Access Pattern-Aware Cache Management for Improving Data Utilization in GPU

Gunjae Koo[*], Yunho Oh[†], Won Woo Ro[†], Murali Annavaram[*]

[*]University of Southern California

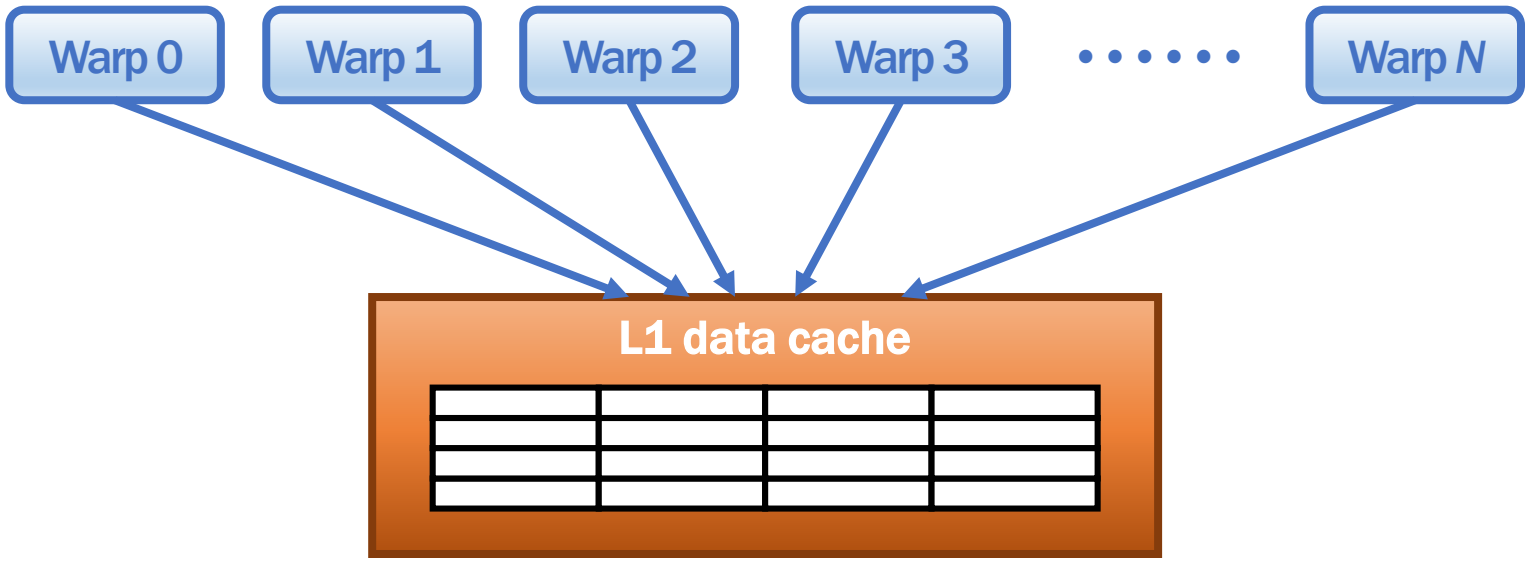
[†]Yonsei University

USC University of Southern California

YONSEI UNIVERSITY

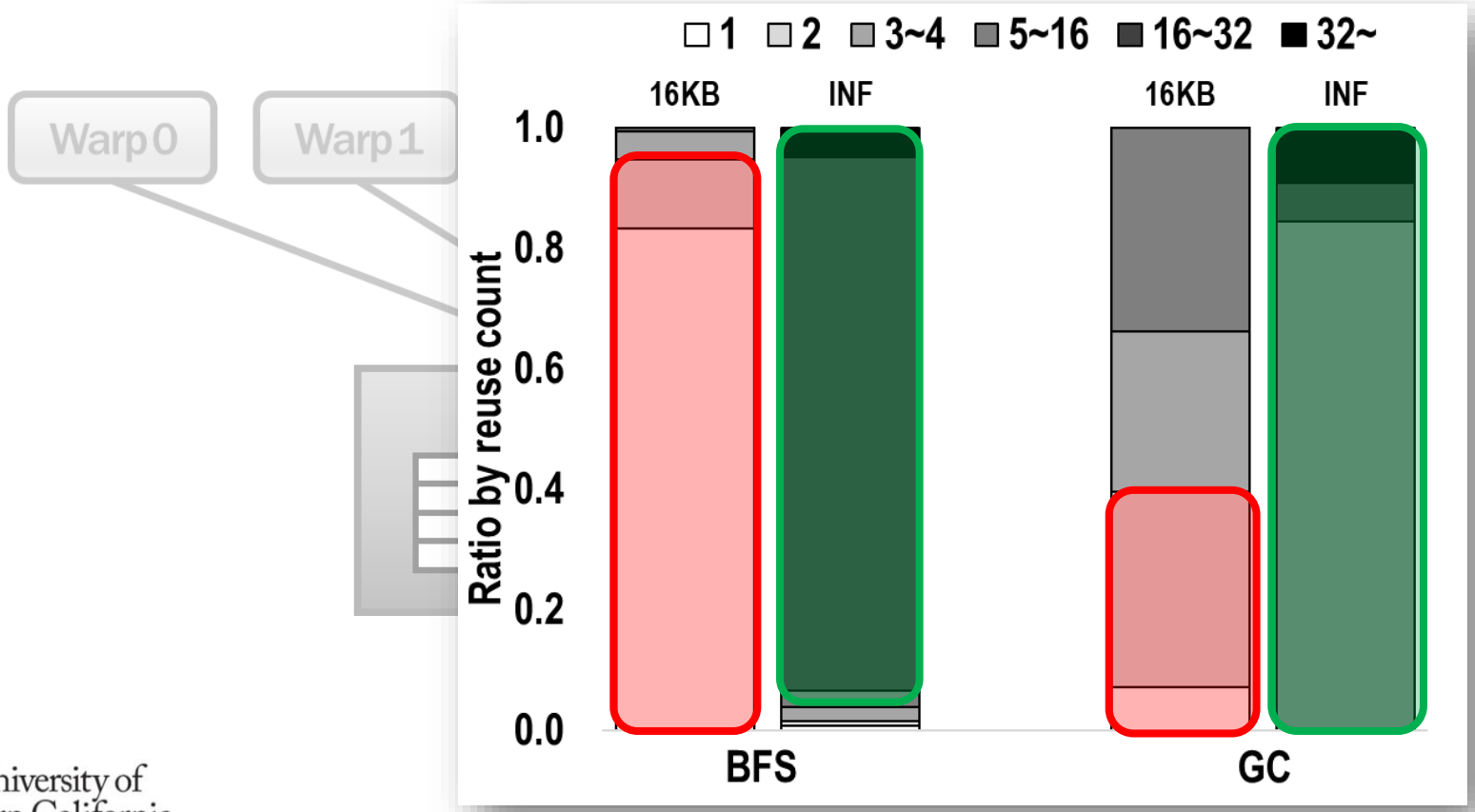
Motivation

- Poor data reuse in cache
  - Small cache size compared to number of warps
  - Cache lines are evicted frequently before re-reference
  - Data locality is not utilized efficiently due to cache size limit



Poor Cache Reuse ≠ Lack of Locality

- Poor cache reuse is not due to lack of locality
  - If given enough cache size multiple re-uses of a cache line are possible
  - But default GPU cache size leaves virtually no opportunity for reuse



