Orchestrated Scheduling and Prefetching for GPGPUs

Adwait Jog, Onur Kayiran, Asit Mishra, Mahmut Kandemir, Onur Mutlu, Ravi Iyer, Chita Das
Parallelize your code!  
Launch more threads!

Is the Warp Scheduler aware of these techniques?

- Multi-threading
- Caching
- Main Memory
- Prefetching

- Improve Memory Scheduling Policies
- Improve Prefetcher (look deep in the future, if you can!)
Our Proposal

- **Prefetch Aware** Warp Scheduler

- **Goals:**
  - Make a **Simple** prefetcher more **Capable**
  - Improve system performance by orchestrating scheduling and prefetching mechanisms

- 25% average IPC improvement over
  - Prefetching + Conventional Warp Scheduling Policy

- 7% average IPC improvement over
  - Prefetching + Best Previous Warp Scheduling Policy
Outline

- Proposal
- Background and Motivation
- Prefetch-aware Scheduling
- Evaluation
- Conclusions
High-Level View of a GPU

Streaming Multiprocessors (SMs)

- CTA
- CTA
- CTA
- CTA

Scheduler

- L1 Caches
- Prefetcher
- ALUs

Interconnect

- Threads
- Warps

Cooperative Thread Arrays (CTAs) Or Thread Blocks

L2 cache

DRAM
Warp Scheduling Policy

- Equal scheduling priority
  - Round-Robin (RR) execution

**Problem**: Warps stall roughly at the same time
TWO LEVEL (TL) SCHEDULING
Accessing DRAM ...

High Bank-Level Parallelism

High Row Buffer Locality

Low Bank-Level Parallelism

High Row Buffer Locality

Idle for a period

Memory Addresses

Legend

Group 1

Group 2

Bank 1

Bank 2
# Warp Scheduler Perspective (Summary)

<table>
<thead>
<tr>
<th>Warp Scheduler</th>
<th>Forms Multiple Warp Groups?</th>
<th>DRAM Bandwidth Utilization</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Bank Level Parallelism</td>
</tr>
<tr>
<td>Round-Robin (RR)</td>
<td>✗</td>
<td>✓</td>
</tr>
<tr>
<td>Two-Level (TL)</td>
<td>✓</td>
<td>✗</td>
</tr>
</tbody>
</table>

- **Round-Robin (RR)**:
  - Forms Multiple Warp Groups: ✗
  - DRAM Bandwidth Utilization:
    - Bank Level Parallelism: ✓
    - Row Buffer Locality: ✓

- **Two-Level (TL)**:
  - Forms Multiple Warp Groups: ✓
  - DRAM Bandwidth Utilization:
    - Bank Level Parallelism: ✗
    - Row Buffer Locality: ✓
Evaluating RR and TL schedulers

Can we further reduce this gap? Via Prefetching?
(1) Prefetching: Saves more cycles

DRAM Requests

Prefetch Requests

Compute Phase (1) Compute Phase (1)

D1 D2 D3 D4

Comp. Phase (2)

D5 D6 D7 D8

Comp. Phase (2)

Saved Cycles

Compute Phase-2 (Group-2) Can Start

Saved Cycles
(2) Prefetching: Improve DRAM Bandwidth Utilization

High Bank-Level Parallelism

High Row Buffer Locality

Memory Addresses

Idle for a period

No Idle period!
Challenge: Designing a Prefetcher

Sophisticated Prefetcher
Our Goal

- Keep the prefetcher **simple**, yet get the performance benefits of a **sophisticated** prefetcher.

To this end, we will design a prefetch-aware warp scheduling policy. **Why?**

A **simple** prefetching does **not** improve performance with **existing** scheduling policies.
Simple Prefetching + RR scheduling

- **Compute Phase (1)**
  - DRAM Requests
  - **D1, D2, D3, D4, D5, D6, D7, D8**

- **Compute Phase (2)**
  - No Saved Cycles
  - Overlap with D2 (Late Prefetch)
  - Overlap with D4 (Late Prefetch)

Time

**RR**
Simple Prefetching + \textit{TL} scheduling

**DRAM Requests**

Group 1
- Compute Phase (1)

Group 2
- Compute Phase (1)

- \[ \text{D1} \rightarrow \text{D2} \rightarrow \text{D3} \rightarrow \text{D4} \]
- \[ \text{D5} \rightarrow \text{D6} \rightarrow \text{D7} \rightarrow \text{D8} \]

**Group 2**
- \text{Comp. Phase (2)}

**Group 1**
- \text{Comp. Phase (2)}

**Overlap with D2** (Late Prefetch)

**Overlap with D4** (Late Prefetch)

\textit{No Saved Cycles (over TL)}
Let’s Try...

