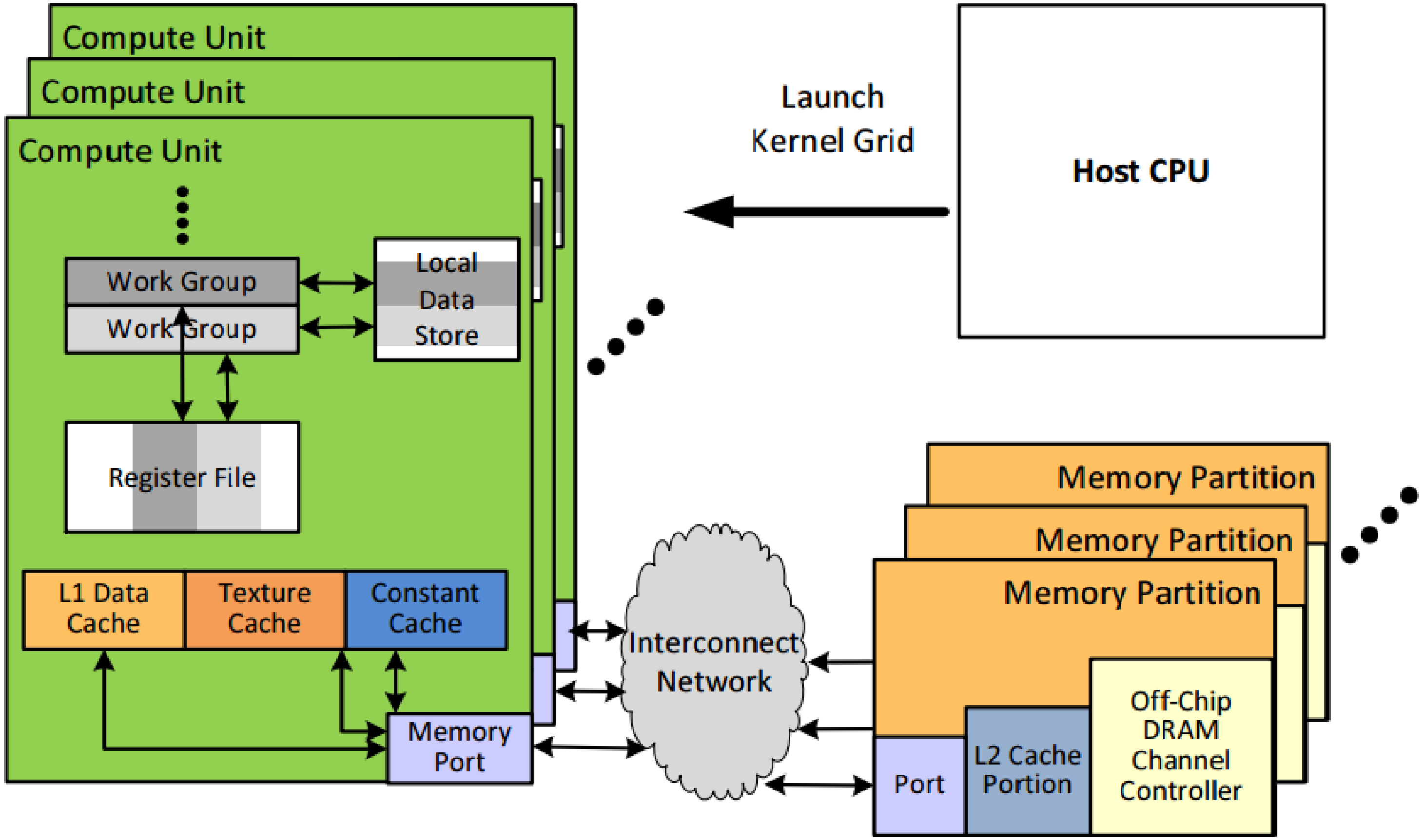
# Cache-Conscious Wavefront Scheduling

Timothy G. Rogers Mike O'Connor Tor M. Aamodt

# Outline

- Background
- Motivation
- Cache-Conscious Wavefront Scheduling (CCWS)
- Evaluation
- Conclusion

