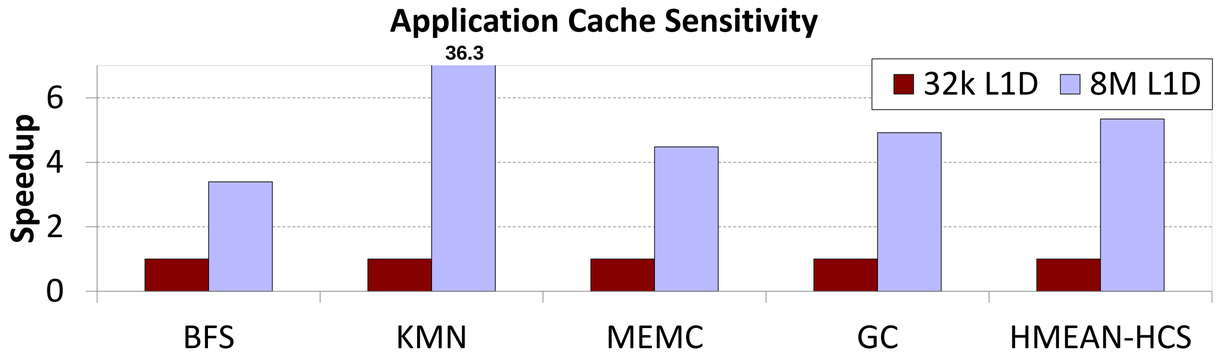
Cache-Conscious Wavefront Scheduling

Timothy G. Rogers[1], Mike O'Connor[2], Tor M. Aamodt[1]

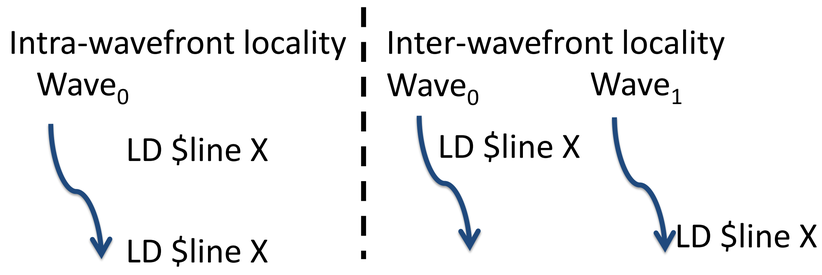
[1]The University of British Columbia [2]AMD Research

1 - Wavefront Locality

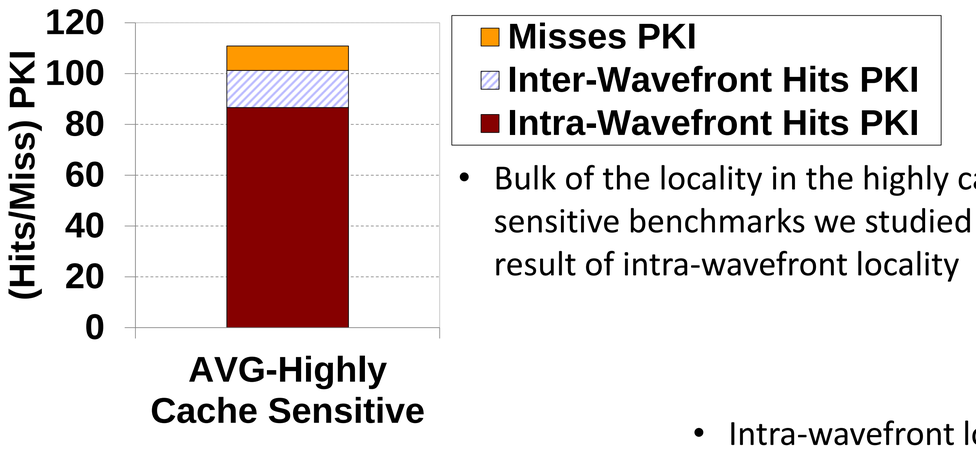
Application Cache Sensitivity



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



Limit study using an unbounded L1D



- Bulk of the locality in the highly cache-sensitive benchmarks we studied was the result of intra-wavefront locality