X → Simple Prefetcher → X + 4
Simple Prefetching with TL scheduling

Bank 1

Bank 2

Memory Addresses

Useless Prefetch (X + 4)

Useless Prefetches

May not be equal to Y

Y

idle for a period

X + 4
Simple Prefetching with TL scheduling

Compute Phase (1)  Compute Phase (1)  Comp. Phase (2)  Comp. Phase (2)

Time

DRAM Requests

D1  D2  D3  D4
D5  D6  D7  D8

No Saved Cycles (over TL)

Useless Prefetches

Saved Cycles

TL  RR

Saved Cycles

Cycles

Useless Prefetches

Compute Phase (1)  Compute Phase (1)

DRAM Requests

D1  D2  D3  D4
D5  D6  D7  D8

U5  U6  U7  U8

Comp. Phase (2)  Comp. Phase (2)
## Warp Scheduler Perspective (Summary)

<table>
<thead>
<tr>
<th>Warp Scheduler</th>
<th>Forms Multiple Warp Groups?</th>
<th>Simple Prefetcher Friendly?</th>
<th>DRAM Bandwidth Utilization</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Bank Level Parallelism</td>
</tr>
<tr>
<td>Round-Robin (RR)</td>
<td>✗</td>
<td>✗</td>
<td>✔</td>
</tr>
<tr>
<td>Two-Level (TL)</td>
<td>✔</td>
<td>✗</td>
<td>✗</td>
</tr>
</tbody>
</table>
Our Goal

- Keep the prefetcher simple, yet get the performance benefits of a sophisticated prefetcher.

To this end, we will design a prefetch-aware warp scheduling policy

A simple prefetching does not improve performance with existing scheduling policies.
Sophisticated Prefetcher

Prefetch Aware (PA) Warp Scheduler

Simple Prefetcher
Prefetch-aware (PA) warp scheduling

- Prefetch-aware (PA) warp scheduling
- Group 1
- Group 2

Non-consecutive warps are associated with one group

See paper for generalized algorithm of PA scheduler
Simple Prefetching with PA scheduling

Reasoning of non-consecutive warp grouping is that groups can (simple) prefetch for each other (green warps can prefetch for red warps using simple prefetcher)

X → Simple Prefetcher → X + 1
Simple Prefetching with PA scheduling

Cache Hits!
Simple Prefetching with PA scheduling

Compute Phase (1)  Compute Phase (1)

DRAM Requests

D1  D3  D5  D7

D2  D4  D6  D8

Comp. Phase (2)  Comp. Phase (2)

Saved Cycles!!! (over TL)

Compute Phase-2 (Group-2) Can Start

Simple Prefetching with PA scheduling

Saved Cycles!!! (over TL)

Prefetch Requests
DRAM Bandwidth Utilization

18% increase in bank-level parallelism
24% decrease in row buffer locality

Bank 1
Bank 2

High Bank-Level Parallelism
High Row Buffer Locality

Simple Prefetcher

X → X + 1
## Warp Scheduler Perspective (Summary)

<table>
<thead>
<tr>
<th>Warp Scheduler</th>
<th>Forms Multiple Warp Groups?</th>
<th>Simple Prefetcher Friendly?</th>
<th>DRAM Bandwidth Utilization</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Bank Level Parallelism</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Row Buffer Locality</td>
</tr>
<tr>
<td>Round-Robin (RR)</td>
<td>✗</td>
<td>✗</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>Two-Level (TL)</td>
<td>✓</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>Prefetch-Aware (PA)</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>✓ (with prefetching)</td>
</tr>
</tbody>
</table>
Outline

- Proposal
- Background and Motivation
- Prefetch-aware Scheduling
- Evaluation
- Conclusions
Evaluation Methodology

- Evaluated on GPGPU-Sim, a cycle accurate GPU simulator

Baseline Architecture
- 30 SMs, 8 memory controllers, crossbar connected
- 1300MHz, SIMT Width = 8, Max. 1024 threads/core
- 32 KB L1 data cache, 8 KB Texture and Constant Caches
- L1 Data Cache Prefetcher, GDDR3@1100MHz

Applications Chosen from:
- Mapreduce Applications
- Rodinia – Heterogeneous Applications
- Parboil – Throughput Computing Focused Applications
- NVIDIA CUDA SDK – GPGPU Applications
Spatial Locality Detector based Prefetching