Prior Warp-level Solutions

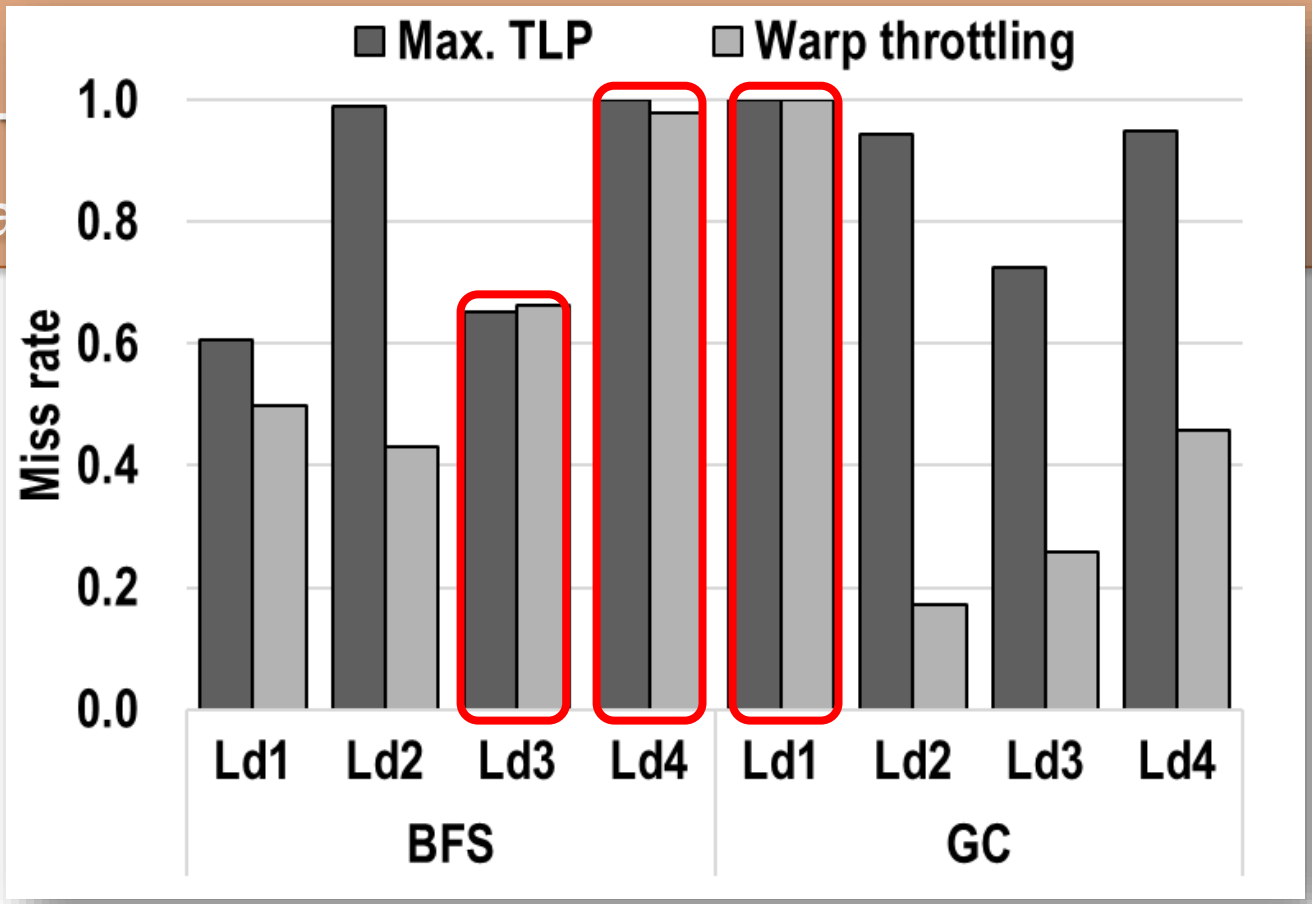
- Blocks or bypasses warps → reduces cache contention

- Sacrifices TLP
- Bypassed warps → even warp loads with locality are penalized
- Not effective for all types of global loads

Prior Warp-level Solutions

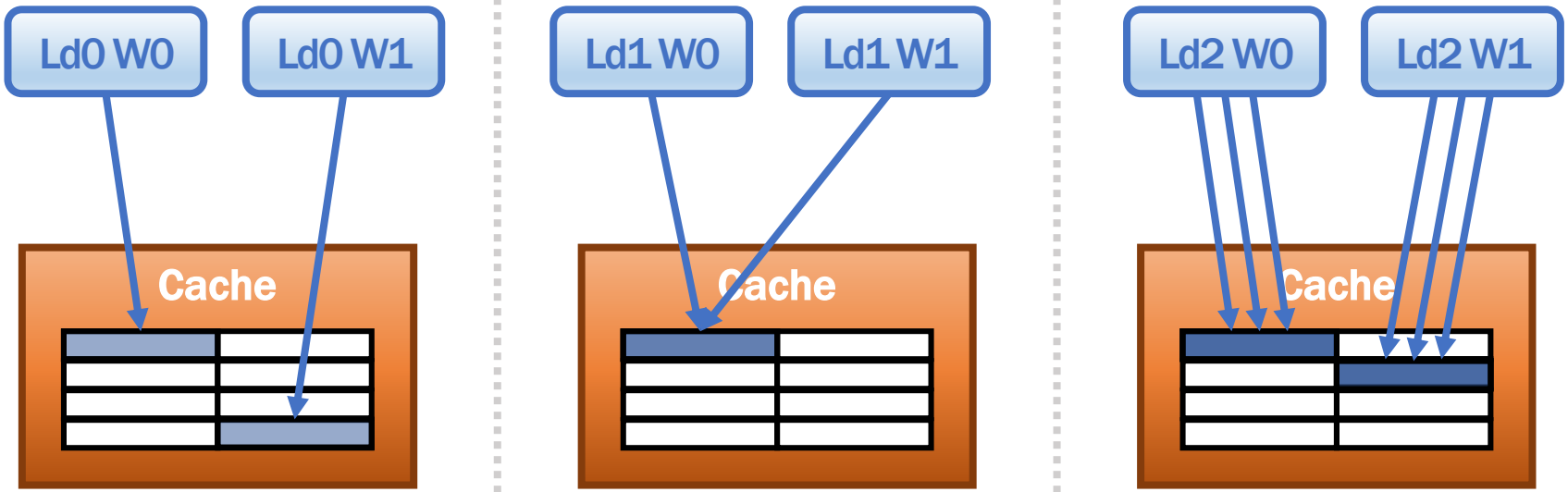
- Blocks or bypasses warps → releases cache contention

- Sacrifices TLP
- Bypassed warps
- Not effective for a



Observations

- Diverged data locality type (access pattern) per load



Streaming | **Inter-warp locality** | **Intra-warp locality**

Per-load cache management is beneficial

Access Pattern-Aware Cache Management

- Access Pattern-Aware Cache Management (APCM)
  - Per-load cache management method for GPU
  - Detects the access pattern of each global load
  - Applies the specific cache management scheme per load