Intra-wavefront locality in a GPU kernel

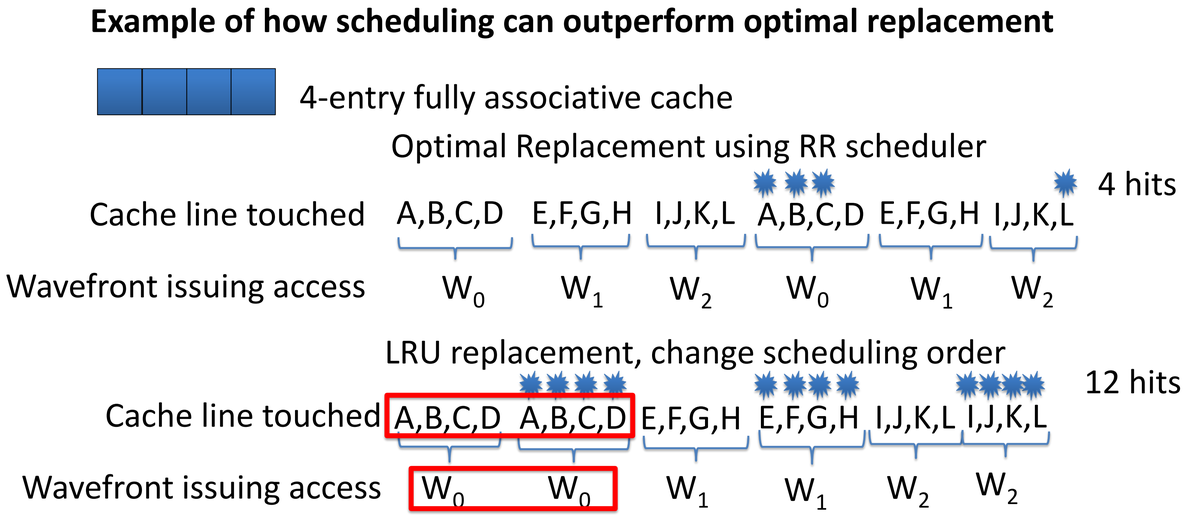
Example 1 Example graph algorithm kernel run by each scalar thread.

```
int node_degree = nodes[thread_id].degree;
int thread_first_edge = nodes[thread_id].starting_edge;
for ( int i = 0; i < node_degree; i++ ) {
    edge_attribtes = edges[thread_first_edge + i];
    int neighbour_node_id = edge_attributes.node;
    int edge_weight = edge_attributes.weight;
    ...
}
```

- Intra-wavefront locality in our highly cache-sensitive benchmarks is largely the result of intra-thread locality
- Inner loop within a scalar thread whose bounds are input data dependent

2 - Scheduling and Replacement Policy

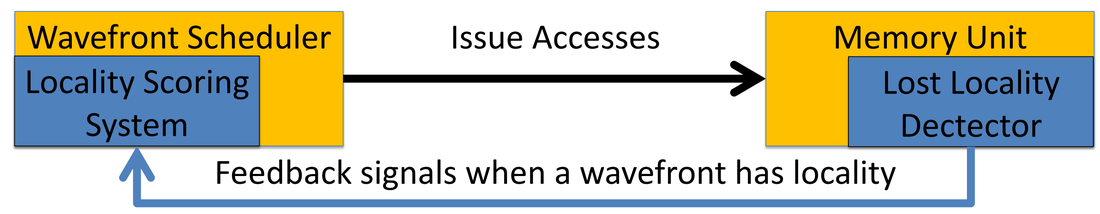
Example of how scheduling can outperform optimal replacement