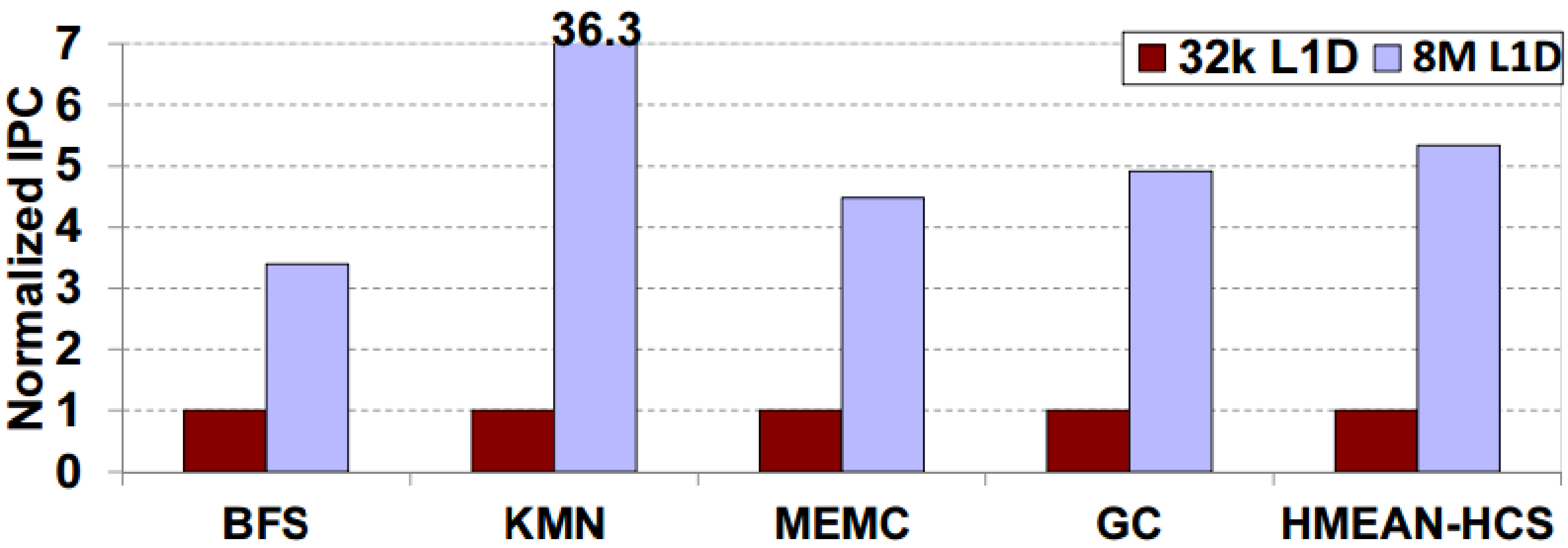
# Overview of the architecture




- Compute unit<->streaming multiprocessor(SM)
- Work group<->block
- Wavefront<->warp

