Data-centric Combinatorial Optimization of Parallel Code

Presenter: Hao Luo
Advisor: Chen Ding

Paper authors: Hao Luo*, Guoyang Chen°, Pengcheng Li*, Chen Ding* and Xipeng Shen°

* University of Rochester
° North Carolina State University
• Motivation

• GPU Cache Model
  • Fully associative cache
  • Set associative cache
  • Non-LRU cache

• Evaluation
Heterogeneity in GPU Programs
Heterogeneity

- Heterogeneous Memory Access Patterns
  - regular v.s. irregular memory accesses

```c
for (int j = warpStart + id; j < warpEnd; j += WARP_SIZE)
{
    int col = cols[j];
    mySum += val[j] *vec[col];
}
```

<table>
<thead>
<tr>
<th>pattern</th>
<th>cols</th>
<th>val</th>
<th>vec</th>
</tr>
</thead>
<tbody>
<tr>
<td>regular</td>
<td>regular</td>
<td>regular</td>
<td>irregular</td>
</tr>
</tbody>
</table>
Heterogeneity

- Multiple On-chip Memory Resources

- thread
  - texture cache
  - shared memory
  - L1 cache
- texture memory
- DRAM
Heterogeneity

- Reconfigurable Cache Structures
  - 16 KB v.s. 48 KB configurations for L1 cache

16 KB, 4-way

48 KB, 6-way
Problem

- Tuning GPU programs is time consuming due to the large search space and complex tuning parameters, e.g. data placement and cache configuration.
Our Approach

• We propose an analytic model to predict GPU cache’s miss ratio to help GPU programmers to tune their code
  • require profiling only once.
  • derive miss ratios of GPU caches for all data placements and multiple sizes.
Cache Model
Model for Fully Associative Cache
Locality Theory

- **Memory Trace**: A sequence of memory references
- **Window**: Subsequence of consecutive references
- **Working Set**: The set of distinct data referenced in a window
- **Working Set Size (WSS)**: The cardinality of a working set
- **Footprint**: Average WSS of all the windows of the same length
memory trace: \(abcabcab\)

window: \(bcabcab\)

window length: 7

working set: \(\{a, b, c\}\)

working set size: 3

(WSS)

footprint: \(\text{WSS}(abcabca) + \text{WSS}(bcabcbab) + \text{WSS}(cababc)\) = 3
Data Placement Problem

- Find the subset of data to place in cache in order to minimize the cache miss ratio.

```c
for (int j = warpStart + id; j < warpEnd; j += WARP_SIZE) {
    int col = cols[j];
    mySum += val[j] * vec[col];
}
```
Predicting Miss Ratio for Data Placement

1. Model each data’s footprint individually

2. Derive miss ratio from footprint

3. To model the miss ratio for co-locating multiple data, simply add their footprints together.
decomposes into 3 subtraces:


cols[0]  _  _  _  cols[32]  _  _  _

_  val[0]  _  _  _  val[32]  _

_  _  vec[47]  _  _  _  vec[174]
Compute Footprint

- Profile a histogram of reuse times.
- Footprint can be derived from the reuse time histogram.

```
vec[0] _ vec[32] _ _ _ vec[0] _ _ _
```

reuse time = 5
hit ratio = \frac{\text{red area}}{\text{red area} + \text{yellow area}}
cols[0]  cols[32]  
+  =  
vals footprints  vals footprints  cols and vals footprints

vals[0]  vals[32]  

Model for Set Associative Cache
Extending Footprint on Set Associative Cache

- Profile footprint for each cache set
- Derive miss ratios between caches of different cache sets from the profiled footprints
6-way, 64 sets

4-way, 32 sets

footprint + footprint = footprint
Model for Non-LRU Cache
GPU Texture Cache

- A 128-byte cacheline is divided into four 32-byte sectors.
- Loading data into cache at sector granularity.
- Evicting data at cacheline granularity.
Applying Footprint on Texture Cache

- Compute reuse time histogram and footprint under two granularities (*sector-granularity* and *cacheline-granularity*).