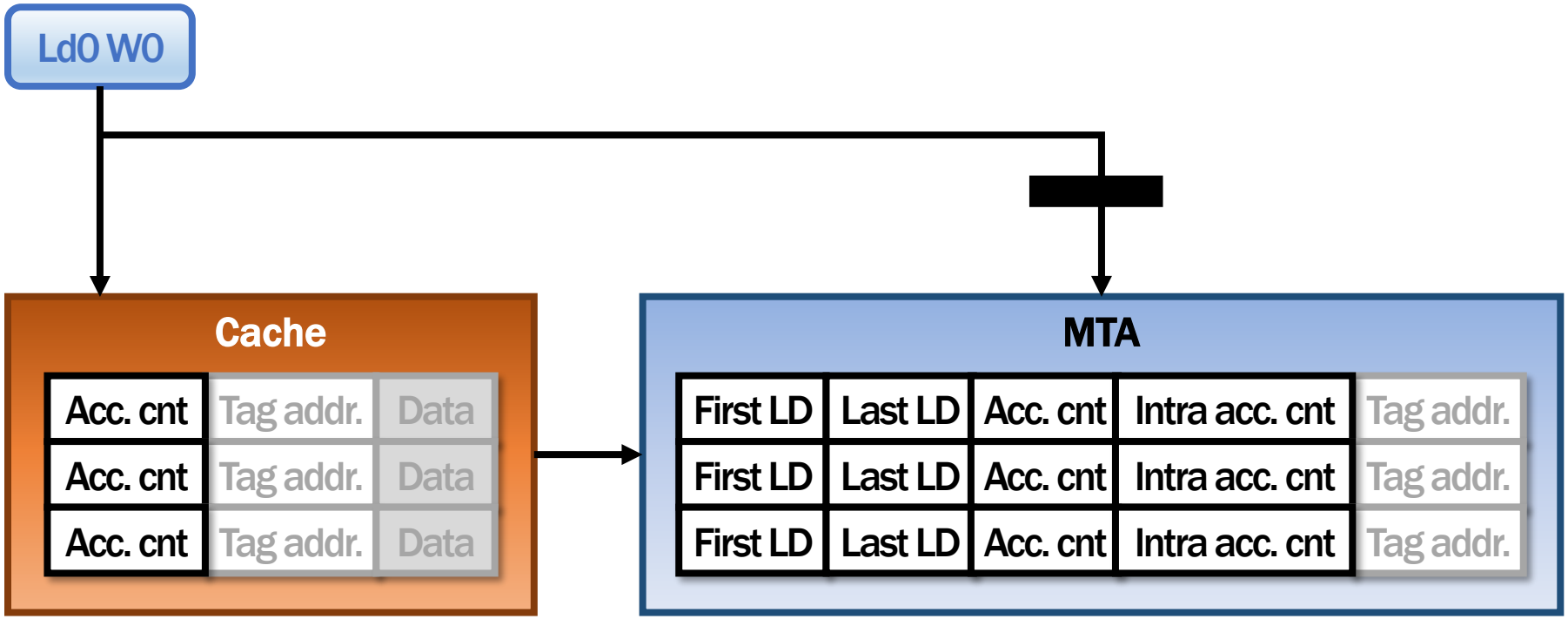
1. How to detect the locality types

2. How to apply cache management schemes

Categorizing LD Categories

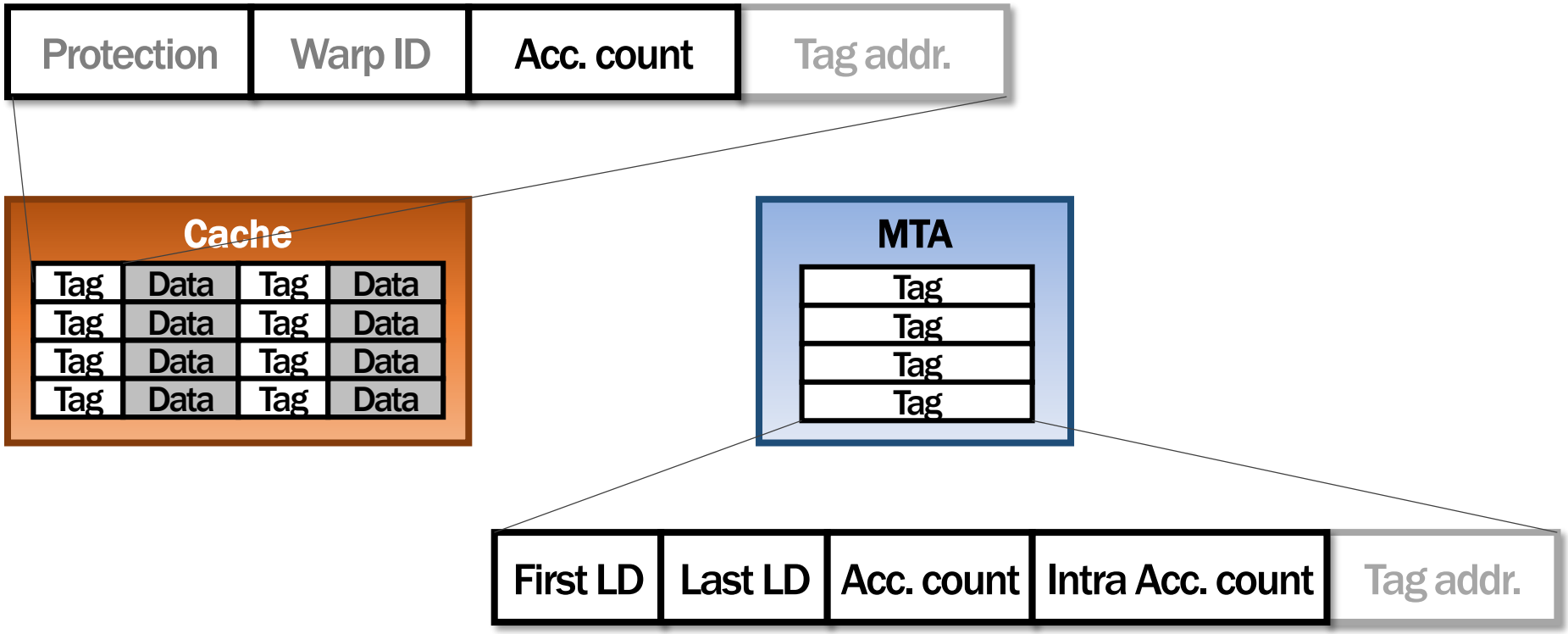
- Detecting access patterns
  - Monitor Tag Array (MTA)
  - Tracks access history of one monitored warp

  → *Strong access pattern similarity across all warps*



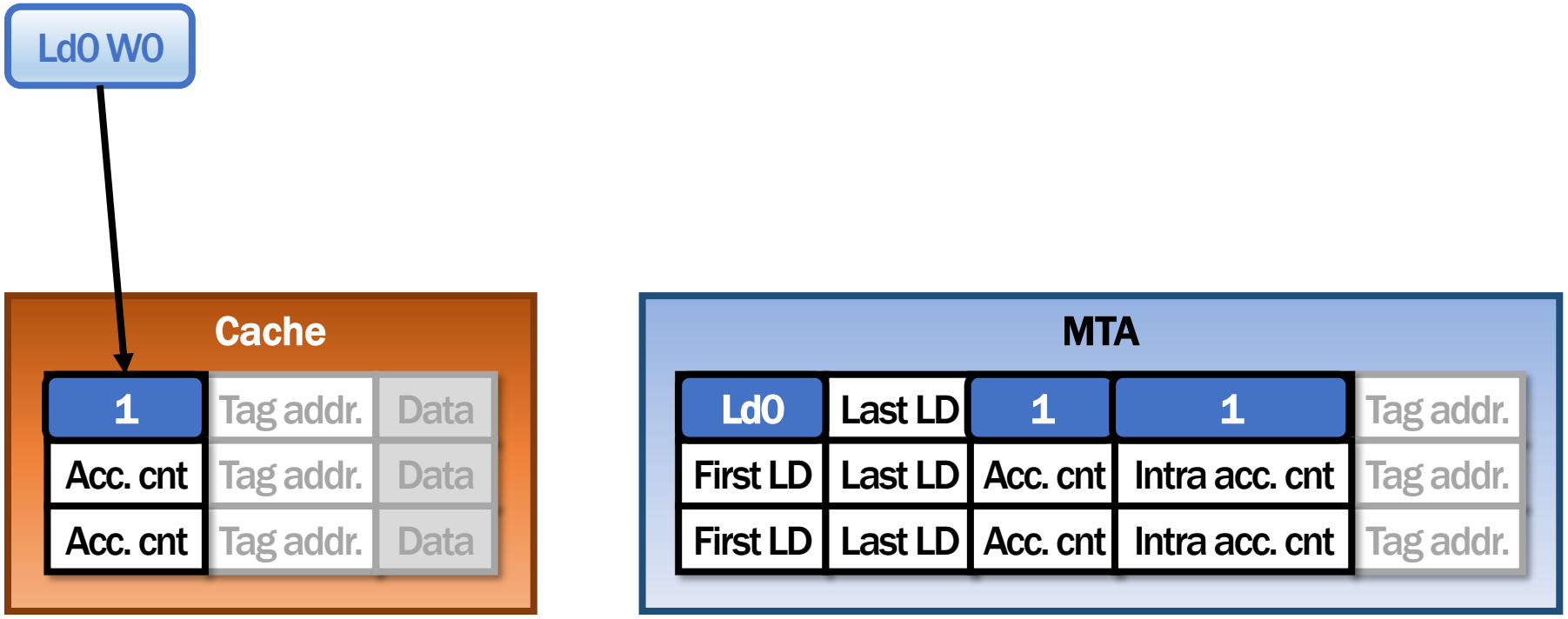
LD Categorization

- LD categorization is based on access count of MTA
  - Warp IDs and load IDs are required



