An Accelerated Procedure for Hypergraph Coarsening on the GPU

Lin Cheng, Hyunsu Cho, and Peter Yoon
Trinity College
Hartford, CT, USA
Outline

• Hypergraph coarsening
• Implementation challenges
• Runtime task planning
• Results
Hypergraph

- Nodes
- **Hyperedges** (nets)
  - Subsets of nodes
Hypergraph

- Hypergraph partitioning
  - Minimize **edge cut**
  - Balance constraint
- NP-complete
Image classification

• Similar images should go to same category
• Need to **compare** images to one another
Hypergraph construction

- Compare multiple pictures at a time
Hypergraph coarsening

• Heuristic: reduce # nodes by fusing
Hypergraph coarsening

• Heuristic: reduce # nodes by fusing
Mondriaan algorithm

- Given a node, find most “similar” neighbor
- Similarity = # hyperedges containing both

![Diagram showing vertices and hyperedges]

$V_2$ and $V_3$ share two hyperedges, hence the affinity is $2$. 

Affinity = 2
Mondriaan algorithm

• Given a node, find most “similar” neighbor
• Similarity = # hyperedges containing both

Binary sparse matrix

Member? (T/F)
Mondriaan algorithm

- Given a node, find most “similar” neighbor
- Similarity = # hyperedges containing both

1. Find edges containing $v_3$
Mondriaan algorithm

- Given a node, find most “similar” neighbor
- Similarity = # hyperedges containing both

1. Find edges containing $v_3$
2. Collect nonzeros
Mondriaan algorithm

• Given a node, find most “similar” neighbor
• Similarity = # hyperedges containing both

3. Inspect column index of each nonzero
Mondriaan algorithm

• Given a node, find most “similar” neighbor
• Similarity = # hyperedges containing both

3. Inspect column index of each nonzero

_similarity scores_
Mondriaan algorithm

- Parallel algorithm:
  Inspect edges in parallel
NVIDIA GPUs: organized in warps
32 threads share one instruction counter

LOAD A[*] INTO X

X

A[*]
Warp divergence

Warp divergence

Thread 5-9: NONE
Warp divergence

Thread 5-9: NONE

stalled
Warp divergence

• Serializes execution
• Caused by **load imbalance**
  • Sparse/irregular data
Mondriaan algorithm

- A naïve strategy results in load imbalance
- Nonzero entry = workload
SHFL to the rescue

• Compiler primitive
• Shuffles content of adjacent registers

\[ x = \text{shfl\_down}(x, 2) \]
SHFL to the rescue

- Compiler primitive
- Shuffles content of adjacent registers
- **Single machine instruction**
- Warp-synchronous; no sync. needed after

```c
x = shfl_down(x, 2)
```
Runtime planning with SHFL

![Diagram showing runtime planning with threads and events]

- Thread 0:
  - Events: $e_0, e_1, e_3$
  - Visible vertices: $v_0, v_1, v_2, v_3$
  - Visible edges: $e_0, e_1, e_3$
  - Value: 6

- Thread 1:
  - Events: $e_5$
  - Visible vertices: $v_0, v_1, v_2, v_3$
  - Visible edges: $e_5$
  - Value: 3

- Thread 2:
  - Events: $e_7$
  - Visible vertices: $v_0, v_1, v_2, v_3$
  - Visible edges: $e_7$
  - Value: 2
Runtime planning with SHFL

Prefix sum

Thread 0

Thread 1

Thread 2

Prefix sum
Runtime planning with SHFL

\[
\text{cei}(11 / 3) = 4
\]
Runtime planning with SHFL

\[ \text{ceil}(11 / 3) = 4 \]
Runtime planning with SHFL

- **Thread 0**
  - $e_0$
  - $e_1$

- **Thread 1**
  - $e_3$
  - $e_5$

- **Thread 2**
  - $e_7$

Equal # of nonzeros

**Range**
- 0 – 3
- 4 – 7
- 8 – 10
Runtime planning with SHFL

Range
0 – 3
4 – 7
8 – 10
Results

- NVIDIA Tesla K20c (5GB mem)
- Intel Xeon E5-2620
- Reference sequential implementation of Mondriaan
Results: long-tailed distribution

Y-axis: # of nonzeros in columns, descending, log-scale

wikipedia

GPU: 50 s
Seq. : 811 s
16 x speed-up

flickr

GPU: 59 s
Seq. : 877 s
15 x speed-up
Results: non-long-tailed distribution

Y-axis: # of nonzeros in columns, descending, log-scale

GPU: 146 s
Seq. : 65 s
0.45 x

GPU: 626 s
Seq. : 41 s
0.07 x
Analysis of results

• Good speedup for data with long-tailed distribution of nonzeros
Analysis of results

• Good speedup for data with **long-tailed distribution of nonzeros**

• Synthetic data
  • First 1,000 columns : 100,000 nonzeros each
  • Next 699,000 columns : all zero
  • Speedup: 123 x
Analysis of results

- Most nodes sparsely connected (<< 32 edges)
- Now: one warp, one instance of Mondriaan

What if this column has < 32 nonzeros?
Work in progress

- Pool multiple instances of Mondriaan into warp
Conclusion

• Implemented hypergraph algorithm handling arbitrary connectivity patterns
• Explored SHFL for task planning

• Future work: more flexible allocation strategy
Acknowledgment

• Trinity College: Student Research Program
• NVIDIA: CUDA Teaching Center