APRES:
Improving Cache Efficiency by Exploiting Load Characteristics on GPUs

Yunho Oh, Keunsoo Kim, Myung Kuk Yoon,
Jong Hyun Park, Won Woo Ro (Yonsei University)
Yongjun Park (Hongik University)
Murali Annavaram (USC)
Inefficient Cache Utilization on GPUs

- Small-Sized L1 Data Cache per Streaming Multiprocessor (SM)
  - Small-Sized L1 Data Cache per Streaming Multiprocessor (SM)
  - Up to 2048 Threads
    - 16~48 KB L1 D$
    - Up to 48 B/Thread

- A Large Portion of Capacity/Conflict Misses (2Cs) on GPUs
  - Decreases GPU performance due to Stalls in Memory Pipeline
Analysis of 2Cs

- Warp 2 Accesses Address 1000.
- Address 1000 is Already Replaced.

Unordered Load Executions Across Warps Increase 2Cs!
Classifying GPU Load Characteristics

Each Static Load Exhibits Same Cache Behavior “Across Warps”.

**High Memory Locality**

- `ld.param rd1, [__Z2]`
- `ld.global fd1, [rd1]`
- `mul.wide rd4, r5, 8`
- `add rd5, rd2, rd4`
- `ld.global fd2, [rd5]`

**Regular Access Pattern**

- `@%2 bra $Lt_1_5634`
- `.loc 14 278 0`
- `div.rn fd4, fd2, fd3`

**Constant, Thread Independent**

**High Memory Locality**

**Miss**
Type 1: Loads Having High Locality

- A Cache Line Can be Re-Referenced by
  - Same Warp (Intra-Warp Locality)
  - Other Warps (Inter-Warp Locality)
Type 2: Loads Having Inter-Warp Strides

- Regular Inter-Warp Strides
- Easy to Predict Memory Addresses to be Accessed In the Future

Inter-Warp Stride = \( \frac{\text{Addr2} - \text{Addr1}}{\text{Warp\_ID2} - \text{Warp\_ID1}} \)

Warp 0

Warp 1

Warp 10

MISS

MISS

Addr 1000

Addr 2000

Addr 11000

Inter-Warp Stride = \( \frac{2000 - 1000}{10 - 1} \) = \( \frac{1000}{9} \) = 111.11

Inter-Warp Stride = \( \frac{11000 - 2000}{10 - 1} \) = \( \frac{9000}{9} \) = 1000
- Type 1 Loads Needs to be Executed Before Cache Lines are Evicted.
- Data Prefetching Increases Cache Hits in Type 2 Loads.
SOLUTION:
Adaptive PREFetching and Scheduling
How APRES Works

- Warp Grouping and Prioritization
- Prefetching for Warp Issuing
- Same Static Load

Previous Example

APRES
APRES Architecture

- **Locality Aware Warp Scheduler (LAWS)** Groups and Prioritizes Warps.

- **Scheduling Aware Prefetcher (SAP)** Brings Data for Warps in a Group.
How LAWS Works

- Warps Execute Instructions In-Order Fashion.
- LAWS Groups Warps with “PC Address of Lastly Executed Load”.

<table>
<thead>
<tr>
<th>PC</th>
<th>Instruction</th>
<th>Warp 0</th>
<th>Warp 1</th>
<th>Warp 2</th>
<th>Warp 3</th>
<th>Warp 4</th>
<th>Warp 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0F8</td>
<td>add r1, r2, r3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x100</td>
<td>ld.global r4, [r1]</td>
<td>Warp 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x1F8</td>
<td>mul r5, r1, 4</td>
<td></td>
<td></td>
<td>Warp 2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x200</td>
<td>ld.global r6, [r5]</td>
<td>Warp 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x2F0</td>
<td>add r6, r1, r4</td>
<td></td>
<td></td>
<td></td>
<td>Warp 3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x300</td>
<td>ld.global r7, [r6]</td>
<td>Warp 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

| Current PC | 0x200 | 0x300 | 0x1F8 | 0x2F0 | 0x100 | 0x0F8 |
| Last Load PC | 0x200 | 0x300 | 0x100 | 0x200 | 0x100 | NULL |