Example

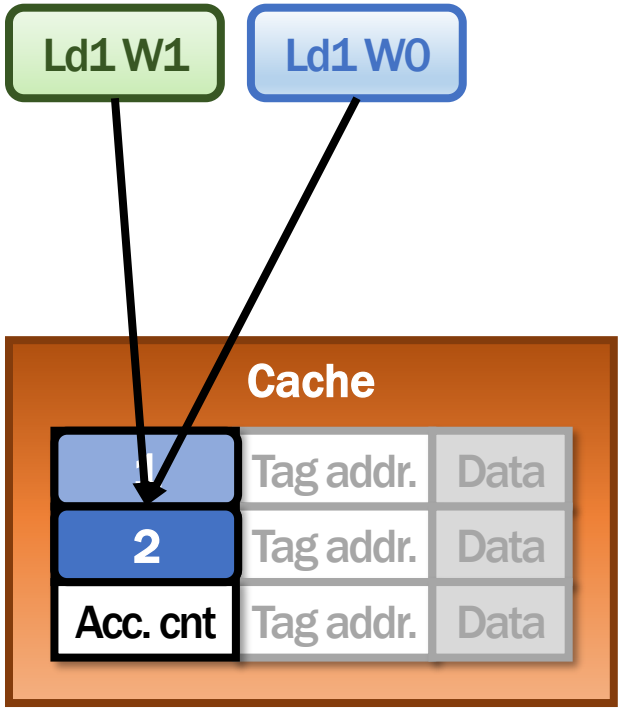
- Detecting access patterns

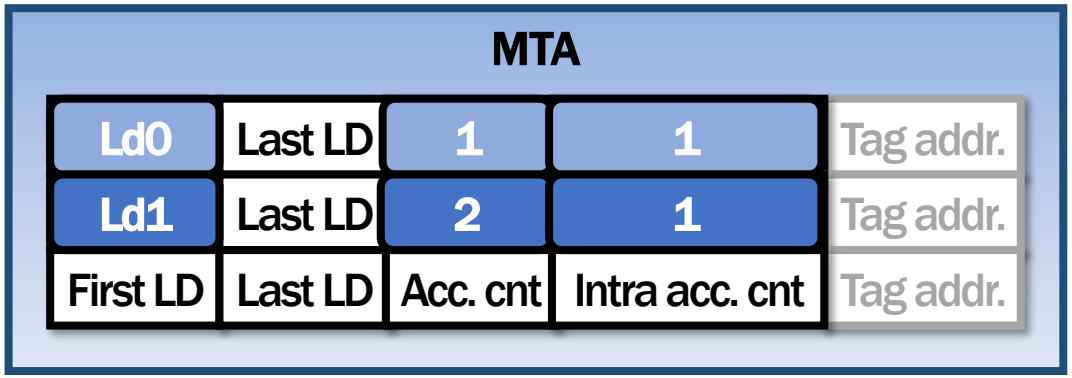


* Monitored warp = W0

Example

- Detecting access patterns





* Monitored warp = W0

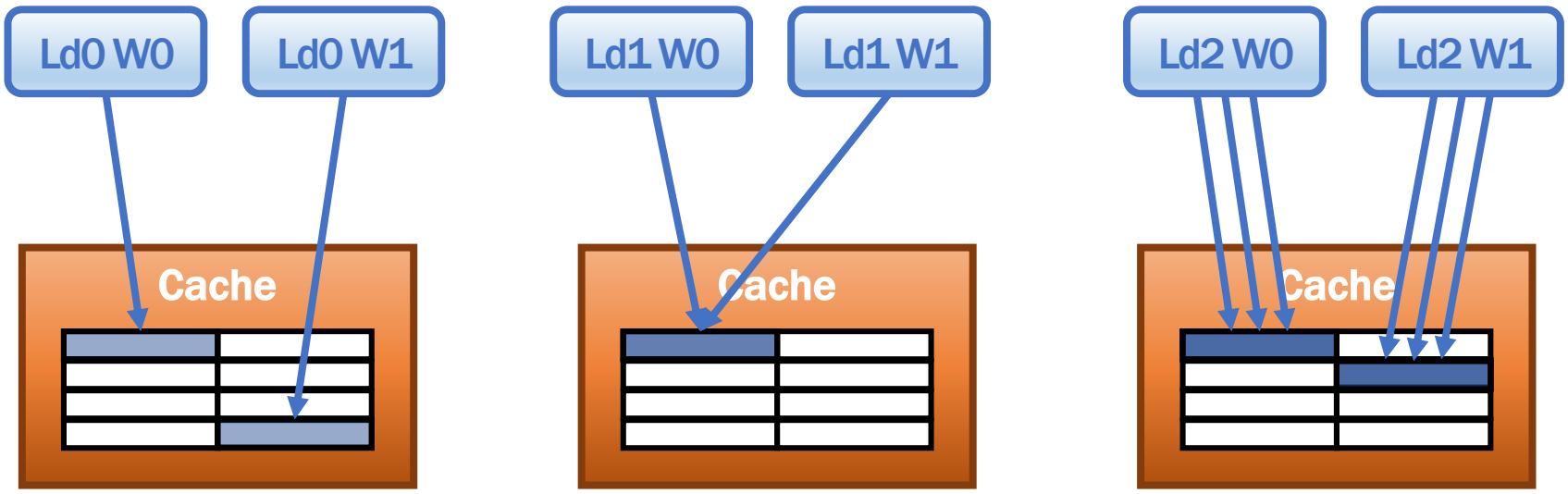
Example

- Detecting access patterns

* Monitored warp = W0

Using Access Count for Categorization

- Detecting access patterns



Streaming

- ✓ Acc. cnt = 1
- ✓ Intra acc. cnt = 1

