GPU Accelerated Multi-predicate Join Algorithms for Listing Cliques in Graphs

Haicheng Wu¹, Daniel Zinn², Molham Aref², and Sudhakar Yalamanchili¹

¹ Georgia Institute of Technology, ² LogicBlox Inc.

Motivations

- Key ingredients for many graph algorithms such as
  - Triangular clustering
  - Cohesive subgraph
- Extensive attention from
  - Graph theory
  - Database
  - Network analysis

Algorithm 1: LFTJ-GPU

- Split the first trie and map sub-trees to parallel threads
- Parallel threads still run sequential LFTJ
- Intersections are implemented as binary searches
- This method also works for CPU parallelization (LFTJ-CPU)

Example: Mapping to 2 Parallel Threads

Data structure: CSR

Algorithm 2: GPU-Optimized

- Same data structure as Algorithm 1
- Operates breadth-first rather than depth-first
  - Pro: More parallelism
  - Con: Larger memory footprint
- Built on top of ModernGPU library [2]
  - Uses Merge-Path framework to partition data for CTAs/threads
  - Load-balancing between CTAs/threads
  - Good memory usage control
  - Optimized for coalesced memory accesses.
- Rely on sorted and unique property to reduce binary searches

BFS Exploration

- Intersects layer by layer from the top to the bottom
- Divide the problem into
  - Parallel node expansion
  - Parallel array intersection

LFTJ on 3 tries (Triangle Example)

1. Find one intersection in the current layer
2. Down to the next layer

Motif

Goal: Porting LeapFrog TrieJoin to GPUs

- LeapFrog TrieJoin (LFTJ)[1]
  - A general multi-predicate join algorithm
  - Worst case optimal
  - Sequential algorithm

Memo: Porting LeapFrog TrieJoin to GPUs

- Join output size depends on input data
  - Maybe empty or maybe much larger than the input size
  - Naive implementations are inefficient
  - Conservatively reserve large memory to meet the worst case
  - Reserve small memory, abort when not enough
- In Algorithm 2, each input computes its output size first
  - Partition the data by their output size for kernels
  - This is also needed by Merge-Path Framework

Free mem: 10

Output size generated by each input

prefix sum

Total output: 23

Use 3 partitions

Results

- Randomly generated data
- Runs on 1 GTX Titan and 1
  Intel i7-4771

GPU-Optimized runs fastest at all time

Find more measurement and analysis in paper[3] including comparison against GraphLab.

RedFox[4] runs a sequence of binary joins to list cliques

References