HIT: Warp 0, 2, and 4 Are Prioritized.
MISS: Warp IDs (0, 2, 4) Are Sent to SAP.
How SAP Works

- **Inter-Warp Stride Prefetcher**

- **LAWS Prioritizes Prefetching Target Warps.**

  PC       Instruction
  0x0F8    add r1, r2, r3
  0x100    ld.global r4, [r1]
  \[...\]
  0x1F8    mul r5, r1, 4
  \[MISS\] ld.global r6, [r5]
  \[...\]
  0x2F0    add r6, r1, r4
  0x300    ld.global r7, [r6]

  **Addresses Accessed In This Load**

  Warp 1
  Warp 2
  Warp 3
  Warp 4
  Warp 5

  Inter-Warp Stride
  Prefetch Inter-Warp Stride
  Prefetch

  More Details in Paper

  Regular!

  \[
  \begin{align*}
  2000 - 2300 &= 300 \\
  0 - 3 &= -3 \\
  2300 - 2100 &= 200 \\
  3 - 1 &= 2 \\
  \end{align*}
  \]

  \[= 100\]
Performance of APRES

- **Performance Improvement in Memory-Intensive Applications**
  - CCWS: 17.8%, CCWS+STR: 24.4%
  - LAWS: 19.0%, APRES: 31.7%

- **24.2% of Performance Improvement in All Applications**

(Baseline: LRR without Prefetching)
How Much APRES Improves Cache Hits

- LAWS Makes 4.3% More Hit-after-Hit Than CCWS.

- APRES Makes 5.6% More Hit-after-Hit Than CCWS+STR.

![L1 Cache Hit Ratio](chart.png)

Memory-Intensive Applications

- Baseline
- CCWS
- LAWS
- CCWS+STR
- APRES
Effect on Prefetching

- **Early Eviction Ratio**
  - CCWS+STR: 13.0%
  - APRES: 8.6%
Conclusion

- Loads with High Memory Locality are Continuously Executed by LAWS.

- Loads with Strided Behavior are Hit in Cache with SAP.

- APRES Improves Cache Efficiency on GPUs.
  - Two load characteristics are properly exploited with adaptively cooperating warp scheduling and inter-warp stride prefetching.
  - 31.7% of performance improvement is achieved in memory-intensive workloads.
Thank You
Backup Slides
Evaluation Environment

- **Simulation Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clock Frequency</td>
<td>1.4GHz</td>
</tr>
<tr>
<td>SMs / GPU</td>
<td>15</td>
</tr>
<tr>
<td>Warp Scheduling Policy</td>
<td>LRR/GTO/CCWS/MASCAR/PA</td>
</tr>
<tr>
<td>SIMT Lane Width</td>
<td>32</td>
</tr>
<tr>
<td>Max # of Warps / SM</td>
<td>48</td>
</tr>
<tr>
<td>Max # of Threads / SM</td>
<td>1536</td>
</tr>
<tr>
<td>Register File Size</td>
<td>128 KB</td>
</tr>
<tr>
<td>Max Registers / SM</td>
<td>32,768</td>
</tr>
<tr>
<td># of Register Banks</td>
<td>32</td>
</tr>
<tr>
<td>Bit Width / Bank</td>
<td>128-bit</td>
</tr>
<tr>
<td># of Entries / Bank</td>
<td>256</td>
</tr>
<tr>
<td>L2 Cache</td>
<td>8-way, 768 KB, 128B line, 200 cycles latency</td>
</tr>
<tr>
<td>DRAM</td>
<td>6-partitioned, 924MHz, 440 cycles latency</td>
</tr>
</tbody>
</table>

- **Benchmarks**
  - GPGPU-sim, Rodinia benchmark suite, Parboil benchmark suite
### Characteristics of Loads in GPU Applications

