Cache-Conscious Wavefront Scheduling
Timothy G. Rogers\textsuperscript{1}, Mike O’Connor\textsuperscript{2}, Tor M. Aamodt\textsuperscript{1}
\textsuperscript{1}The University of British Columbia \textsuperscript{2}AMD Research

1 - Wavefront Locality

<table>
<thead>
<tr>
<th>Application</th>
<th>Sensitivity</th>
<th>32k L1D</th>
<th>8M L1D</th>
</tr>
</thead>
<tbody>
<tr>
<td>BFS</td>
<td>4.2</td>
<td>36.3</td>
<td></td>
</tr>
<tr>
<td>KMN</td>
<td>4.2</td>
<td>36.3</td>
<td></td>
</tr>
<tr>
<td>MEMC</td>
<td>4.2</td>
<td>36.3</td>
<td></td>
</tr>
<tr>
<td>GC</td>
<td>4.2</td>
<td>36.3</td>
<td></td>
</tr>
<tr>
<td>HMEAN-HCS</td>
<td>4.2</td>
<td>36.3</td>
<td></td>
</tr>
</tbody>
</table>

- Intra-wavefront locality
- Inter-wavefront locality

Applications: Breadth First Search (BFS), K-Means (KMN), Memcached-GPU (MEMC), Garbage Collection (GC)

Limit study using an unbounded L1D

AVG-Highly Cache Sensitive

Intra-wavefront locality in a GPU kernel

Example 1: Example graph algorithm kernel run by each scalar thread.

```c
int node_degree = node[thread_id].node_degree;
int thread_first_edge = node[thread_id].thread_first_edge;
for (int i = node_degree; i < node_degree + 1; i++) {
    edge_attributes = node.attributes(thread_first_edge + i);
    int neighbour_node_id = edge_attributes.node_id;
    int edge_weight = edge_attributes.weight;
    ...
}
```

2 - Scheduling and Replacement Policy

Example of how scheduling can outperform optimal replacement

- 4-entry fully associative cache

Optimal Replacement using RR scheduler


Wavefront issuing access: W\textsubscript{0}, W\textsubscript{1}, W\textsubscript{2}, W\textsubscript{3}, W\textsubscript{4}

- Using the wavefront scheduler to re-order the access stream can have a larger effect on cache hit rate than a more effective replacement policy using a poor choice of scheduler

3 - CCWS Implementation

- Wavefront Scheduler
  - Locality Scoring System
  - Issue Accesses
  - Memory Unit
  - Lost Locality Detector

Feedback signals when a wavefront has locality

- L1 Data Cache
  - Tag, WID, Data
  - VTA Hit
  - No VTA Hit
  - LDS

Locality Scoring System

- Victim Tags
  - W\textsubscript{0}, Tag, Tag
  - W\textsubscript{1}, Tag, Tag
  - W\textsubscript{2}, Tag, Tag

4 - Results

Performance

- LRR
- GTO
- 2LVL-GTO
- Best-SWL
- CCWS

Locality Captured

Cache misses using LRU and Belady replacement

- LRR
- GTO
- 2LVL-GTO
- Best-SWL
- CCWS
- LRR-BEL
- GTO-BEL
- 2LVL-GTO-BEL
- Best-SWL-BEL
- CCWS-BEL

UBC

AMD Research

100