Efficient Graph Matching and Coloring on the GPU

Jonathan Cohen
Patrice Castonguay
Key Idea

Cost model: count global syncs

Reasoning:

One kernel invocation = One global sync

1) Read graph data
2) Compute something,
3) Write results
4) Wait for all threads to finish (sync)

Assume “read graph data” and “wait” (sync) dominate

Model is too crude today, but leads to algorithms that scale to future trends (and bigger machines)

Reducing kernel launches generally improves perf

Conclusion: want coloring and matching algorithms requiring fewest number of kernel launches
Graph Coloring

- Assignment of “color” (integer) to vertices, with no two adjacent vertices the same color
- Each color forms independent set (conflict-free)
  - reveals parallelism inherent in graph topology
- “inexact” coloring is often ok
- Our focus: fast, cheap, non-optimal colorings
Parallel Graph Coloring – Luby-Jones

Parallel graph coloring algorithm of Luby / Jones-Plassman
Parallel Graph Coloring – Luby-Jones

- Classic approach: compute array of random numbers
- First optimization: compute a hash function of vertex index on the fly
- Vertex can compute hash number of its neighbors’ indices
- Trades bandwidth for compute, skip kernel to assign random numbers
Parallel Graph Coloring – Luby-Jones

- Round 1: Each vertex checks if local maximum
- => Adjacent vertices can’t both be local maxima
- If max, color=green.
Parallel Graph Coloring – Luby-Jones

- Round 2: Each vertex checks if local maximum, ignoring green
- If max, color=blue
Round 3: Each vertex checks if local maximum, ignoring colored nbrs
If max, color=purple
Round 4: Each vertex checks if local maximum, ignoring colored nbrs
If max, color=red
Round 5: Each vertex checks if local maximum, ignoring colored nbrs
If max, color=white
Round 6: Each vertex checks if local maximum, ignoring colored nbrs
If max, color=yellow
Completes in 6 rounds
Parallel Graph Coloring – Min-Max

Realization: Local min and local max are both independent sets
They are disjoint => can produce 2 colors per iteration
Parallel Graph Coloring – Min/Max

- Round 1: Each vertex checks if it’s a local maximum or minimum.
- If max, color=blue. If min, color=green
Parallel Graph Coloring – Min/Max

- Round 2: Each vertex checks if it’s a local maximum or minimum.
- If max, color=pink. If min, color=red
Parallel Graph Coloring – Min/Max

- Round 3: Each vertex checks if it’s a local maximum or minimum.
- If max, color=purple. If min, color=white
- Improvement: 3 rounds versus 6
Parallel Graph Coloring – Multi-Hash

- Use multiple hash functions to obtain multiple 2-coloring of the graph
- Hash function 1:
Parallel Graph Coloring – Multi-Hash

- Use multiple hash functions to obtain multiple 2-coloring of the graph
- Hash function 2:
Parallel Graph Coloring – Multi-Hash

- Use multiple hash functions to obtain multiple 2-coloring of the graph
- Hash function 3:
Parallel Graph Coloring – Multi-Hash

- Combine all 2-colorings – completes in 1 round!
- Creates well-balanced graph colorings
- Empirically: produces better colorings than Luby-Jones – not sure why
100% Coloring Results

Speedup

- JonesPlassman
- JonesPlassmanHash
- MinMax
- MinMaxHash
- MultiHash

Graph showing speedup results for various benchmarks.
Number of Colors (100% Coloring)
95% Coloring Results

The chart shows the speedup for various benchmarks under different coloring algorithms: JonesPlassman, JonesPlassmanHash, MinMax, MinMaxHash, and MultiHash. The x-axis represents different benchmarks (e.g., af_shell10, bcst38, cant, consph, cop_20k_A, dawson5, hood, ldoor, lock1074, msdoor, pdb1HYS, poisson2D, poisson3Da, poisson3Db, pwtk, shipsec1, thermal1), and the y-axis represents the speedup.

The speedup values vary across the benchmarks, with some showing significant improvements (e.g., msdoor, poisson3Db) and others showing more modest improvements (e.g., af_shell10, bcst38). The MultiHash algorithm generally shows the highest speedup, followed by MinMaxHash and MinMax, with JonesPlassman and JonesPlassmanHash showing lower speedup compared to the other algorithms.
Graph Matching

- Set of edges such that no two edges share a vertex
- “Maximum matching” – matching that includes the largest number of edges
- Equivalent: Independent set on dual of graph
  - independent *pairs* of connected vertices
One-Phase Handshaking
One-Phase Handshaking

- Each vertex extends a hand to its strongest neighbour
Set aggregates

Each vertex checks if its strongest neighbour extended a hand back.
One-Phase Handshaking

- Repeat with unmatched vertices
One-Phase Handshaking
One-Phase Handshaking
One-Phase Handshaking
One-Phase Handshaking
One-Phase Handshaking
One-Phase Handshaking
One-Phase Handshaking
Two-Phase Handshaking
Two-Phase Handshaking

- Extend a first hand to your strongest neighbour
Two-Phase Handshaking

- Extend a second hand to the strongest vertex among those who gave a hand to you
Two-Phase Handshaking

- Keep only edges which have a handshake
- New graph has maximum degree 2
Two-Phase Handshaking

Now do one-phase handshaking
Two-Phase Handshaking

- Find perfect matches
Two-Phase Handshaking

Repeat
Two-Phase Handshaking

- Repeat
Two-Phase Handshaking

Repeat
Two-Phase Handshaking

- Repeat
Two-Phase Handshaking

Repeat
Two-Phase Handshaking

Repeat
N-Way Handshaking
**N-Way Handshaking**

- Extend N hands at once (N=2)
- Similar to first 2 steps of two-phase, but in a single step
N-Way Handshaking

- Discard edges without a match
- Resulting graph has max degree N (N=2)
N-Way Handshaking

Now do one-phase
N-Way Handshaking

- Select perfect matches
N-Way Handshaking

- Repeat
N-Way Handshaking

Repeat
N-Way Handshaking

- Repeat
N-Way Handshaking

Repeat
Graph Matching Performance (90% Matching Target)
Graph Coloring and Matching algorithms can be highly data parallel

Key optimizations:
- More work per thread, fewer global synchronizations
- Replace random numbers with hash functions

One view: recast in terms of generalized Sparse Matrix-Vector product (SpMV)
- For each row (in parallel)
  - Visit each neighbor, compute something
  - Compute reduction
  - Write out single result
Questions?

- Tech report and source code with lots more details is forthcoming
- Thanks to entire NVAMG team