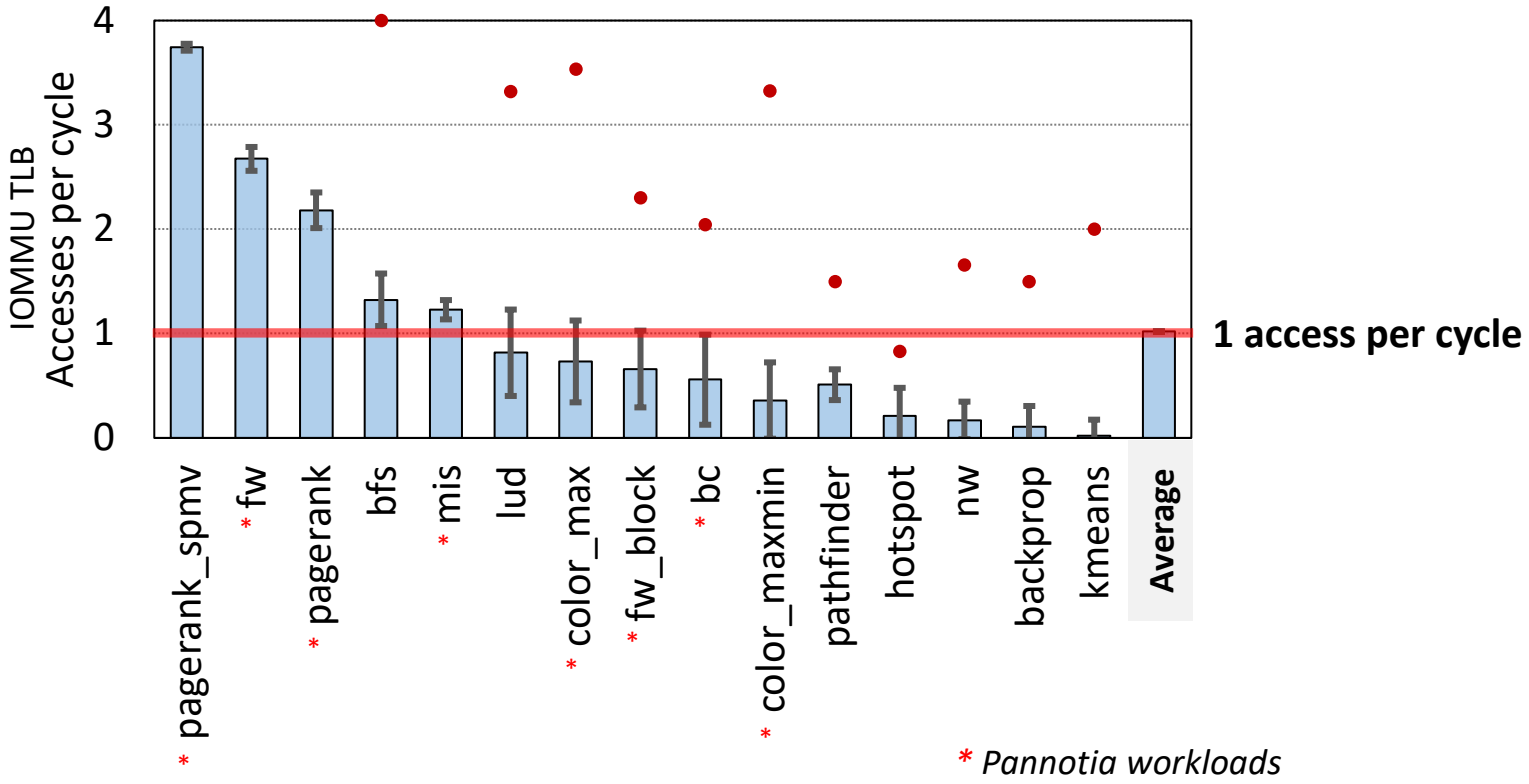
Summary & Conclusion

- **Problems**: **High translation bandwidth demand** for GPUs
- **Solution**: Use **virtual cache hierarchy as a *bandwidth filter***
- **Benefits**: achieve almost ideal performance as an ideal MMU

On-die accelerators finally provide an environment where **virtual caches** are both *practical and provide significant benefits*.

Questions?

