PACT-2014 The 23rd International Conference on Parallel Architectures and Compilation Techniques

CAWS: Criticality-Aware Warp Scheduler for GPGPU Workloads

Shin-Ying Lee and Carole-Jean Wu

School of Computing, Informatics, and Decision Systems Engineering

Arizona State University

Ira A. Fulton Schools of Engineering

ARIZONA STATE UNIVERSITY

Center for Embedded Systems

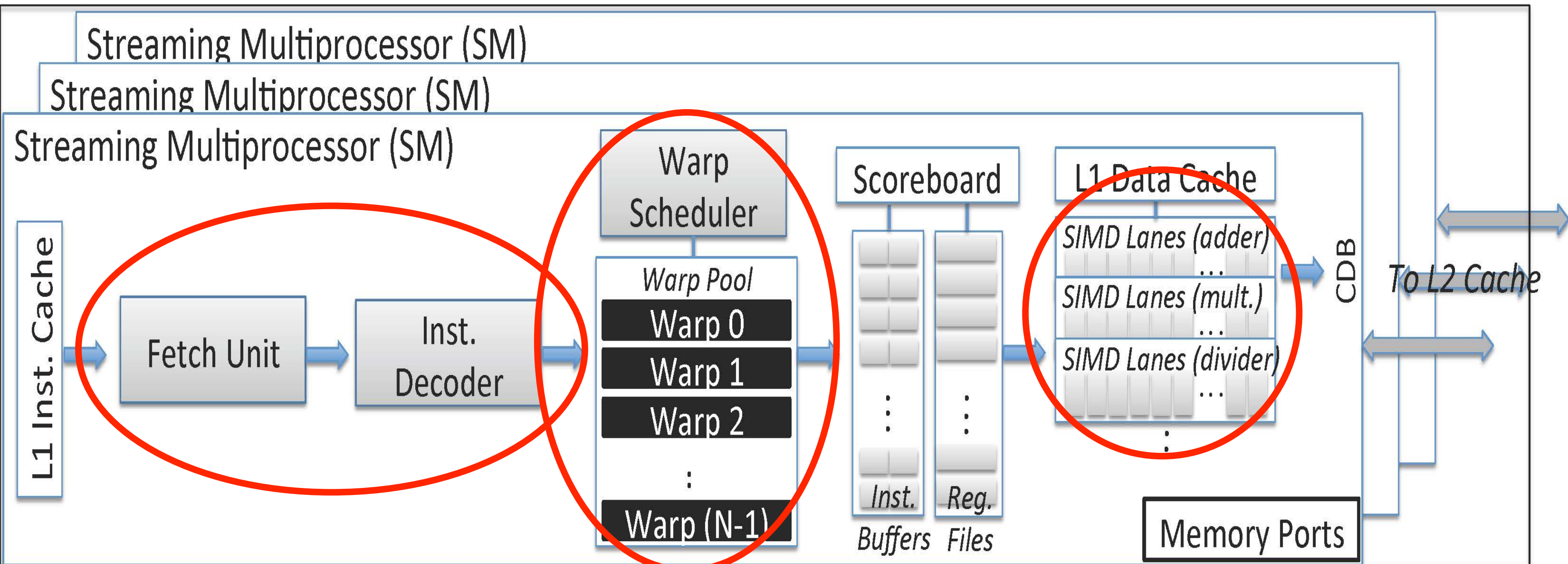
An NSF Industry/University Cooperative Research Center

Graphics Processing Units (GPUs)

- In addition to render video frames, modern GPUs are also deployed to process parallel workloads.
- The benefit of using GPUs to perform general computation is to hide operation latency
  - *Massive multi-threading*
  - *Fast context-switching*

Unified GPU Architecture (Computation Path)

- The computation and graphics unified GPU architecture to support general computation



Research Questions

- Is the current GPU design good enough to hide execution latency?
- What kind of execution latency is hidden under the GPU scheduler?
- How do we propose a better scheduling policy to improve GPU's latency hiding ability?