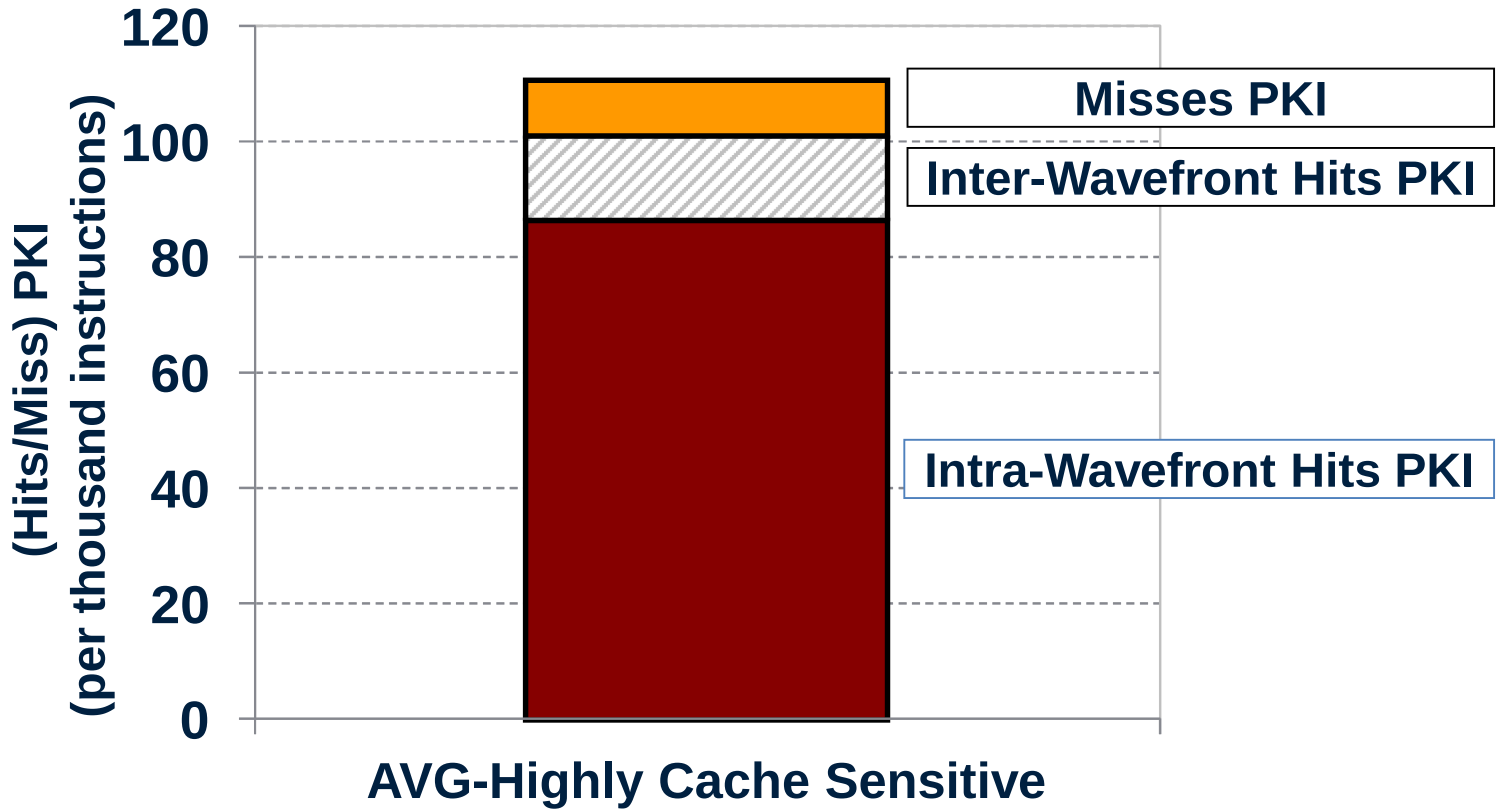
# Highly cache-sensitive applications

- Breadth First Search (BFS)
- K-Means (KMN)
- Memcached-GPU (MEMC)
- Parallel Garbage Collection (GC)




- Increase 32k L1D to 8M
  - Minimum **3x** speedup
  - Mean speedup **>5x**

# Quantifying intra-/inter-wavefront locality




- intra-wavefront locality: data is initially referenced and re-referenced from the same wavefront
- inter-wavefront locality: data that is initially referenced by one wavefront and re-referenced by another

