CAWA: Coordinated Warp Scheduling and Cache Prioritization for Critical Warp Acceleration of GPGPU Workloads

Shin-Ying Lee   Akhil Arunkumar   Carole-Jean Wu
Arizona State University
6/17/2015
Nowadays GPUs are used to process parallel workloads.

Hundreds or thousands of threads are running concurrently:
- thread-block
- warp/wavefront
Modern GPU Architecture

• Modern GPUs consist of multiple stream-multiprocessors (SMs), which are similar to SIMD processors.
The Warp Criticality Problem

- Significant warp execution time disparity for warps in the same thread block

Research Questions

• What is the source of warp criticality?

• How can we effectively accelerate critical warp execution?
Outline

• Introduction and Motivation
• Source of Warp Criticality
• CAWA: Coordinated Criticality-Aware Warp Acceleration
  – CPL: Criticality Prediction Logic
  – gCAWS: greedy Criticality-Aware Warp Scheduler
  – CACP: Criticality-Aware Cache Prioritization
• Methodology and Evaluation Results
• Conclusion
Source of Warp Criticality

• Workload Imbalance
• Diverging Branch Behavior
• Memory Contention and Memory Access Latency
• Execution Order of Warp Scheduling
Workload Imbalance & Diverging Branch

- Workload imbalance or diverging branch behavior makes warps have different number of dynamic instruction counts.
Memory Contention

• While warps experience different latency to access memory, memory contention can induce warp criticality.
Warp Scheduling Order

- The warp scheduler may introduce additional stall cycles for a ready warp, resulting in warp criticality

![Graph showing execution time normalized to the fastest warp for different warps, sorted by criticality. The graph includes bars for scheduler latency and other latency.](image-url)
Outline

• Introduction and Motivation
• Source of Warp Criticality
• CAWA: Coordinated Criticality-Aware Warp Acceleration
  – CPL: Criticality Prediction Logic
  – gCAWS: greedy Criticality-Aware Warp Scheduler
  – CACP: Criticality-Aware Cache Prioritization
• Methodology and Evaluation Results
• Discussion and Conclusion
CAWA: Criticality-Aware Warp Acceleration
Coordinated warp scheduling and cache prioritization design

• Criticality Prediction Logic (CPL)
  – Predicting and identifying the critical warp at runtime
• greedy Criticality Aware Warp Scheduler (gCAWS)
  – Prioritizing and accelerating the critical warp execution
• Criticality-Aware Cache Prioritization (CACP)
  – Prioritizing and allocating cache lines for critical warp reuse
CAWA: Criticality-Aware Warp Acceleration

- Criticality Prediction Logic (CPL)
  - Predicting and identifying the critical warp at runtime
- greedy Criticality Aware Warp Scheduler (gCAWS)
  - Prioritizing and accelerating the critical warp execution
- Criticality-Aware Cache Prioritization (CACP)
  - Prioritizing and allocating cache lines for critical warp reuse
CAWA_{CPL} : Criticality Prediction Logic

- Evaluating number of additional cycles a warp may experience

\[ \text{Criticality} = n\text{Inst} \times w.\text{CPIavg} + n\text{Stall} \]

- n\text{Inst} is decremented whenever an instruction is executed

---

**Table: Branch path and Per-warp #inst.**

<table>
<thead>
<tr>
<th></th>
<th>Branch path</th>
<th>Per-warp #inst.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Yes:</strong> thread branch divergence</td>
<td></td>
<td>m+n</td>
</tr>
<tr>
<td><strong>No:</strong> thread branch divergence</td>
<td></td>
<td>m</td>
</tr>
<tr>
<td>taken</td>
<td></td>
<td>n</td>
</tr>
</tbody>
</table>
CAWA: Criticality-Aware Warp Acceleration

• Criticality Prediction Logic (CPL)
  – Predicting and identifying the critical warp at runtime

• greedy Criticality Aware Warp Scheduler (gCAWS)
  – Prioritizing and accelerating the critical warp execution

• Criticality-Aware Cache Prioritization (CACP)
  – Prioritizing and allocating cache lines for critical warp reuse

Reduce delay from the scheduler
CAWA_{gCAWS} : greedy Criticality-Aware Warp Scheduler

- Prioritizing warps based on their criticality given by CPL
- Executing warps in a greedy* manner
  - Select the most critical ready-warp
  - Keep on executing the select warp until it stalls

<table>
<thead>
<tr>
<th>Warp Pool</th>
<th>Criticality</th>
</tr>
</thead>
<tbody>
<tr>
<td>Warp 0</td>
<td>5</td>
</tr>
<tr>
<td>Warp 1</td>
<td>10</td>
</tr>
<tr>
<td>Warp 2</td>
<td>3</td>
</tr>
<tr>
<td>Warp 3</td>
<td>7</td>
</tr>
</tbody>
</table>