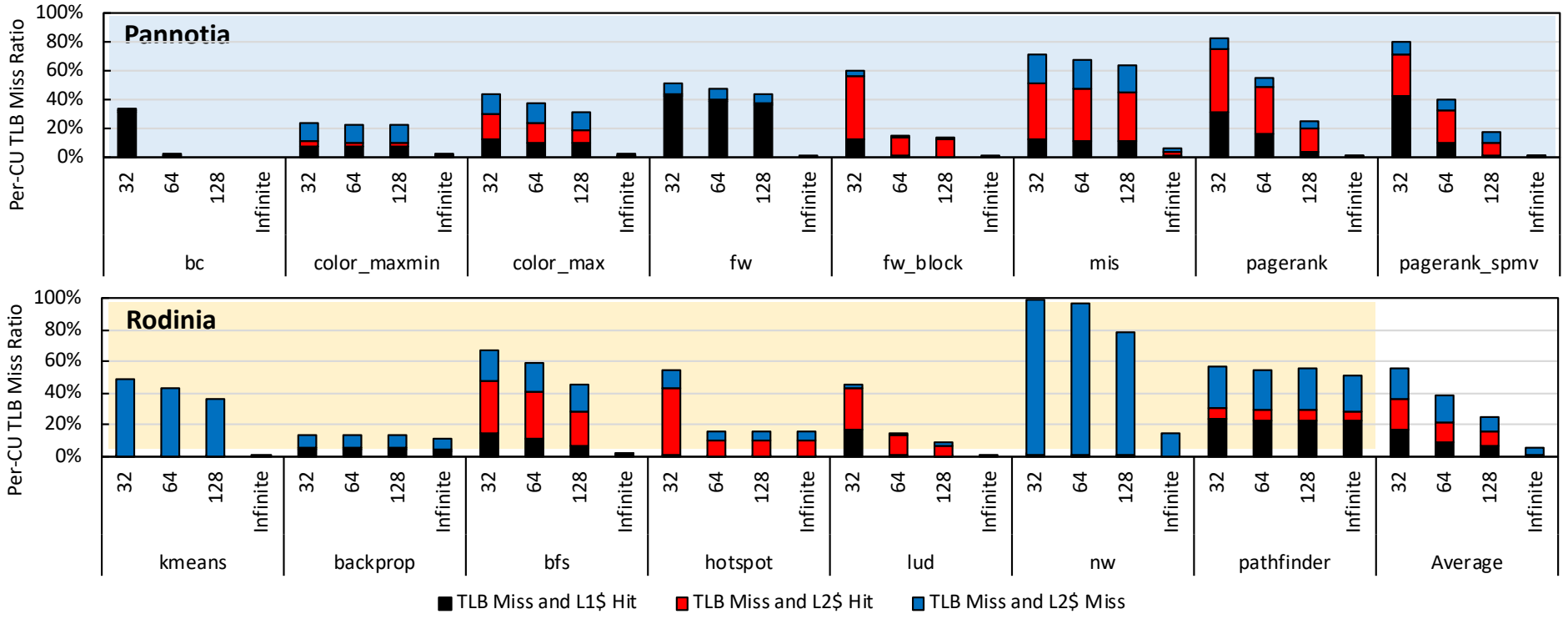
Frequent shared TLB accesses



IOMMU TLB accesses (i.e., *per-CU TLB misses for all 16 CUs*)