# Motivation

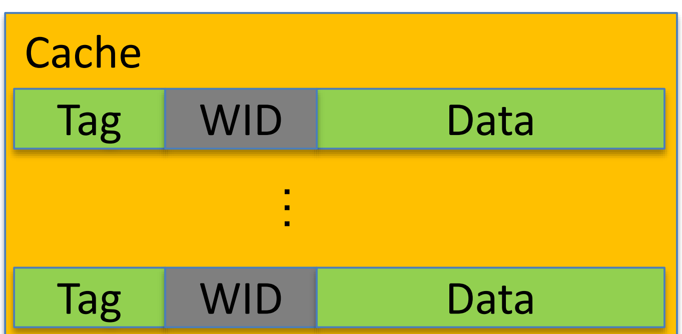
- Improve performance of cache-sensitive workloads by decreasing L1 data cache misses
- Inspiration: cache replacement and insertion policies that attempt to predict when cache lines will be reused
- Proposed wavefront scheduler: changes the re-reference interval to reduce the number of interfering references between repeated accesses to high locality data

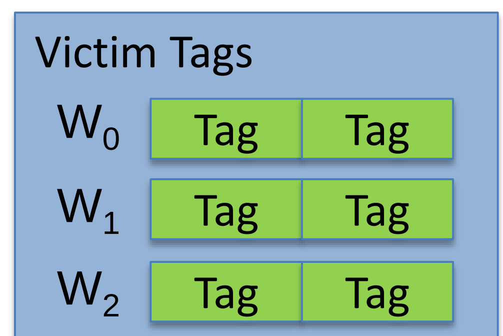
# Cache-Conscious Wavefront Scheduling (CCWS)

- Dynamically determine the number of wavefronts allowed to access the memory system and which wavefronts those should be.
- Major components:
  - Lost Locality Detector
  - Locality Scoring System

# Lost Locality Detector

- Uses a victim tag array (VTA)
- Each wavefront has its own small VTA
- VTA only stores cache tags and does not store line data
- When that line is evicted from the cache, its tag information is written to that wavefront's portion of the VTA.
- Whenever there is a miss in the L1D cache, the VTA is probed. If the tag is found in that wavefront's portion of the VTA, the LLD sends a VTA hit signal to <u>the locality scoring system</u>