Prefetch:- Not accessed (demanded) Cache Lines

Prefetch-aware Scheduler

D = Demand, P = Prefetch

See paper for more details
Improving Prefetching Effectiveness

Fraction of Late Prefetches
- RR+Prefetching: 89%
- TL+Prefetching: 86%
- PA+Prefetching: 69%

Prefetch Accuracy
- RR+Prefetching: 85%
- TL+Prefetching: 89%
- PA+Prefetching: 90%

Reduction in L1D Miss Rates
- RR+Prefetching: 2%
- TL+Prefetching: 4%
- PA+Prefetching: 16%
Performance Evaluation

Results are Normalized to RR scheduling

- RR+Prefetching
- TL
- TL+Prefetching
- Prefetch-aware (PA)
- PA+Prefetching

1.01  1.16  1.19  1.20  1.26

See paper for Additional Results
Conclusions

- Existing warp schedulers in GPGPUs cannot take advantage of simple prefetchers
  - Consecutive warps have good spatial locality, and can prefetch well for each other
  - But, existing schedulers schedule consecutive warps closeby in time → prefetches are too late
- We proposed prefetch-aware (PA) warp scheduling
  - Key idea: group consecutive warps into different groups
  - Enables a simple prefetcher to be timely since warps in different groups are scheduled at separate times
- Evaluations show that PA warp scheduling improves performance over combinations of conventional (RR) and the best previous (TL) warp scheduling and prefetching policies
  - Better orchestrates warp scheduling and prefetching decisions
THANKS!

QUESTIONS?
BACKUP
Effect of Prefetch-aware Scheduling

Percentage of DRAM requests (averaged over group) with:
- 1 miss
- 2 misses
- 3-4 misses

to a macro-block

Recovered by Prefetching

High Spatial Locality Requests
## Working (With Two-Level Scheduling)

### MACRO BLOCK

<table>
<thead>
<tr>
<th>X</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>X + 1</td>
<td>D</td>
</tr>
<tr>
<td>X + 2</td>
<td>D</td>
</tr>
<tr>
<td>X + 3</td>
<td>D</td>
</tr>
</tbody>
</table>

High Spatial Locality Requests

### MACRO BLOCK

<table>
<thead>
<tr>
<th>Y</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>Y + 1</td>
<td>D</td>
</tr>
<tr>
<td>Y + 2</td>
<td>D</td>
</tr>
<tr>
<td>Y + 3</td>
<td>D</td>
</tr>
</tbody>
</table>
Working (With Prefetch-Aware Scheduling)

MACRO BLOCK

<table>
<thead>
<tr>
<th>X</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>X + 1</td>
<td>P</td>
</tr>
<tr>
<td>X + 2</td>
<td>D</td>
</tr>
<tr>
<td>X + 3</td>
<td>P</td>
</tr>
</tbody>
</table>

High Spatial Locality Requests

MACRO BLOCK

<table>
<thead>
<tr>
<th>Y</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>Y + 1</td>
<td>P</td>
</tr>
<tr>
<td>Y + 2</td>
<td>D</td>
</tr>
<tr>
<td>Y + 3</td>
<td>P</td>
</tr>
</tbody>
</table>
Working (With Prefetch-Aware Scheduling)

<table>
<thead>
<tr>
<th>MACRO BLOCK</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td></td>
</tr>
<tr>
<td>X + 1</td>
<td>D</td>
</tr>
<tr>
<td>X + 2</td>
<td></td>
</tr>
<tr>
<td>X + 3</td>
<td>D</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>MACRO BLOCK</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Y</td>
<td></td>
</tr>
<tr>
<td>Y + 1</td>
<td>D</td>
</tr>
<tr>
<td>Y + 2</td>
<td></td>
</tr>
<tr>
<td>Y + 3</td>
<td>D</td>
</tr>
</tbody>
</table>

Cache Hits
Effect on Row Buffer locality

24% decrease in row buffer locality over TL
Effect on Bank-Level Parallelism

18% increase in bank-level parallelism over TL
Simple Prefetching + RR scheduling

Memory Addresses

Bank 1

Bank 2

Bank 1

Bank 2
Simple Prefetching with \( TL \) scheduling

Memory Addresses

Legend

- **Group 1**
- **Group 2**
CTA-Assignment Policy (Example)

Multi-threaded CUDA Kernel

CTA-1  CTA-2  CTA-3  CTA-4

SIMT Core-1

CTA-1  CTA-2
Warp Scheduler
L1 Caches  ALUs

SIMT Core-2

CTA-3  CTA-4
Warp Scheduler
L1 Caches  ALUs