Inter-warp locality

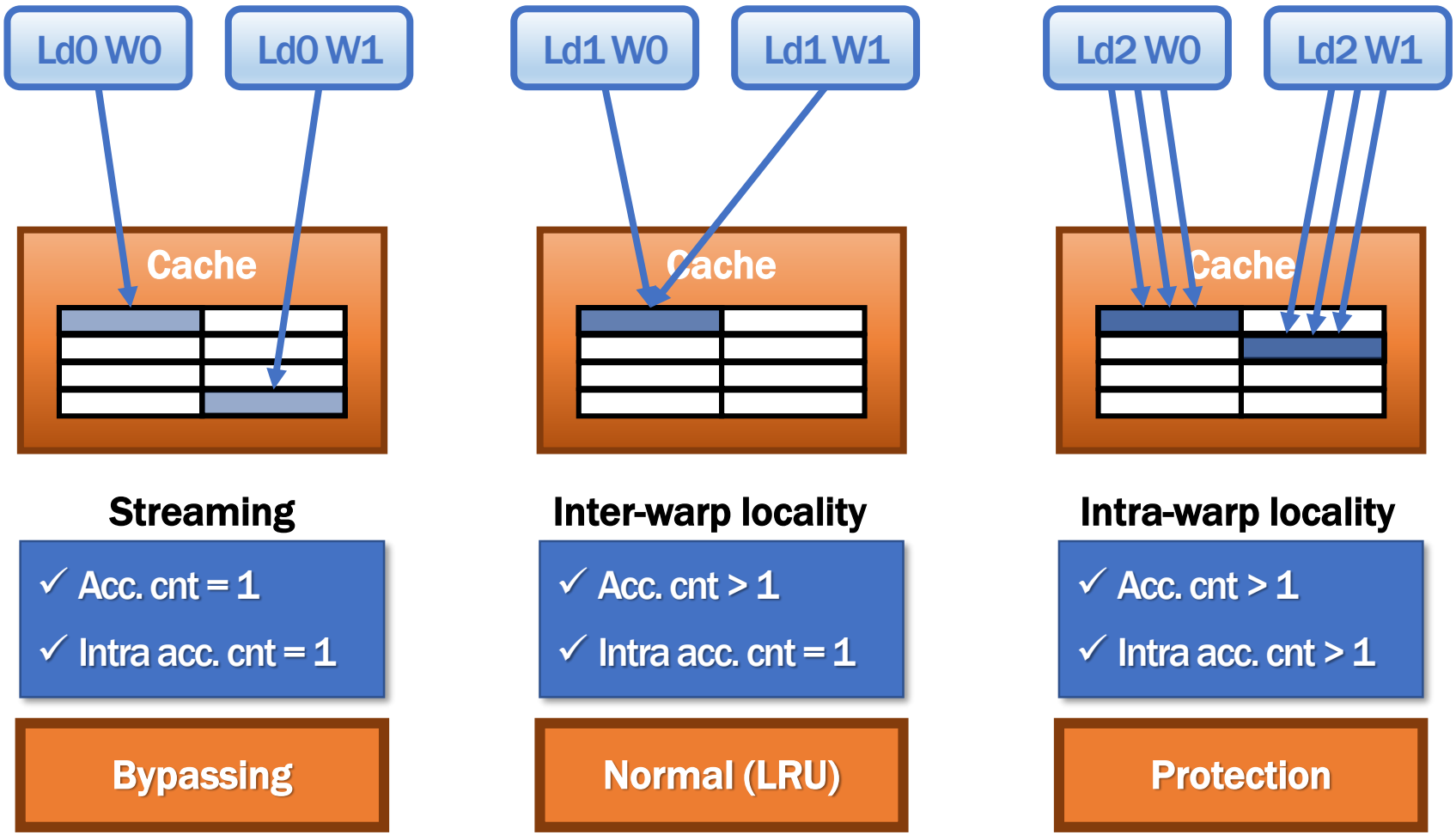
- ✓ Acc. cnt > 1
- ✓ Intra acc. cnt = 1

Intra-warp locality

- ✓ Acc. cnt > 1
- ✓ Intra acc. cnt > 1

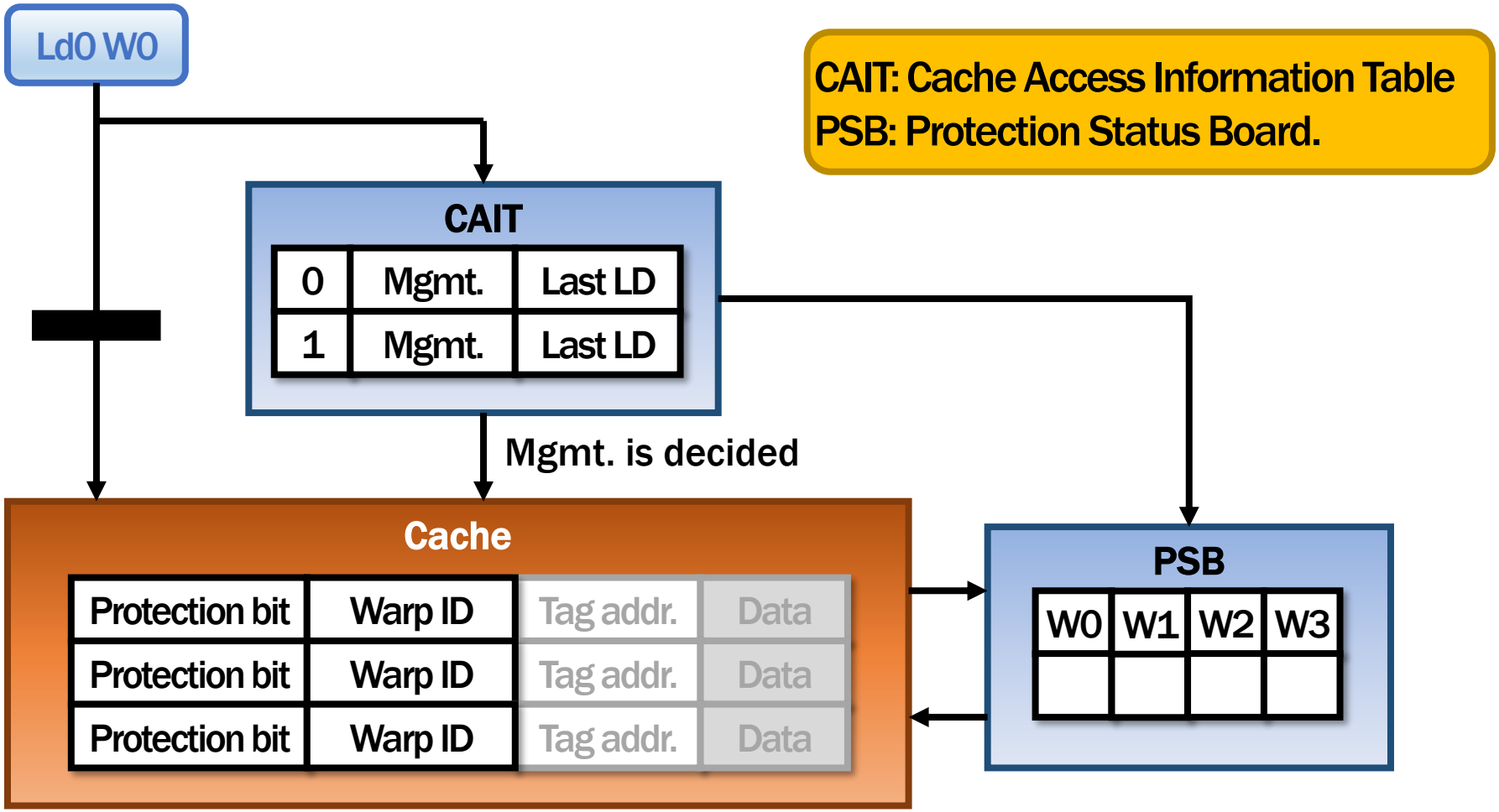
Per-LD Cache Management

- Applying per-load cache management



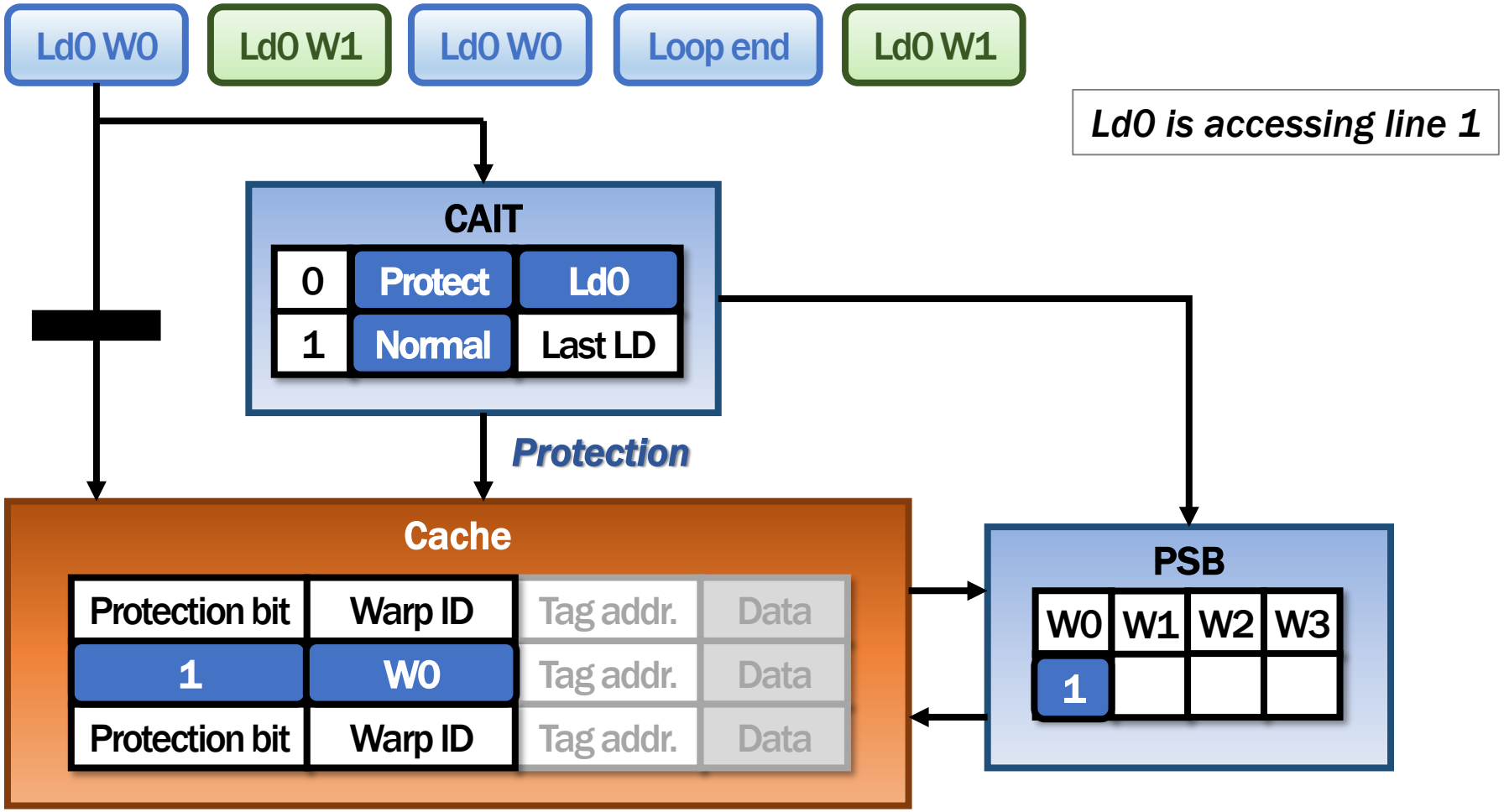
APCM Hardware

- Applying per-load cache management



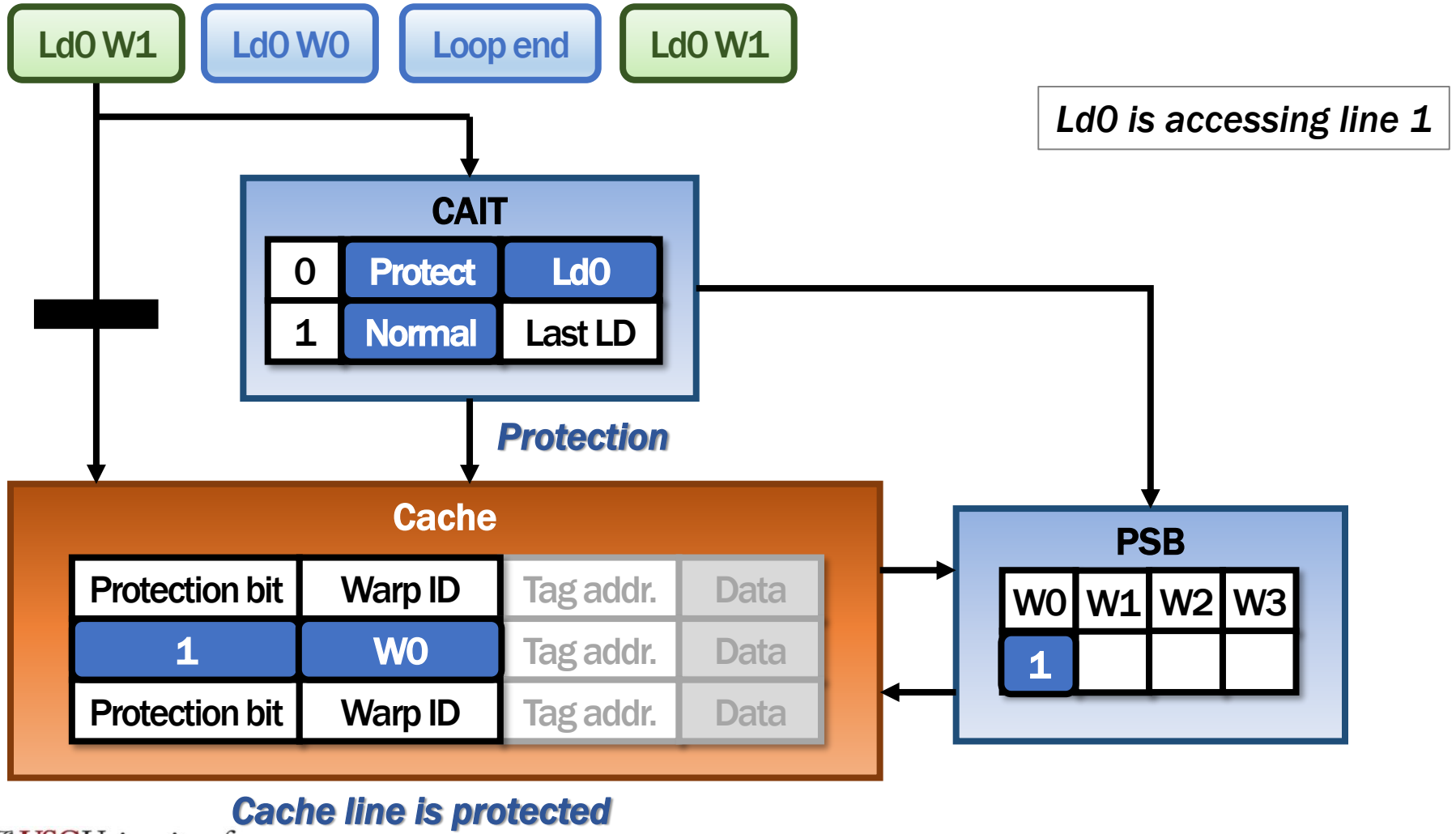
APCM by Example

- Cache line protection example