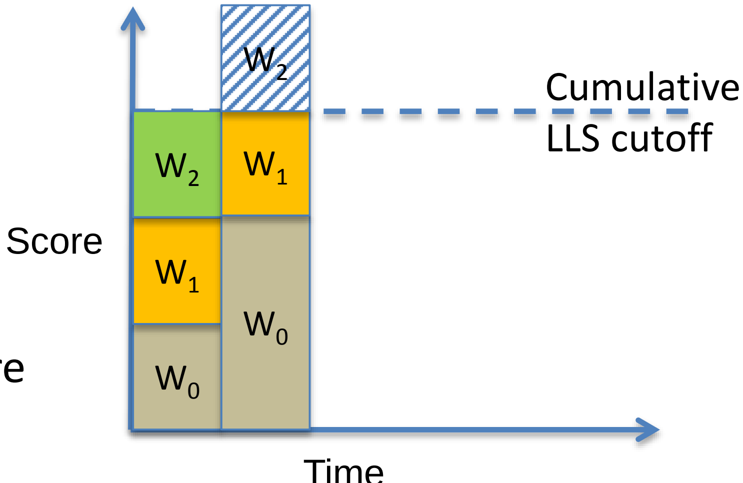



- WID: wavefront ID

# Locality Scoring System

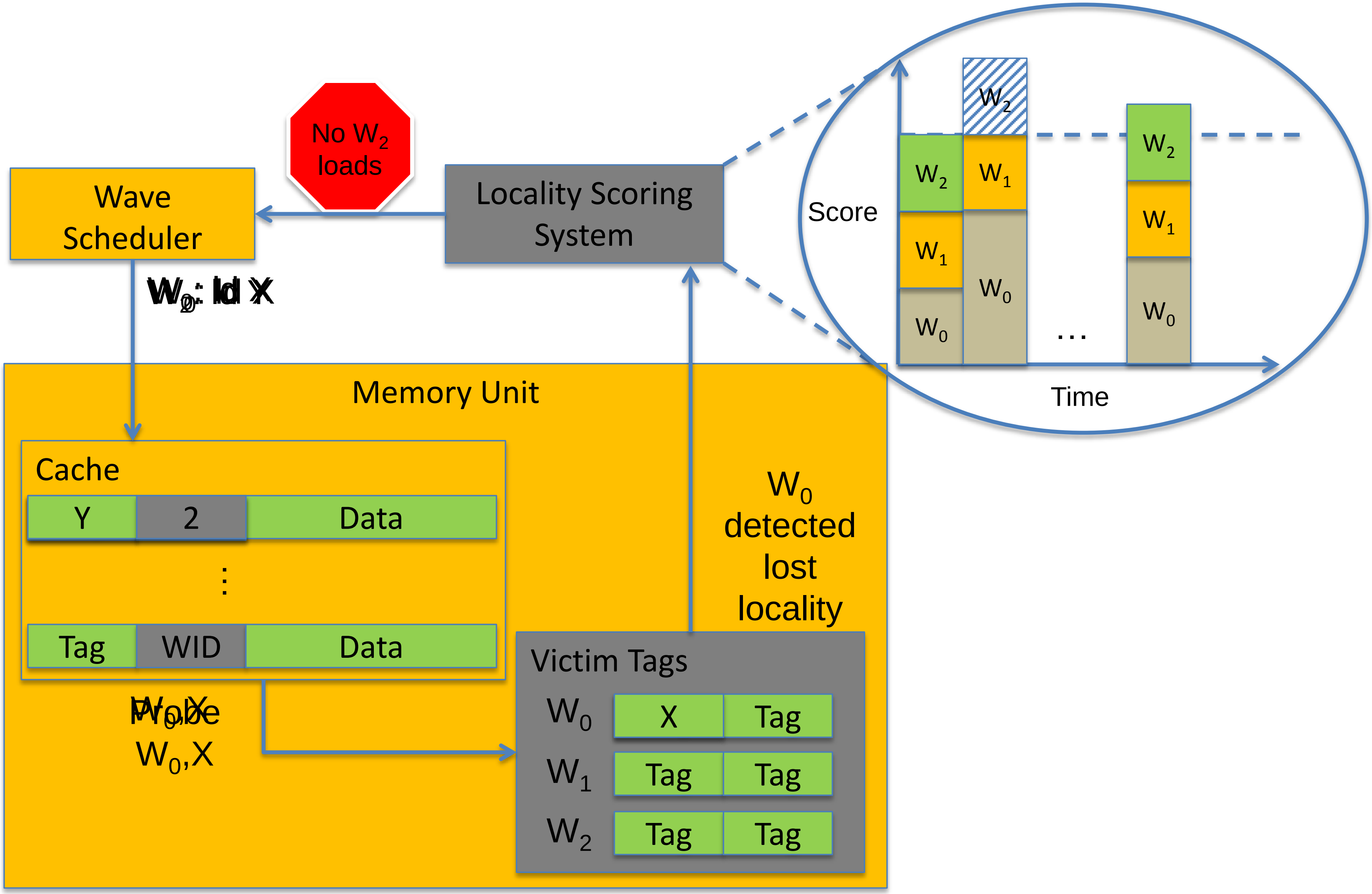
- Assign each wavefront a constant base lost-locality score (LLS) when assigned to a core
- A LLD sends a VTA hit signal for one wavefront -> wavefront's LLS ↑
- The scores each decrease by one point every cycle until they reach the base locality score.
- VTA hit signals inform the scoring system that a wavefront has missed on a cache line that may have been a hit if that wavefront had more exclusive access to the L1D cache.
- The locality scoring system gives wavefronts losing the most intra-wavefront locality more exclusive L1D cache access by preventing the wavefronts with the smallest LLS from issuing load instructions.




