Exploiting Inter-Warp Heterogeneity to Improve GPGPU Performance

Rachata Ausavarungnirun

Saugata Ghose, Onur Kayiran, Gabriel H. Loh
Chita Das, Mahmut Kandemir, Onur Mutlu

SAFARI
AMD
Carnegie Mellon
Penn State
Overview of This Talk

• **Problem:**
  – A single long latency thread can stall an entire warp

• **Observation:**
  – Heterogeneity: some warps have more long latency threads
  – Cache bank queuing worsens GPU stalling significantly

• **Our Solution: Memory Divergence Correction**
  – **Differentiate each warp** based on this heterogeneity
  – **Prioritizes warps** with fewer of long latency threads

• **Key Results:**
  – **21.8% Better performance and 20.1% better energy efficiency** compared to state-of-the-art
Outline

• Background on Inter-warp Heterogeneity
• Our Goal
• Solution: Memory Divergence Correction
• Results
Latency Hiding in GPGPU Execution

Time

GPU Core Status

Active

Stall

Active

Warp A

Warp B

Warp C

Warp D

GPU Core
Different Sources of Stalls

• Cache misses
Heterogeneity: Cache Misses

- **Warp A**
  - Stall Time
  - Cache Hit
  - Main Memory

- **Warp B**
  - Reduced Stall Time
  - Cache Hit
  - Main Memory

- **Warp C**
  - Cache Miss
  - Cache Hit
  - Main Memory

**Time**
Different Sources of Stalls

• Cache misses

• Shared cache queuing latency
Queuing at L2 Banks

• 45% of requests stall 20+ cycles at the L2 queue
• Queuing latency gets worse as parallelism increases
Outline

• Background on Inter-warp Heterogeneity
• Our Goal
• Solution: Memory Divergence Correction
• Results
Our Goals

• Improve performance of GPGPU application
  – Take advantage of warp heterogeneity
  – Lower L2 queuing latency
• Eliminate misses from mostly-hit warp
• Simple design
Outline

• Background on Inter-warp Heterogeneity
• Our Goal

• Solution: Memory Divergence Correction
  – Warp-type Identification
  – Warp-type Aware Cache Bypassing
  – Warp-type Aware Cache Insertion
  – Warp-type Aware Memory Scheduling

• Results
Mechanism to Identify Warp-type

- **Key Observations**
  - Warp retains its ratio of hit ratio
    - Hit ratio $\rightarrow$ number of hits / number of access
  - High intra-warp locality $\rightarrow$ high hit ratio
  - Warps with random access pattern $\rightarrow$ low hit ratio
  - Cache thrashing $\rightarrow$ additional reduction in hit ratio
Mechanism to Identify Warp-type

- **Key Observations**
  - Warp retains its ratio of hit ratio
Mechanism to Identify Warp-type

- **Key Observations**
  - Warp retains its ratio of hit ratio

- **Mechanism**
  - Profile hit ratio for each warp
  - Assign warp-type based on profiling information
  - Warp-type get reset periodically
Warp-types in MeDiC

<table>
<thead>
<tr>
<th>Memory Request</th>
<th>Shared L2 Cache</th>
<th>Memory Scheduler</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bank 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bank 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bank 2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bank n</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Warp Type ID**

- **All-hit**
- **Mostly-hit**
- **Balanced**
- **Mostly-miss**
- **All-miss**

**Request Buffers**

- **Higher Priority**
- **Lower Priority**

**Bypassing Logic**

- Request Buffers

**SAFARI**
MeDiC

- Warp-type aware cache bypassing
- Warp-type aware cache insertion policy
- Warp-type aware memory scheduling
MeDiC

Warp-type aware cache bypassing

Memory Request

Warp Type ID

Bypassing Logic

Request Buffers

Bank 0

Bank 1

Bank 2

Bank n

Shared L2 Cache

Memory Scheduler

Warp-type aware cache insertion policy

Low Priority

High Priority

Any Requests in High Priority

To DRAM

Mostly-miss, All-miss

……

……
Warp-type Aware Cache Bypassing

• **Goal:** Only try to cache accesses that benefit from lower latency to access the cache

• **Our Solution:**
  - **All-miss and mostly-miss** warps $\rightarrow$ Bypass L2
  - **Other** warp-types $\rightarrow$ Allow cache access

• **Key Benefits:**
  - **All-hit and mostly-hit** are likely to stall less
  - Mostly-miss and all-miss accesses $\rightarrow$ likely to miss
  - Reduce queuing latency for the shared cache
MeDiC