Outline

- GPU Latency Characterization
- Warp/Wavefront Criticality
- Criticality-Aware Warp Scheduling (CAWS)
- Discussion and Conclusion

Root-cause of Warp Stall

- Pipeline hazards
  - Data hazard
  - Structural hazard
  - Control hazard
- Memory access latency
- Instruction fetch policy
- Synchronization barrier
- Warp scheduling policy

Latency Characterization Algorithm

```
probe(w)
        w.Scheduling+=CurTime-w.PrevTime
        w.PrevTime=CurTime
        if(Sync)                                w.Sync++
        else if(EmptyInstBuffer)                w.Fetch++
        else if(BranchTaken)                    w.CtrlHazard++
        else if(DataDependency)
                if(OldDataCacheMiss)            w.DataCacheMiss++
                else                            w.DataHazard++
        else if(FU_unavailable)                 w.StructuralHazard
        else                                    w.Exec++
```

Warp/Wavefront Scheduler

1. Select a warp according to the scheduling policy at every cycle
2. Probe and record the selected warp's current status
3. Iteratively till find a ready warp

while(NotVisitedAllWarps)

w = FindNextWarp()

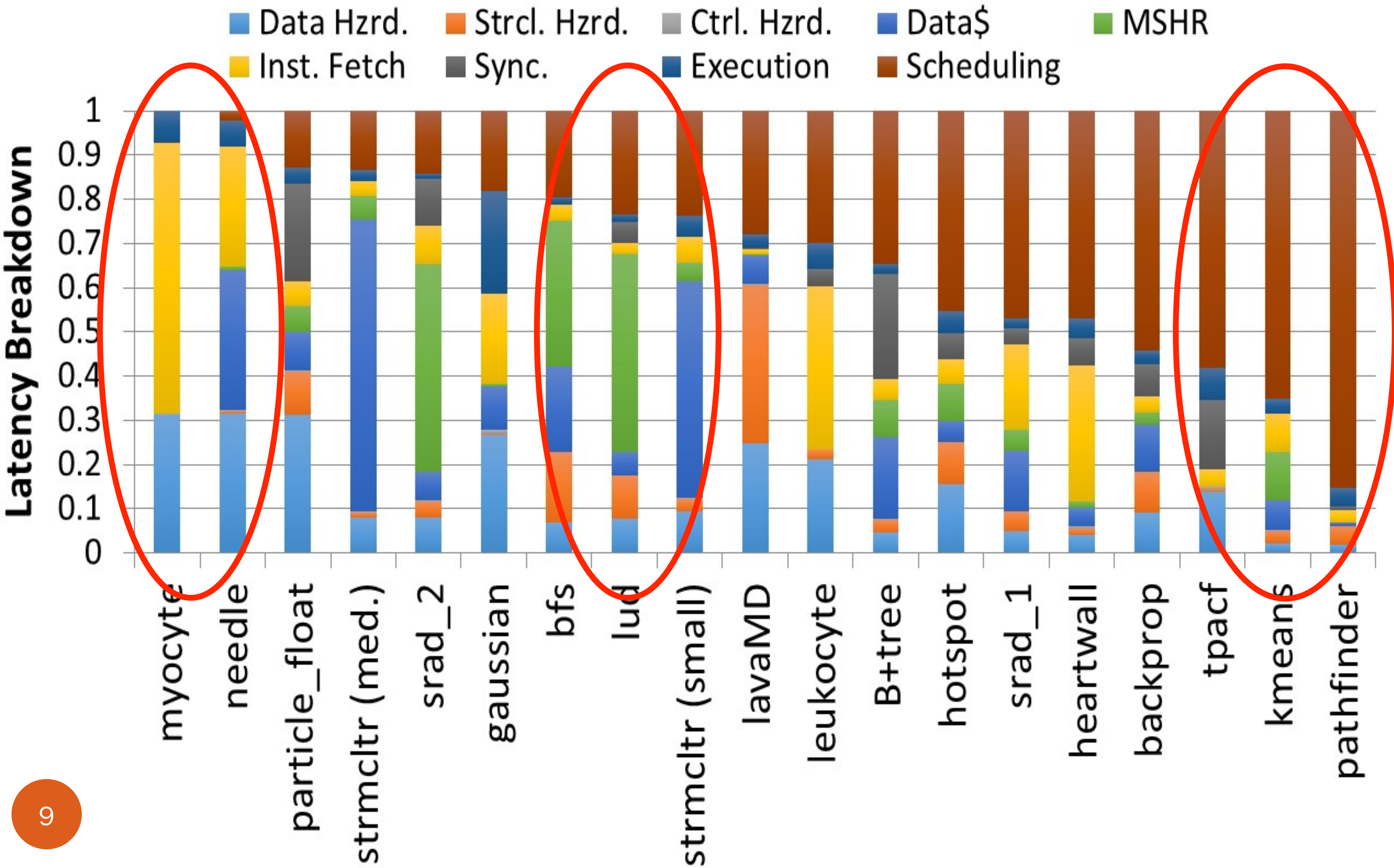
probe(w)