LLS cutoff=NumActiveWaves × BaseLocalityScore

CAMELab.org
Computer Architecture and Memory Systems Lab
UT DALLAS

# CCWS Implementation

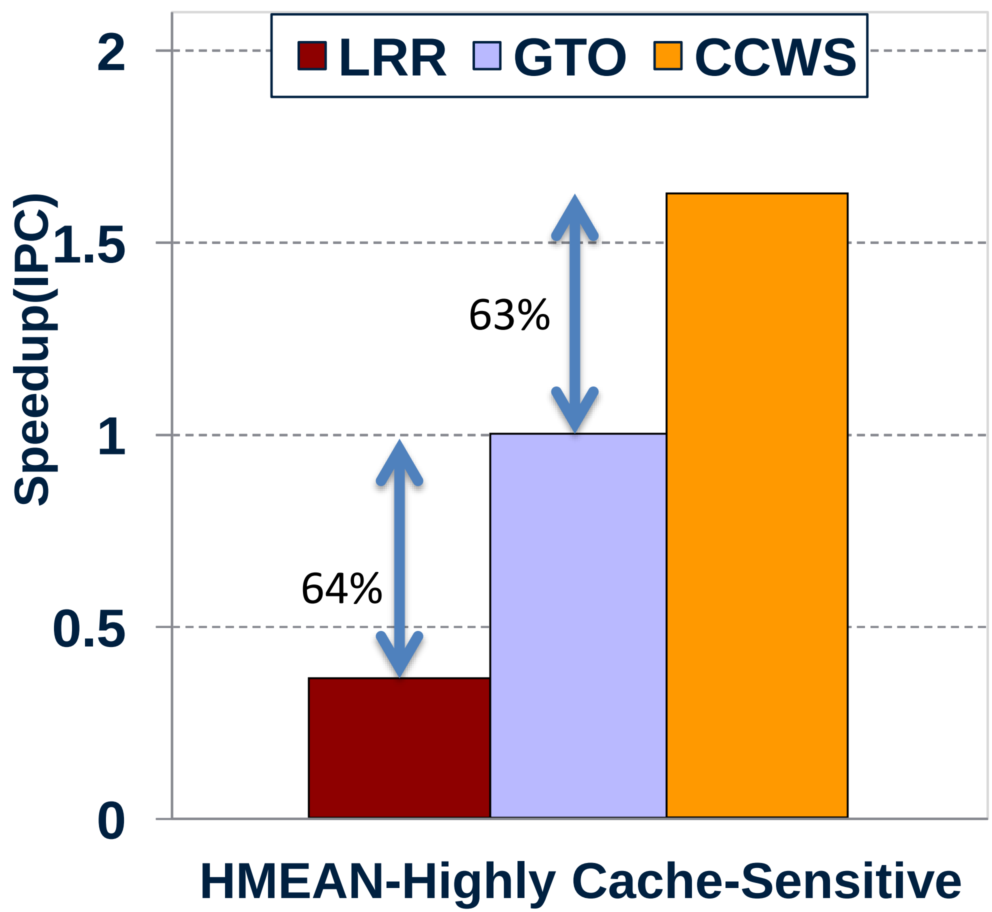


# Methodology

- **GPGPU-Sim (version 3.1.0)**
- 30 Compute Units (1.3 GHz)
  - 32 wavefront contexts (1024 threads total)
- 32k L1D cache per compute unit
  - 8-way
  - 128B lines
  - LRU replacement
- 1M L2 unified cache