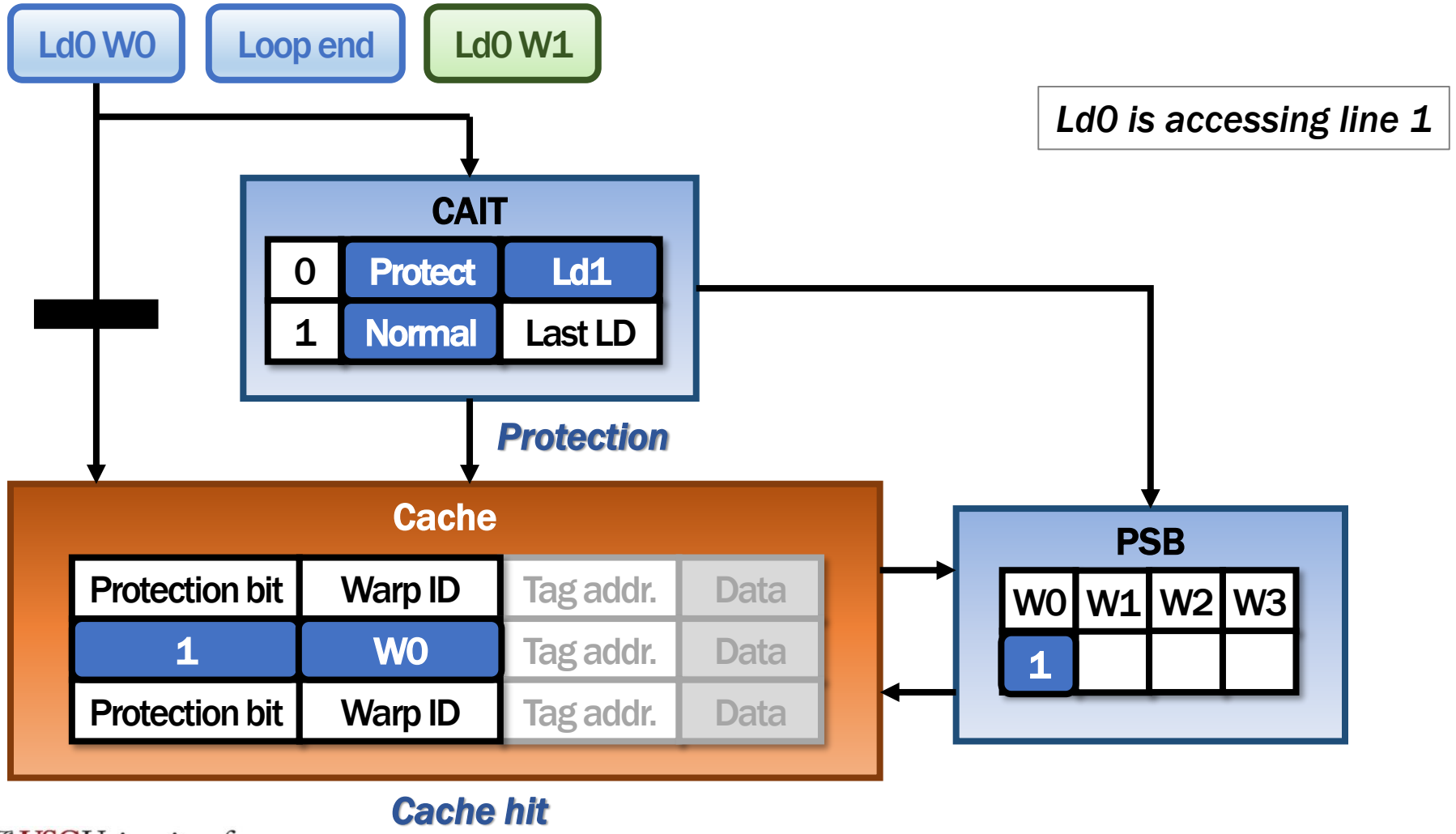
APCM by Example

- Cache line protection example



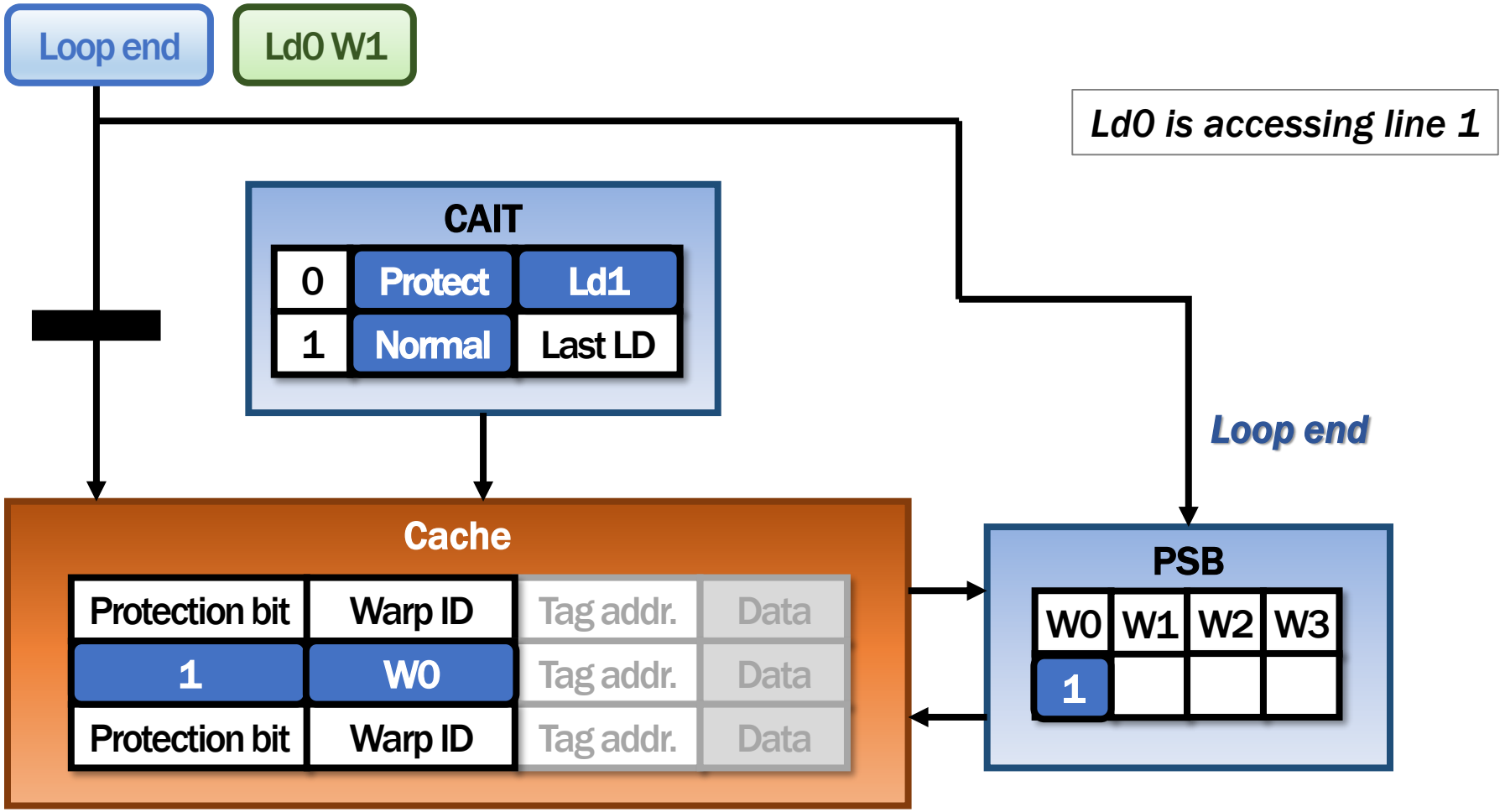
APCM by Example

- Cache line protection example



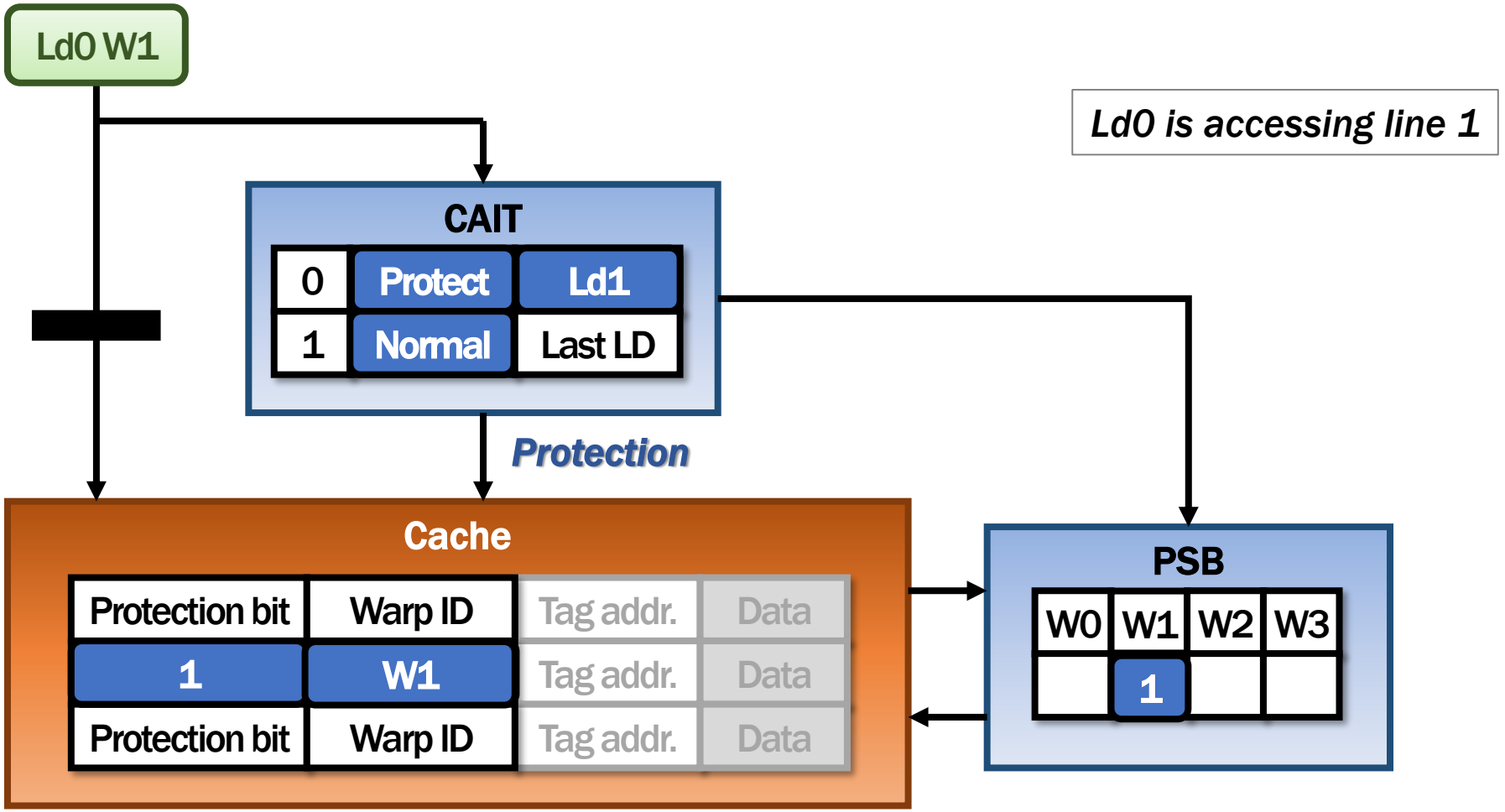
APCM by Example

- Cache line protection example



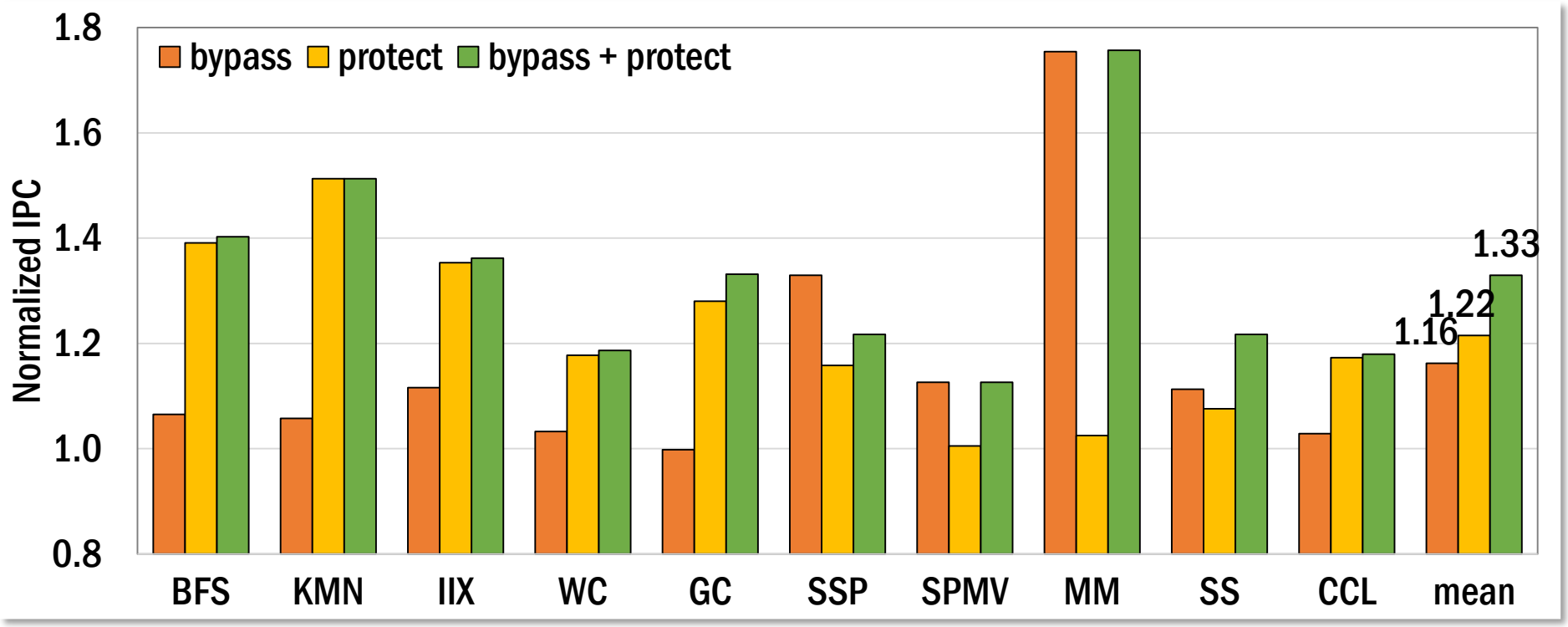
APCM by Example

- Cache line protection example



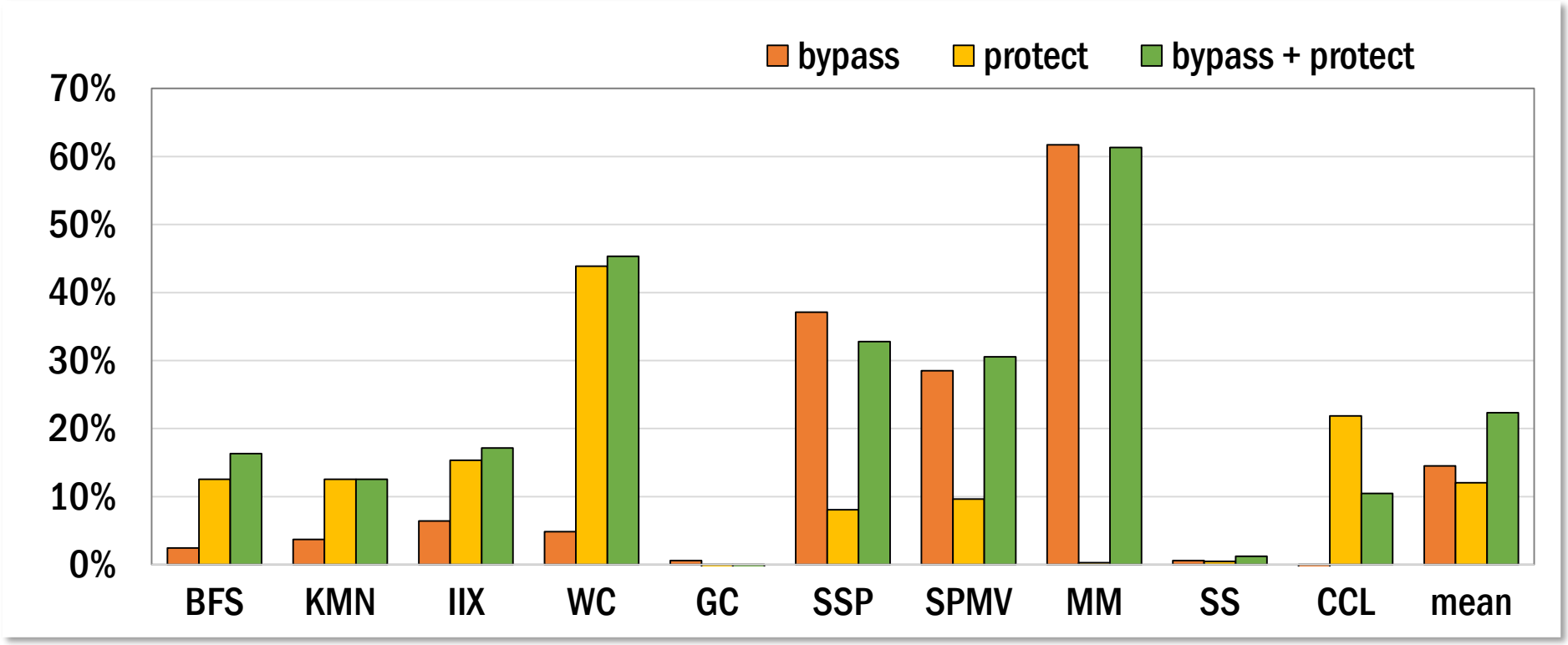
Results: Performance

- Performance improvement
  - CS + CM applications: 33% over baseline
  - All applications: 22%



Results: Miss Reduction

- Cache miss reduction
  - CS + CM applications: 22% over baseline



Results: More Design Explorations

- Performance improvement
  - With more recent architecture configuration: 42% over baseline
  - With GTO scheduling: 27% over GTO baseline
  - With warp-throttling: 49% over the best static warp throttling

See paper for more results

Conclusion

- Per-load cache management for GPU
- Effectively detects data locality type per load with small dedicated tag array
- Applies specific cache management scheme by the access pattern information
- Improves performance and data cache efficiency

Thank you

Access Pattern-Aware Cache Management
for Improving Data Utilization in GPU

Gunjae Koo, Yunho Oh, Won Woo Ro, and Murali Annavaram

(gunjae.koo@usc.edu)