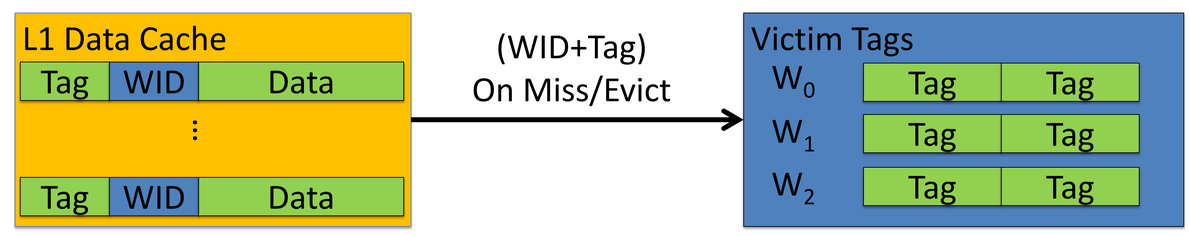
- Using the wavefront scheduler to re-order the access stream can have a larger effect on cache hit rate than a more effective replacement policy using a poor choice of scheduler

3 - CCWS Implementation

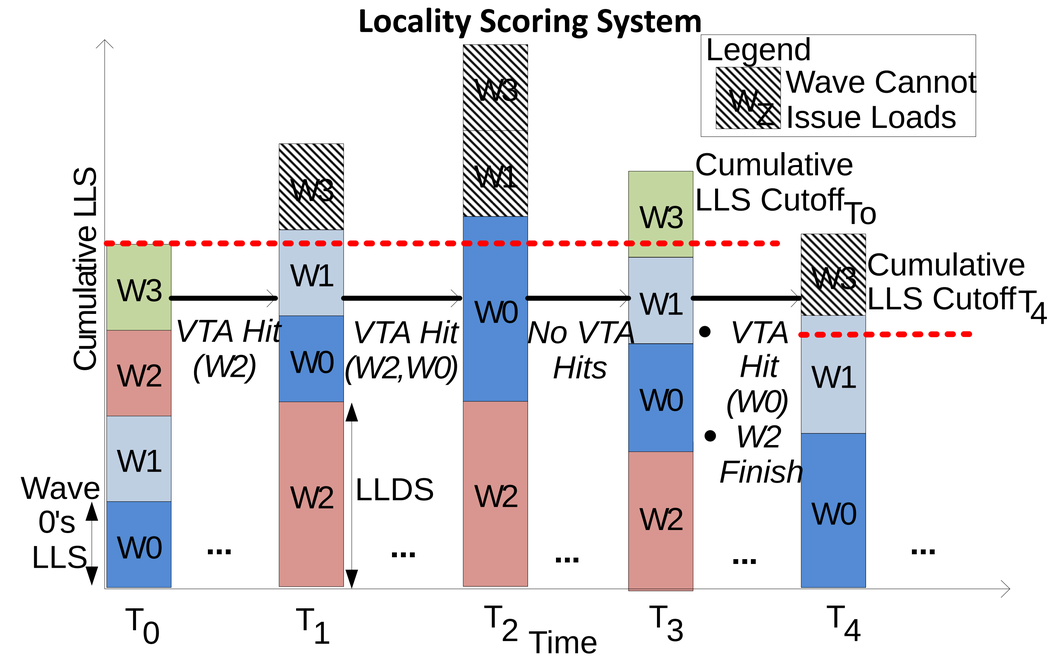
Memory system feedback