if(w is a ready warp)

issue(w)

break

Latency Characterization Results

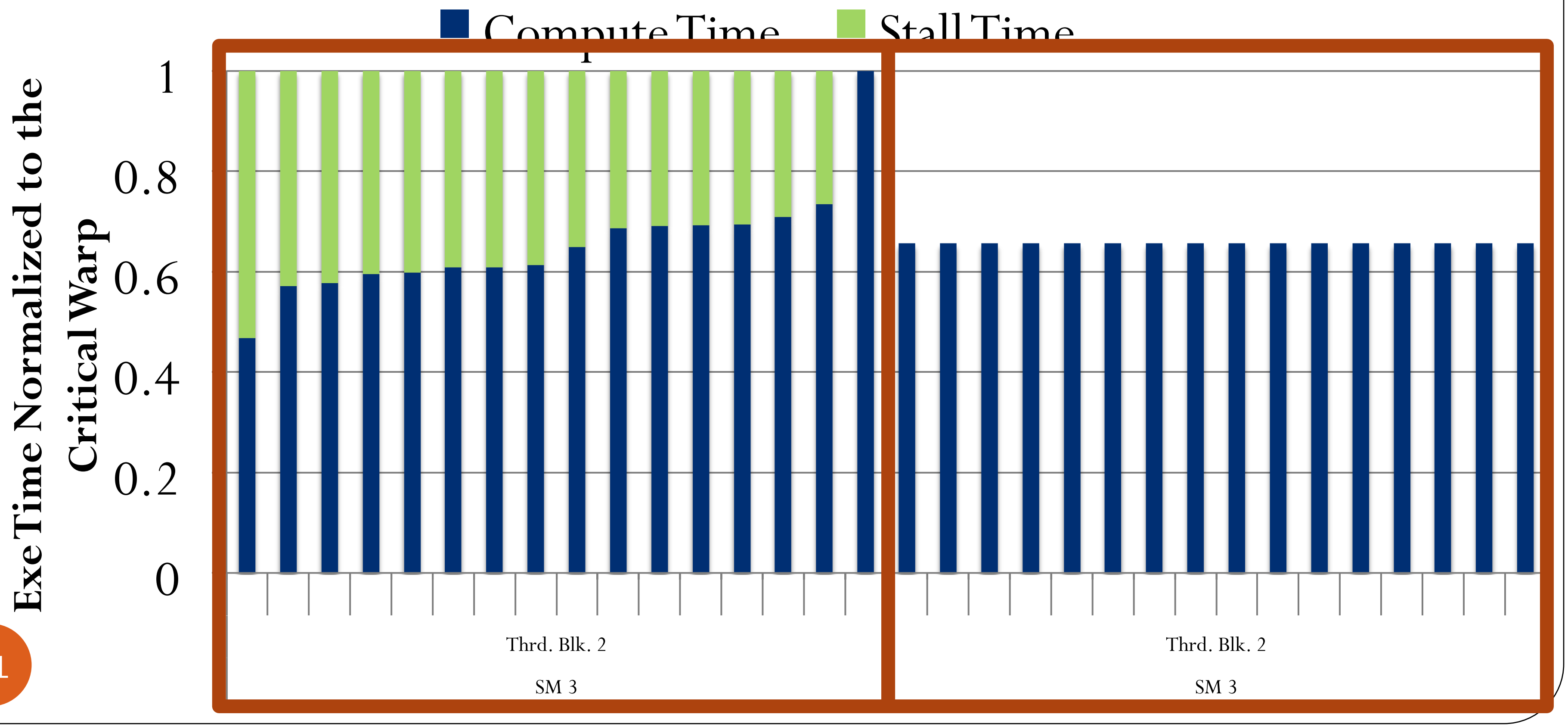


Outline

- GPU Latency Characterization
- Warp/Wavefront Criticality
- Criticality-Aware Warp Scheduling (CAWS)
- Discussion and Conclusion