**%Load**: Portion of each load among total load executions, **#L/#R**: # of unique cache lines per reference

**Miss Rate**: L1 data cache miss rate, **Stride**: Stride in bytes, **%Stride**: Portion of stride among total stride detected

#### Load with Small #L/#R: High Memory Locality

<table>
<thead>
<tr>
<th>App</th>
<th>PC</th>
<th>%Load</th>
<th>#L/#R</th>
<th>Miss Rate</th>
<th>Stride</th>
<th>%Stride</th>
</tr>
</thead>
<tbody>
<tr>
<td>BFS</td>
<td>0x110</td>
<td>51.6%</td>
<td>0.04</td>
<td>0.78</td>
<td></td>
<td>16.3%</td>
</tr>
<tr>
<td></td>
<td>0xF0</td>
<td>26.4%</td>
<td>0.12</td>
<td>0.90</td>
<td></td>
<td>13.3%</td>
</tr>
<tr>
<td></td>
<td>0x198</td>
<td>9.5%</td>
<td>0.11</td>
<td>0.83</td>
<td></td>
<td>14.7%</td>
</tr>
<tr>
<td>MUM</td>
<td>0x7A8</td>
<td>66.2%</td>
<td>0.01</td>
<td>0.17</td>
<td></td>
<td>36.3%</td>
</tr>
<tr>
<td></td>
<td>0x460</td>
<td>21.3%</td>
<td>0.04</td>
<td>0.04</td>
<td></td>
<td>46.8%</td>
</tr>
<tr>
<td></td>
<td>0x8A0</td>
<td>12.3%</td>
<td>0.07</td>
<td>0.17</td>
<td></td>
<td>34.3%</td>
</tr>
<tr>
<td>SPMV</td>
<td>0x1E0</td>
<td>51.5%</td>
<td>0.13</td>
<td>0.32</td>
<td></td>
<td>24.0%</td>
</tr>
<tr>
<td></td>
<td>0x200</td>
<td>23.8%</td>
<td>0.25</td>
<td>0.25</td>
<td></td>
<td>19.3%</td>
</tr>
<tr>
<td></td>
<td>0xE0</td>
<td>7.2%</td>
<td>0.65</td>
<td>0.81</td>
<td></td>
<td>12.5%</td>
</tr>
<tr>
<td>KM</td>
<td>0xE8</td>
<td>100.0%</td>
<td>0.03</td>
<td>0.99</td>
<td>4352</td>
<td>78.2%</td>
</tr>
<tr>
<td>PA</td>
<td>0x2210</td>
<td>51.7%</td>
<td>0.03</td>
<td>0.98</td>
<td>8832</td>
<td>42.7%</td>
</tr>
<tr>
<td></td>
<td>0x2230</td>
<td>39.9%</td>
<td>0.002</td>
<td>0.16</td>
<td></td>
<td>36.2%</td>
</tr>
<tr>
<td></td>
<td>0x2088</td>
<td>3.2%</td>
<td>0.02</td>
<td>0.02</td>
<td>256</td>
<td>91.5%</td>
</tr>
</tbody>
</table>
# Characteristics of Loads in GPU Applications