- Combining *sector-granularity* reuse time histogram with *cacheline-granularity* footprint curve to derive miss ratio.
128-byte Footprint Curve

Cache Size

Fill Time

32-byte Reuse Time Histogram

hit ratio = \frac{\text{red area}}{\text{red area} + \text{yellow area}}
Evaluation
## Comparison

<table>
<thead>
<tr>
<th>Models</th>
<th>Features</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline</td>
<td>simulated set associative LRU cache and texture cache</td>
</tr>
<tr>
<td><strong>Set-RD</strong> <em>(Nugteren et. al. HPCA 14)</em></td>
<td>a fixed number of cache sets and one reuse distance histogram per cache set.</td>
</tr>
</tbody>
</table>
| **PORPLE** *(Chen et. al. Micro 14)* | 1. reuse distance of whole trace, no set associativity.  
2. Assume equal partition among arrays. |
| **Footprint** *(Our model)* | 1. Modeled set associativity and texture cache.  
2. Require scanning through trace only once. |
Miss Ratio Prediction Error

- Absolute error in predicted miss ratios compared against cache simulation

![Bar chart showing the miss ratio prediction error for various applications and libraries. The chart includes bars for Footprint and PORPLE, with error percentages for individual applications such as BFS, Scan, Reduction, Sort, Stencil2D, Trans, ParticleFilter, SpMV, FFT, Triad, MD, CFD, MM, and mean. The chart indicates absolute errors ranging from 0% to 18%, with specific values for each application.]
## Predicted Miss Ratios for Matrix Multiplication

<table>
<thead>
<tr>
<th>Measured Miss Ratio</th>
<th>Predicted Miss Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>simulation</td>
<td>PORPLE</td>
</tr>
<tr>
<td>(most accurate)</td>
<td>Set-RD</td>
</tr>
<tr>
<td>42%</td>
<td>3.4%</td>
</tr>
<tr>
<td></td>
<td>45%</td>
</tr>
<tr>
<td></td>
<td>49%</td>
</tr>
</tbody>
</table>
Time Overhead

- Set-RD and simulation require multiple passes over the trace to derive miss ratios of all data placements and all configurations.

- Footprint and PORPLE can derive miss ratio for all array placements in one trace pass.
Time Overhead

- Normalized time overhead. Each number represents the time needed to derive the miss ratios of all data placements and all configurations.

- The results are normalized to the time required for scanning through the memory trace in one pass for one given configuration.

<table>
<thead>
<tr>
<th>Simulation</th>
<th>Footprint</th>
<th>PORPLE</th>
<th>Set-RD</th>
</tr>
</thead>
<tbody>
<tr>
<td>14</td>
<td>1.6</td>
<td>4.7</td>
<td>524</td>
</tr>
</tbody>
</table>
Summary
The proposed model is an efficient approach to predict GPU cache performance.

Its composability enables accurate modeling of conflict misses and fast evaluation of data placements.
References
(Cox et. al. ISCA ’98) Multi-level texture caching for 3D graphics hardware, Michael Cox, Narendra Bhandari and Michael Shantz, ISCA ’98

(Smith ICSE ’76) On the effectiveness of set associative page mapping and its applications in main memory management. Alan J. Smith, ICSE ’76

(Hill and Smith TOCS ’89) Evaluating associativity in CPU caches. Mark D. Hill, Alan J. Smith, Transactions on Computer Systems, ‘89

(Chen Micro ’14) PORPLE: An extensible optimizer for portable data placement on GPU, Guoyang Chen, Bo Wu, Dong Li, Xipeng Shen, Micro ’14

(Nugteren HPCA ’14) A detailed GPU cache model based on reuse distance theory, Cedric Nugteren, Gert-Jan van den Braak, Henk Corporaal, Henri Bal, HPCA ’14
Thank You!