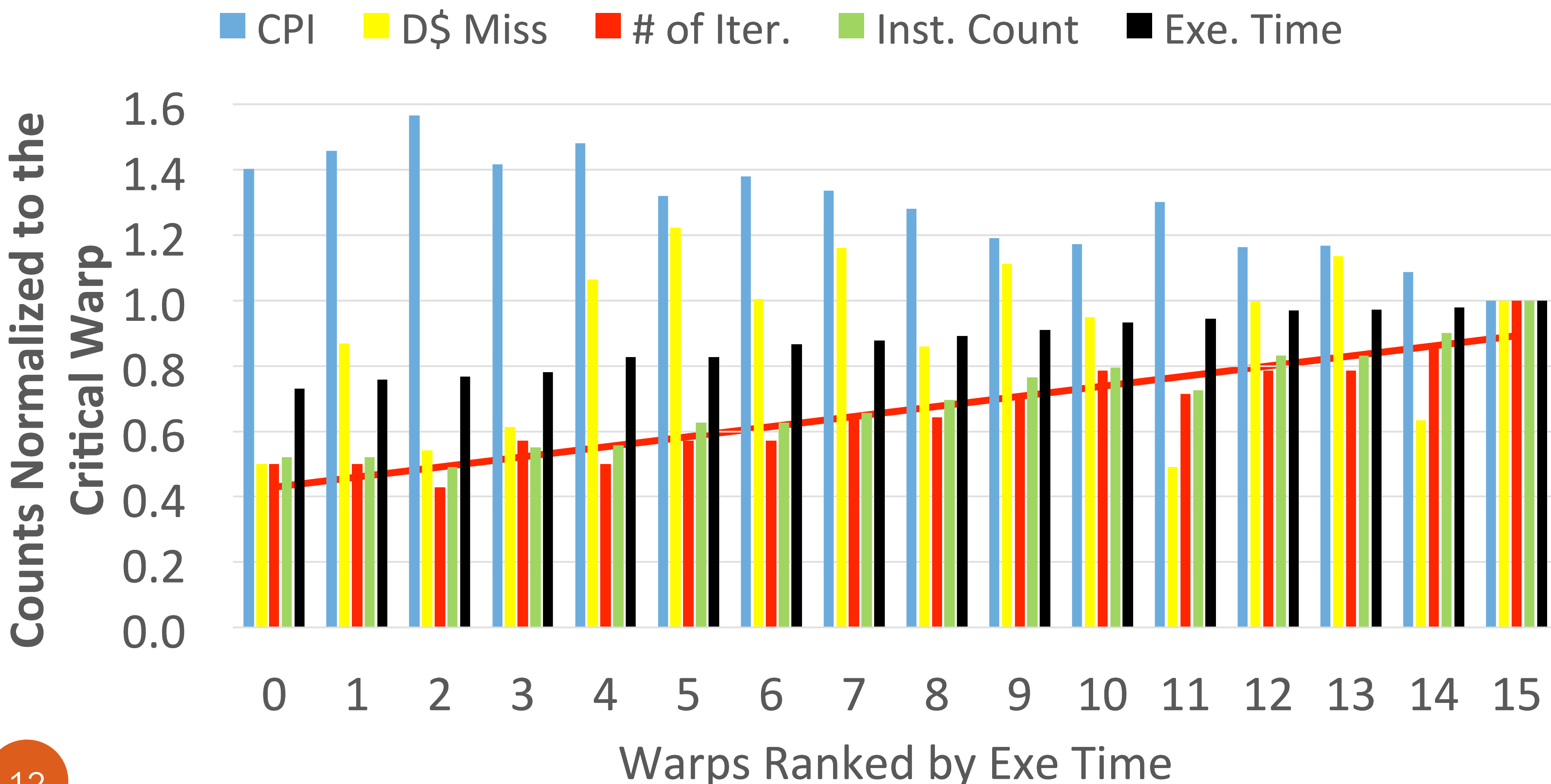
Warp/Wavefront Criticality

- Slow warps/wavefronts take much longer time to finish their designated jobs.
- Fast warps are blocked at an implicit/explicit synchronization barrier to wait for the slow warps/wavefronts



Factor Causing Warp Criticality for *bfs*

workload imbalance caused by branches



Outline

- GPU Latency Characterization
- Warp/Wavefront Criticality
- Criticality-Aware Warp Scheduling (CAWS)
- Discussion and Conclusion

Criticality-Aware Warp Scheduling