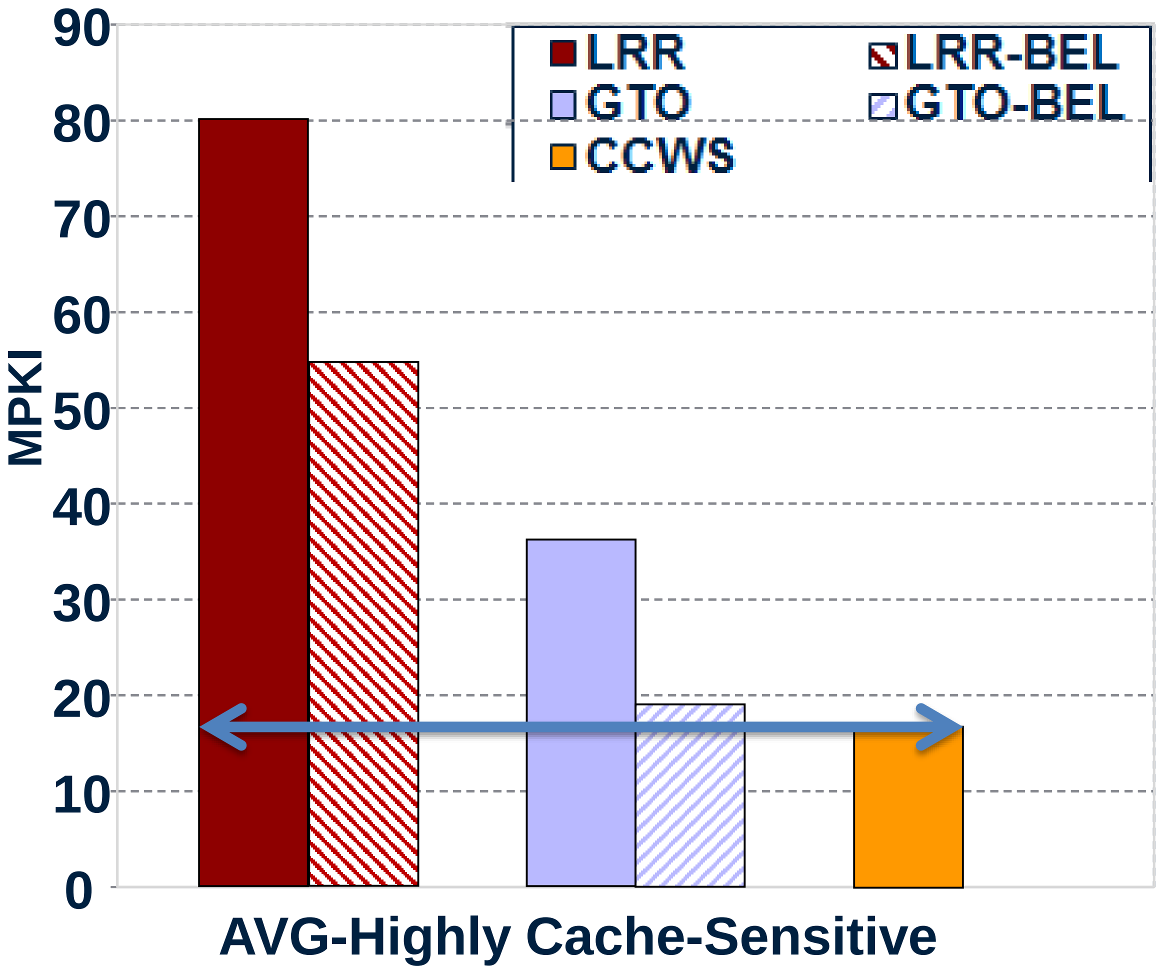
- **Stand Alone GPGPU-Sim Cache Simulator**
- Trace-based cache simulator
- Fed GPGPU-Sim traces
- Used for oracle replacement(Belady-optimal replacement policy)

# Performance Results




- LRR: loose round-robin scheduler. If a wavefront cannot issue during its turn, the next wavefront in round-robin order is given the chance to issue.
- GTO: greedy-then-oldest scheduler. GTO runs a single wavefront until it stalls then picks the oldest ready wavefront. The age of a wavefront is determined by the time it is assigned to the core.