## Loads with Regular Strides

<table>
<thead>
<tr>
<th>App</th>
<th>PC</th>
<th>%Load</th>
<th>#L/#R</th>
<th>Miss Rate</th>
<th>Stride</th>
<th>%Stride</th>
</tr>
</thead>
<tbody>
<tr>
<td>NW</td>
<td>0x490</td>
<td>18.9%</td>
<td>0.98</td>
<td>1.0</td>
<td>-1966080</td>
<td>16.3%</td>
</tr>
<tr>
<td></td>
<td>0xD18</td>
<td>18.8%</td>
<td>0.97</td>
<td>1.0</td>
<td>-1966080</td>
<td>13.3%</td>
</tr>
<tr>
<td></td>
<td>0x108</td>
<td>1.8%</td>
<td>0.94</td>
<td>1.0</td>
<td>-1966080</td>
<td>14.7%</td>
</tr>
<tr>
<td>LUD</td>
<td>0x20F0</td>
<td>30.2%</td>
<td>0.58</td>
<td>0.32</td>
<td>2048</td>
<td>66.6%</td>
</tr>
<tr>
<td></td>
<td>0x2080</td>
<td>30.2%</td>
<td>0.57</td>
<td>0.25</td>
<td>2048</td>
<td>83.3%</td>
</tr>
<tr>
<td></td>
<td>0x22E0</td>
<td>30.1%</td>
<td>0.66</td>
<td>0.81</td>
<td>2048</td>
<td>77.3%</td>
</tr>
<tr>
<td>SRAD</td>
<td>0x250</td>
<td>31.2%</td>
<td>0.99</td>
<td>0.99</td>
<td>16384</td>
<td>78.2%</td>
</tr>
<tr>
<td></td>
<td>0x230</td>
<td>31.2%</td>
<td>0.99</td>
<td>1.0</td>
<td>16384</td>
<td>75.0%</td>
</tr>
<tr>
<td></td>
<td>0x350</td>
<td>31.2%</td>
<td>0.52</td>
<td>0.99</td>
<td>16384</td>
<td>80.7%</td>
</tr>
<tr>
<td>KM</td>
<td>0xE8</td>
<td>100.0%</td>
<td>0.03</td>
<td>0.99</td>
<td>4352</td>
<td>78.2%</td>
</tr>
<tr>
<td>PA</td>
<td>0x2210</td>
<td>51.7%</td>
<td>0.03</td>
<td>0.98</td>
<td>8832</td>
<td>42.7%</td>
</tr>
<tr>
<td></td>
<td>0x2230</td>
<td>39.9%</td>
<td>0.002</td>
<td>0.16</td>
<td>0</td>
<td>36.2%</td>
</tr>
<tr>
<td></td>
<td>0x2088</td>
<td>3.2%</td>
<td>0.02</td>
<td>0.02</td>
<td>256</td>
<td>91.5%</td>
</tr>
<tr>
<td>BP</td>
<td>0x3F8</td>
<td>19.4%</td>
<td>0.59</td>
<td>1.0</td>
<td>128</td>
<td>75.5%</td>
</tr>
<tr>
<td></td>
<td>0x408</td>
<td>19.4%</td>
<td>0.59</td>
<td>1.0</td>
<td>128</td>
<td>64.1%</td>
</tr>
<tr>
<td></td>
<td>0x478</td>
<td>19.4%</td>
<td>0.59</td>
<td>0.03</td>
<td>128</td>
<td>67.1%</td>
</tr>
</tbody>
</table>
Performance of STR and SLD

- **STR prefetching**

- **SLD prefetching**
Motivations: Prefetching on GPUs

- **Previously Proposed Techniques on GPUs**
  - Stride-based prefetching (STR)\(^1\)
  - Spatial Locality Detection (SLD) based prefetching\(^2\)

- **Performance Improvement (with Advanced Warp Scheduler)**
  - STR: 17.5% (with Cache-Conscious Wavefront Scheduler\(^3\))
  - SLD: 9.4% (with Prefetch-Aware Scheduler\(^2\))

- **Early Eviction Problem**
  - Delays other prefetching and demand requests

---

1. Lee et al. Many-Thread Aware Prefetching Mechanisms for GPGPU Applications. MICRO 2010
2. Jog et al. Orchestrated Scheduling and Prefetching for GPGPUs. ISCA 2013
4. Sethia et al. MASCAR: Speeding up GPU Warps by Reducing Memory Pitstops. HPCA 2015

---

![Graph showing performance improvement and early eviction problem](image_url)
Prefetching Techniques on GPUs Do Not Utilize Excessive Bandwidth.

- APRES reduces average traffics by 2.1%.
- CCWS+STR reduces traffics by 3.8%.
Dynamic Energy Consumption

- **Total Energy Consumption Is Reduced.**
  - Power overhead of APRES is less than 3%.
  - APRES reduces energy consumption by 10.8%.