- To equalize the execution time disparity, the Criticality-Aware Warp Scheduling (CAWS) algorithm prioritizes and schedule warps by warps' weight.
- Assign warps different weight based on their criticality.
- Slower warps receive more time slots to run advance.

Types of CAWS Policies

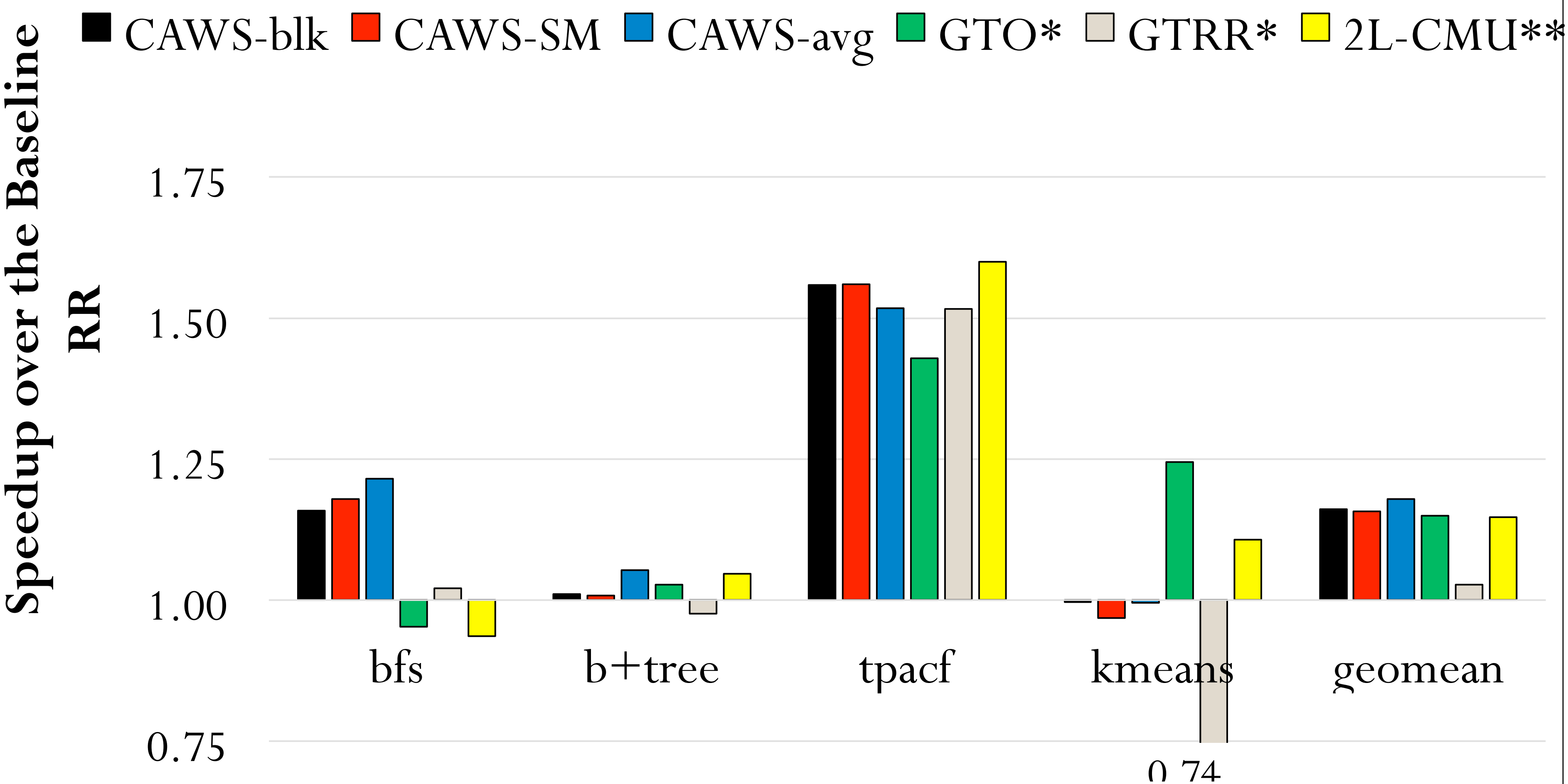
- CAWS-Blk
  - Prioritizing warps within a thread block
  - To equalize execution time disparity within a thread block
- CAWS-SM
  - Prioritizing warps within as SM
  - To equalize execution time disparity within an SM
- CAWS-Avg
  - Identifying the critical warps based on the average execution time across an SM