# Cache Miss Rate



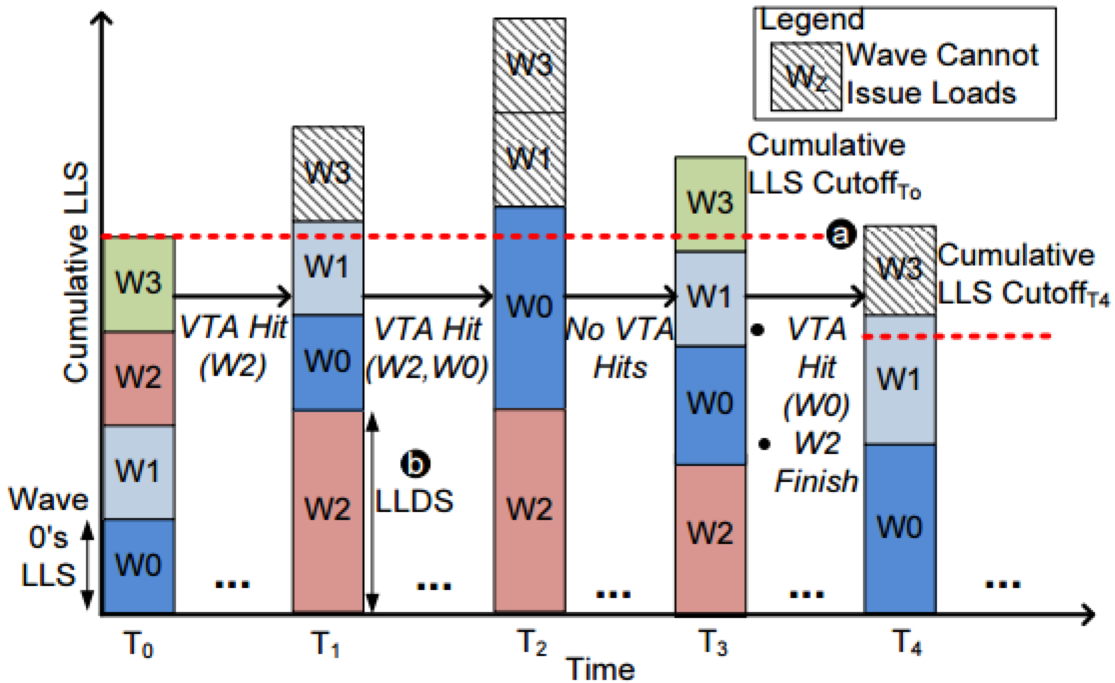

•Belady Optimal:
The *most* efficient caching algorithm – always discard the information that will not be needed for the longest time in the future. (generally impossible, is used to compare the effectiveness of the actually chosen cache algorithm)

•CCWS: less cache misses than other schedulers optimally replaced

# Conclusion

- At an estimated cost of 0.17% total chip area, CCWS reduces the number of threads actively issued on a core when appropriate. (mainly comes from the victim cache array)
- This leads to a harmonic mean 63% improvement in throughput on highly cache-sensitive benchmarks, without impacting the performance of cache-insensitive workloads.

# Locality scoring system operation example




- T1->T2: capping each wavefront's score at LLDS, regardless of how many VTA hits it has received
- T3->T4: when a wavefront is added or removed from the system, the cumulative LLS cutoff changes
- LLDS=lost-locality detected score

# Lost-Locality Detected Score (LLDS)

$$LLDS = \frac{VTAHits_{Total}}{InstIssued_{Total}} \cdot K_{THROTTLE} \cdot CumLLSCutoff$$

- $VTAHits_{Total}$=the total number of VTA hits across all this compute unit's wavefronts
- $InstIssued_{Total}$=all the instructions this compute unit has issued
- The fraction: an indication of how much locality is being lost on this core per instruction issued
- $K_{THROTTLE}$: tune how much throttling is applied when locality is lost. A larger constant favors less multithreading by pushing wavefronts above the cutoff value more quickly and for a longer period of time. (K=8)