Lost Locality Detector



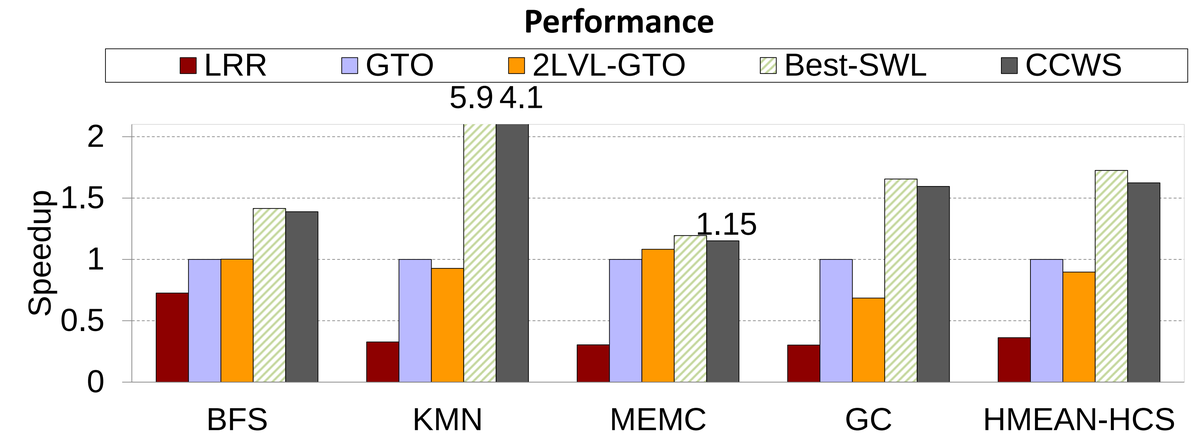
Locality Scoring System



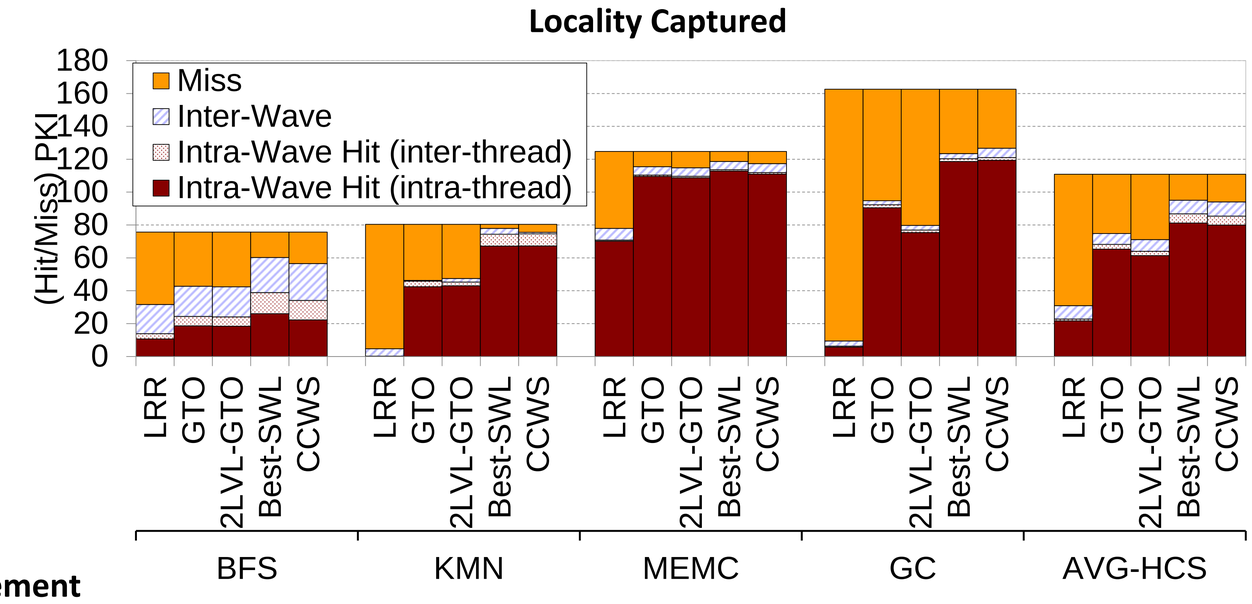
$$LLDS = \frac{VTAHits_{TOTAL}}{InstIssued_{TOTAL}} \times K_{THROTTLE} \times CumulativeLLSCutoff$$

4 - Results

Performance



Locality Captured



Cache misses using LRU and Belady replacement