Experimental Environment

- GPU simulation infrastructure
  - GPGPU-sim v3.2.0*
  - nVIDIA nvcc toolchain v3.2
  - nVIDIA GTX480 architecture
- Benchmarks
  - Imbalance workloads: *bfs*
  - Small parallel regions: *b+tree*
  - I-cache intensive: *tpacf*
  - D-cache intensive: *kmeans*

*A. Bakhoda, et al, "*Analyzing CUDA Workloads Using a Detailed GPU Simulator*," ISPASS-2009, 2009

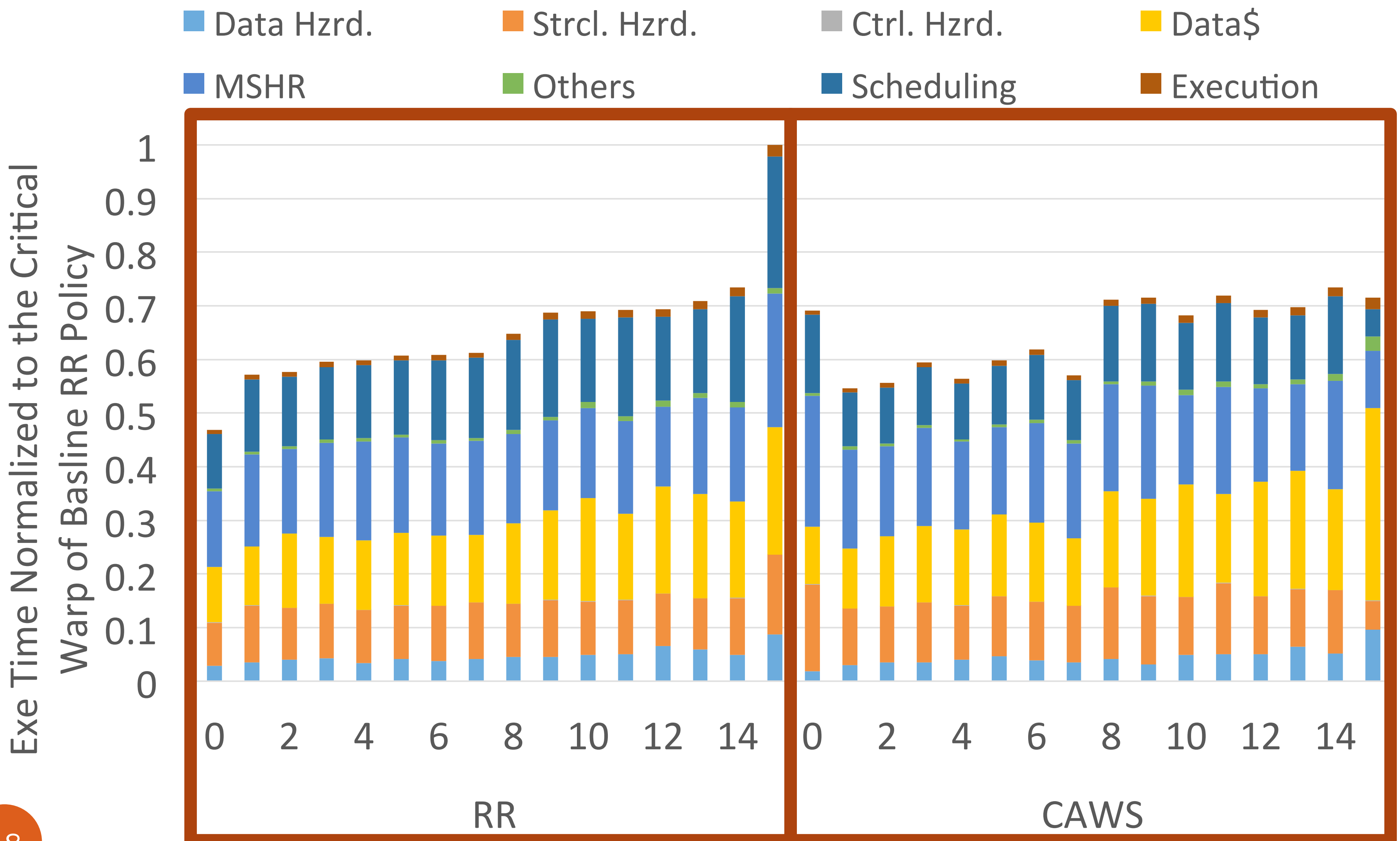
CAWS Speedup with Oracle Knowledge



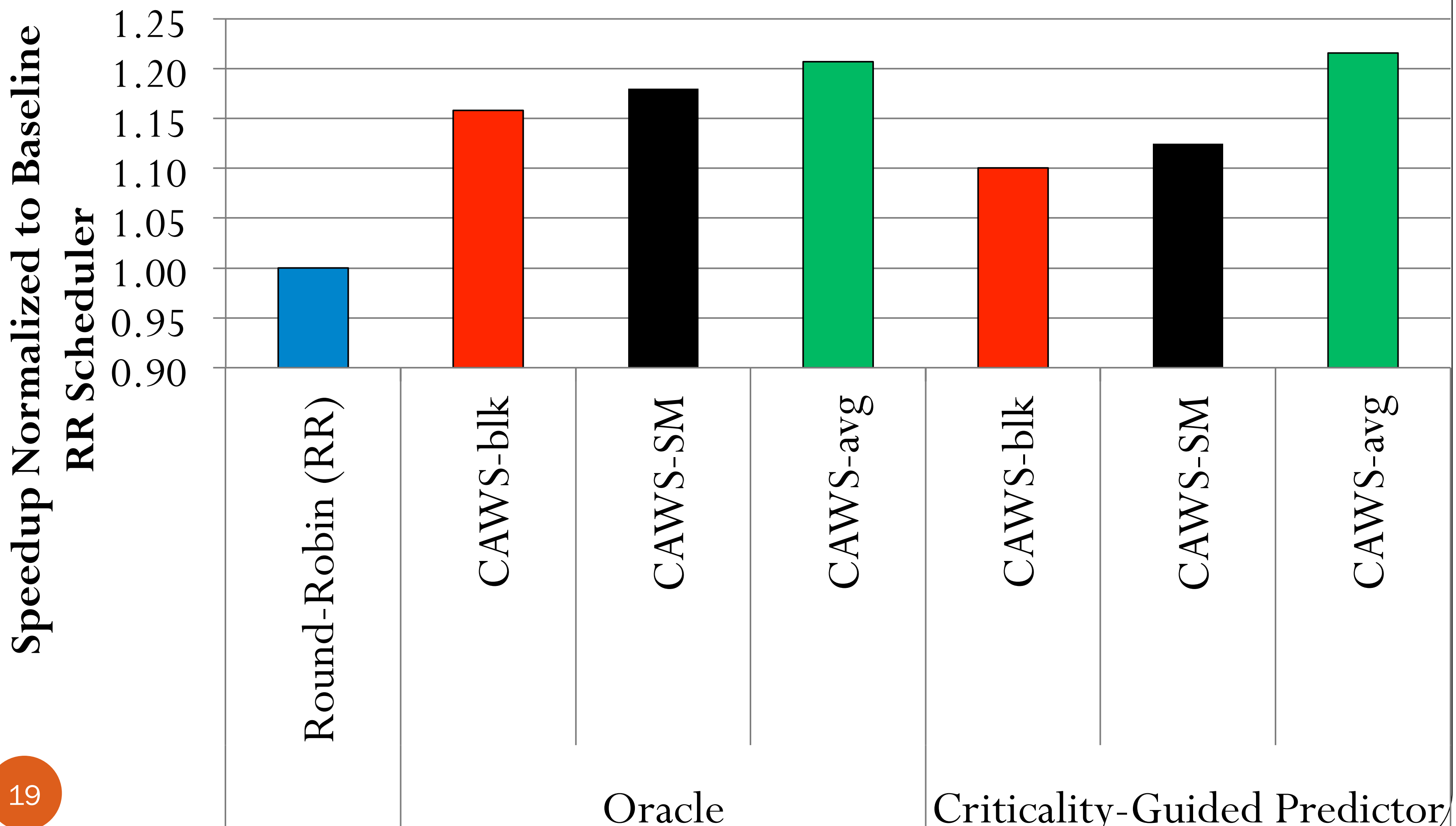
*T. G. Rogers, "*Cache-conscious Wavefront Scheduling*," MICRO-45, 2012