Warp-type aware cache bypassing

Mostly-miss, All-miss

Memory Request

Warp Type ID

Bypassing Logic

Request Buffers

Bank 0

Bank 1

Bank 2

Bank n

Shared L2 Cache

Warp-type aware cache Insertion Policy

Low Priority

High Priority

Any Requests in High Priority

To DRAM

Memory Scheduler
Warps Can Fetch Data for Others

- **All-miss and mostly-miss** warps sometimes prefetch cache blocks for other warps
  - Blocks with high reuse
  - Shared address with all-hit and mostly-hit warps

- **Solution:** Warp-type aware cache insertion
Warp-type Aware Cache Insertion

• **Goal:** Ensure cache blocks from **all-miss** and **mostly-miss** warps are **more likely to be evicted**

• **Our Solution:**
  – **All-miss** and **mostly-miss** → Insert at LRU
  – **All-hit, mostly-hit** and balanced → Insert at MRU

• **Benefits:**
  – All-hit and mostly-hit are **less likely** to be evicted
  – **Heavily reused cache blocks** from mostly-miss are likely to **remain in the cache**
MeDiC

Warp-type aware cache bypassing

Warp-type aware cache Insertion Policy

Memory Request

Warp Type ID

Bypassing Logic

Request Buffers

Bank 0

Bank 1

Bank 2

Bank n

Shared L2 Cache

Low Priority

High Priority

Any Requests in High Priority

Memory Scheduler

N

Y

To DRAM

Mostly-miss, All-miss

MeDiC Warp-type Aware Memory Scheduler

Warp-type aware cache bypassing

SAFARI
Not All Blocks Can Be Cached

• Despite the best effort, accesses from mostly-hit warps can still miss in the cache
  – Compulsory misses
  – Cache thrashing

• Solution: Warp-type aware memory scheduler
Warp-type Aware Memory Sched.

- **Goal:** Prioritize mostly-hit over mostly-miss
- **Mechanism:** Two memory request queues
  - High-priority $\rightarrow$ all-hit and mostly-hit
  - Low-priority $\rightarrow$ balanced, mostly-miss and all-miss
- **Benefits:**
  - Memory requests from mostly-hit are serviced first
  - Still maintain high row buffer hit rate $\rightarrow$ only a few mostly-hit requests
Warp-type aware cache bypassing

- Mostly-miss, All-miss

Memory Request

Warp Type ID

Bypassing Logic

Request Buffers

- Bank 0
- Bank 1
- Bank 2
- Bank n

Shared L2 Cache

Low Priority

High Priority

Any Requests in High Priority

Memory Scheduler

Warp-type aware cache Insertion Policy

To DRAM
MeDiC: Example

- Queuing Latency
- Cache/Mem Latency
  - Cache Miss
  - Cache Hit

Warp A
- High Priority
Warp A’
- Lower stall time

Warp B
- Bypass Cache
- MRU Insertion
Warp B’
- Lower queuing latency
Outline

• Background on Inter-warp Heterogeneity
• Our Goal
• Solution: Memory Divergence Correction
• Results
Methodology

• Modified GPGPU-sim modeling GTX480
  – Model L2 queue and L2 queuing latency

• Comparison points:
  – FR-FCFS [Rixner+, ISCA’00]
    • Commonly used in GPGPU-scheduling
    • Prioritizes row-hit requests → Better throughput
  – EAF [Seshadri+, PACT’12]
    • Tracks blocks that is recently evicted to detect high reuse
  – PCAL [Li+, HPCA’15]
    • Uses tokens to limit number of warps that gets to access the L2 cache → Lower cache thrashing
    • Warps with highly reuse access gets more priority
Results: Performance of MeDiC

MeDiC is effective in identifying warp-type and taking advantage of latency heterogeneity.
Results: Energy Efficiency of MeDiC

Performance improvement outweighs the additional energy from extra cache misses.
Other Results in the Paper

• Comparison against PC-based and random cache bypassing policy
  – MeDiC provides better performance
• Breakdowns of WMS, WByp and WIP
  – Each component is effective
• Explicitly taking reuse into account
  – MeDiC is effective in caching highly reuse blocks
• Sensitivity analysis of each individual components
  – Minimal impact on L2 miss rate
  – Minimal impact on row buffer locality
Conclusion

• Problem:
  – A single long latency thread can stall an entire warp

• Observation:
  – Heterogeneity: some warps have more long latency threads
  – Cache bank queuing worsens GPU stalling significantly

• Our Solution: Memory Divergence Correction
  – Differentiate each warp based on this heterogeneity
  – Prioritizes warps with fewer of long latency threads

• Key Results:
  – 21.8% Better performance and 20.1% better energy efficiency compared to state-of-the-art
Exploiting Inter-Warp Heterogeneity to Improve GPGPU Performance

Rachata Ausavarungnirun, Saugata Ghose, Onur Kayiran, Gabriel H. Loh, Chita Das, Mahmut Kandemir, Onur Mutlu