Warp Scheduler Selection Sequence

Traditional Approach (e.g. RR, 2L, GTO):

\[ W0 \rightarrow W1 \rightarrow W2 \rightarrow W3 \]

gCAWS:

\[ W1 \rightarrow W3 \rightarrow W0 \rightarrow W2 \]

*Rogers et al., “Cache-Conscious Wavefront Scheduler,” MICRO’12
CAWA: Criticality-Aware Warp Acceleration

• Criticality Prediction Logic (CPL)
  – Predicting and identifying the critical warp at runtime

• greedy Criticality Aware Warp Scheduler (gCAWS)
  – Prioritizing and accelerating the critical warp execution

• Criticality-Aware Cache Prioritization (CACP)
  – Prioritizing and allocating cache lines for critical warp reuse

Reduce delay from the memory
Critical Cache Block Reuse Behavior

Design a PC/MEM-based cache reuse predictor

- Capture cache lines that will be used by critical warps
- Capture cache lines that are likely to be reused
**CAWA_{CACP} : Criticality-Aware Cache Prioritization**

*Wu et al., “SHiP: Signatured-based Hit Predictor for High Performance Caching,” MICRO’11*
CAWA\textsubscript{CACP} : Criticality-Aware Cache Prioritization
Outline

• Introduction and Motivation
• Source of Warp Criticality
• CAWA: Coordinated Criticality-Aware Warp Acceleration
  – CPL: Criticality Prediction Logic
  – gCAWS: greedy Criticality-Aware Warp Scheduler
  – CACP: Criticality-Aware Cache Prioritization
• Methodology and Evaluation Results
• Discussion and Conclusion
Experimental Environment

• GPU simulation infrastructure
  – GPGPU-sim v3.2.0*
  – nVIDIA nvcc toolchain v3.2 + gcc v4.4.7
  – nVIDIA GTX480 architecture
  – 15 SMs (maximum 48-warps/8-blocks per SM)
  – 16kB (8-sets/16-ways) L1D$ per SM
  – 768kB (64-sets/16-ways) shared L2$

• Benchmarks
  – 12 applications from Rodinia benchmark suites**
  – including 7 applications with high degree of criticality

*Bakhoda et al., “Analyzing CUDA Workloads Using a Detailed GPU Simulator,” ISPASS’09

**Che et al., “Rodinia: A Benchmark Suite for Heterogeneous Computing,” ISPASS’09
CAWA aims to retain data for critical warps


Cache conscious wavefront scheduling. Rogers, O’Connor, and Aamodt. MICRO `12.
gCAWS Performance Improvement

**IPC Normalized to Baseline RR**

- **bfs**
- **b+tree**
- **heartwall**
- **kmeans**
- **needle**
- **srad_1**
- **strcltr Small**
- **GMEAN (selected)**
- **GMEAN (all 12 apps)**

### Performance Improvement
- **gCAWS**
- **CAWA**

- **2.63x**
- **3.13x**

**Large memory footprint**

**Static cache partitioning needs improvement**
Performance Improvement with CAWA

CACP can effectively reduce cache interference with any schedulers.

Schedulers need modification for robust and higher performance improvement with CACP.

CAWA CACP

MPKI Normalized to Baseline RR

bfs  b+tree  heartwall  kmeans  needle  srad_1  strcltr_small  GMEAN(selected)  GMEAN (all 12)
Outline

• Introduction and Motivation
• Source of Warp Criticality
• CAWA: Coordinated Criticality-Aware Warp Acceleration
  – CPL: Criticality Prediction Logic
  – gCAWS: greedy Criticality-Aware Warp Scheduler
  – CACP: Criticality-Aware Cache Prioritization
• Methodology and Evaluation Results
• Conclusion
Research Questions

• What is the source of warp criticality?
  – workload imbalance and/or diverging branch
  – different memory access latencies
  – additional scheduling latency

• How can we effectively accelerate critical warp execution?
  – CAWA\textsubscript{gCAWS} allocates more time resources for critical warps
  – CAWA\textsubscript{CACP} retains (1) cache lines that will be reused by critical warps in the critical cache partition and (2) cache lines that will be reused with higher priority
Conclusion

• This is the first work that proposes an effective solution for mitigating the degree of warp criticality by accelerating the execution of critical warps of GPGPUs.

• CAWA consists of
  – An accurate dynamic warp prediction scheme to guide critical warp acceleration
  – A coordinated management scheme to improve warp scheduling and cache prioritization for critical warp acceleration

• Our solution improves GPGPU performance by an average of 23% (9% across all applications).
Thank you!

CAWA: Coordinated Warp Scheduling and Cache Prioritization for Critical Warp Acceleration of GPGPU Workloads

Shin-Ying Lee    Akhil Arunkumar    Carole-Jean Wu
Arizona State University
6/17/2015