* *V. Narasiman, et al. "*Improving GPU Performance via Large Warps and Two-level Scheduler*," MICRO-44, 2011

Latency Characterization of *bfs*



CAWS with Criticality Prediction on *bfs*



Outline

- GPU Latency Characterization
- Warp/Wavefront Criticality
- Criticality-Aware Warp Scheduling (CAWS)
- Discussion and Conclusion

Research Questions

- Is the current GPU design good enough to hide execution latency?
- What kind of execution latency is hidden under the GPU scheduler?
- How do we propose a better scheduling policy to improve GPU's latency hiding ability?

Conclusion

- This is the first latency characterization algorithm for GPUs that enables a deep understanding of how well the latency hiding ability is for modern GPU schedulers
- We present the GPGPU workloads' results to indicate places for performance improvement.
- We design a family of CAWS policies to improve GPGPU workload performance by equalizing warp execution time.
- The CAWS policies can potentially achieve 17% of performance improvement on average.

Thank You!

CAWS: Criticality-Aware Warp Scheduler for GPGPU Workloads

Shin-Ying Lee and Carole-Jean Wu

Arizona State University

BACKUP

bfs Algorithm

```
while(not visited all node)
        kernel_1
                travers all nodes' children in current depth
        /* an implicit barrier here */

        kernel_2
                go to next depth
        /* an implicit barrier here */
```

CAWS Implementation

- Implemented by count-down counters
  - Every warp has its own priority counter.
  - A priority counter's initial value is corresponding the warp's priority/criticality
  - The counter value decrements cycle by cycle.
  - Scheduler picks up the warp having the lowest counter value to be issued.
  - Once a warp is issued, its counter value is reset to the initial value.

Criticality Inversion

- A fast warp becomes a new critical warp due to starving.
- It may limit the overall speedup or make performance even worse.

Policy	Speedup	Criticality Inversion
CAWS-blk	1.16	8.89%
CAWS-SM	1.18	2.22%
CAWS-Avg	1.21	0%

Comparison of CAWS Policies

	Pros	Cons
CAWS-Blk	To quickly finish a thread block	Useless for thread blocks having no critical warp
CAWS-SM	Useful for cases such that critical warps mapped to thee same thread block	Some thread blocks might get starving
CAWS-Avg	Criticality inversion avoidance	More complicated to implement

Latency Characterization Comparison