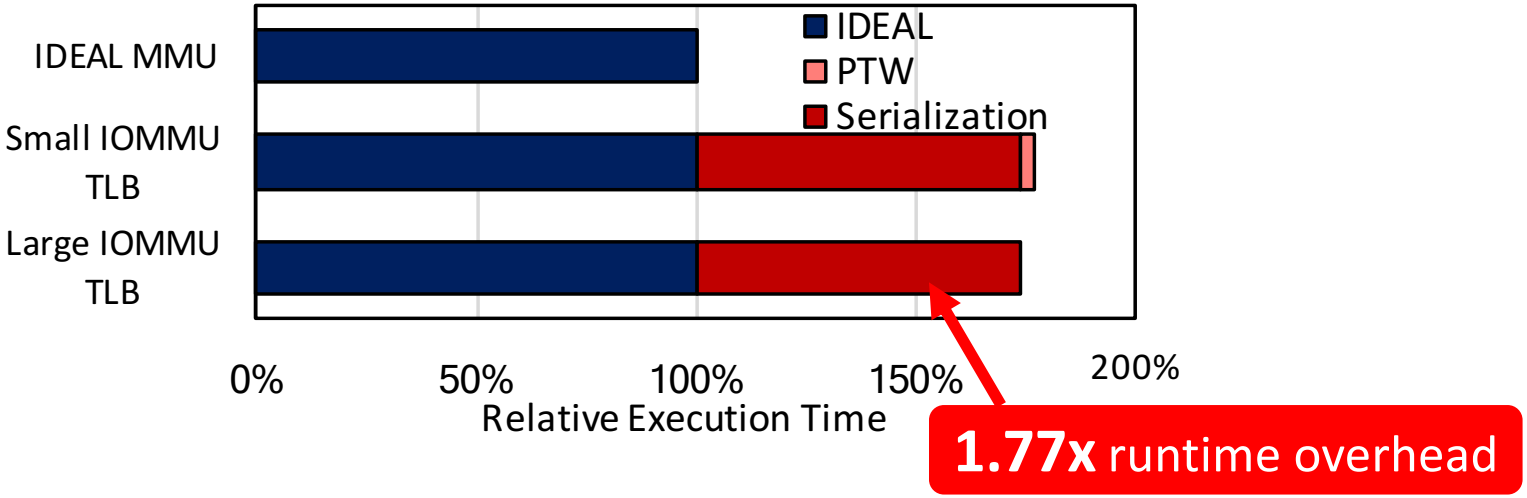
Observation: GPU workloads very frequently access the IOMMU TLB.

VC effective TLB miss filter



Observation: A large fraction of TLB misses find data in GPU caches.

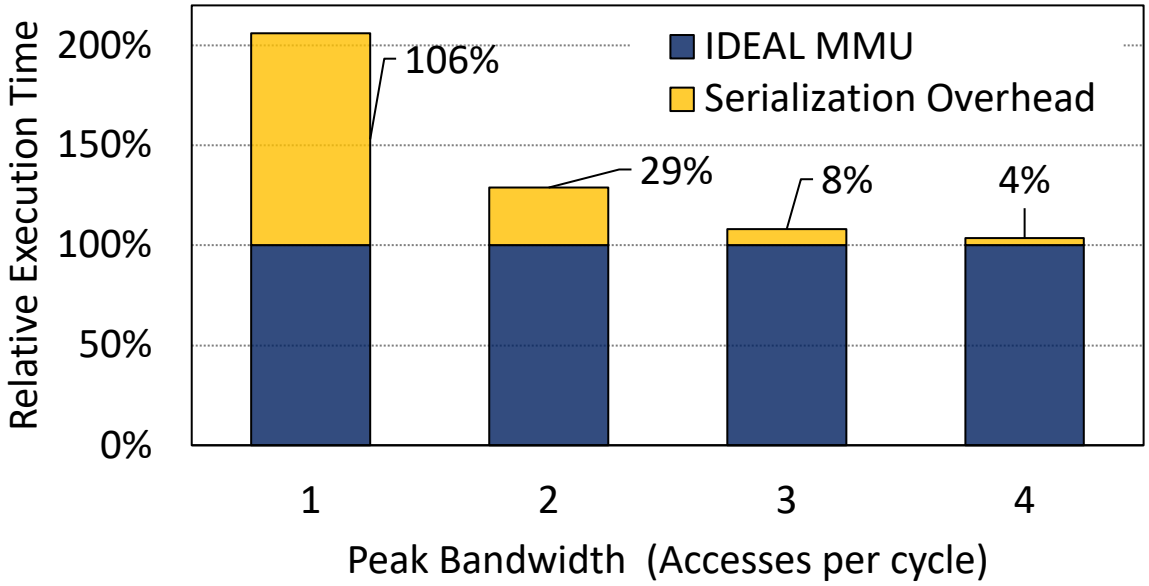
Serialization overhead @ IOMMU



Design	Per-CU TLB	IOMMU TLB	B/W Limit
IDEAL MMU	Infinite size	Infinite size	Infinite
Small IOMMU TLB	32-entry	512-entry	1 access/cycle
Large IOMMU TLB	32-entry	16K-entry	1 access/cycle

Observation: The GPU shows high serialization overhead at the IOMMU TLB due to its limited bandwidth.

Performance impact of BW limit



Impact of the IOMMU TLB bandwidth limit on the serialization overhead

Observation: GPU requires impractically high bandwidth to alleviate the serialization overhead at the IOMMU TLB.

Comparison with Larger per-CU TLB