15418 Final Project - Parallel Graph Coloring

Yuehan Hu
Zhanming Liang

Summary

Platforms:
CPU: OpenMP on GHC 8-core Intel CPU
GPU: CUDA on NVIDIA RTX 2080

Goal:
Implement and compare parallel graph coloring on CPU and GPU

Best CPU Implementation:
★ Multi-phase speculative coloring
★ Achieved 6.7× compute-time speedup on 8 threads
★ Maintained near-zero conflict rates (< 0.02%)
★ Preserved color quality matching baseline

GPU Implementation:
★ CUDA speculative coloring
★ Achieved strong kernel-only speedups on large graphs

Experimental Setup:
Benchmarked on 11 graphs that included both synthetic and real-world datasets
Graph sizes ranged from 10K to 1M vertices
All experiments run on the GHC cluster

Background

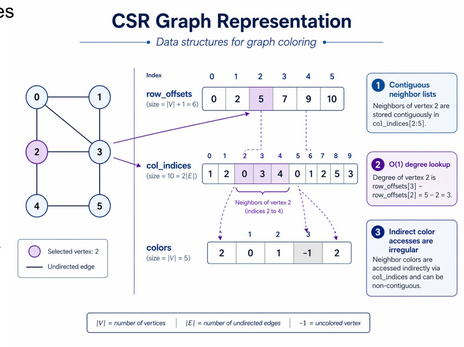
Problem:
Assign a color to each vertex in an undirected graph
Adjacent vertices must have different colors
Goal: use as few colors as possible

Key Data Structures:
CSR graph format
- row_offsets stores the start/end of each vertex's neighbor list
- col_indices stores neighbor IDs contiguously
- gives O(1) degree lookup
- enables efficientneighbor traversal

Why this layout
- good cache behavior for sparse graph traversal
- compact and efficient on both CPU & GPU

Color state
- colors[0..n-1]
- initialized to -1 for uncolored vertices

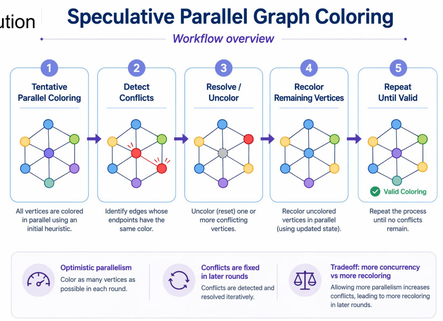
Temporary color tracking
- CPU: per-thread used[] byte array
- GPU: 128-bit register bitmask



Core Algorithm:
For each vertex v:
- scan neighbor colors
- mark used colors
- assign the smallest available color

Work per vertex is O(degree(v))
Total sequential work is O(V + E)

Parallel Strategy Pseudocode:
Speculative coloring with conflict resolution
- Color vertices in parallel
- Detect conflicts between
  - adjacent same-colored vertices
- Recolor only conflicting vertices
  - in the next round
- Repeat rounds until converge
- Typically converges in 1–2 rounds



Approach

CPU:

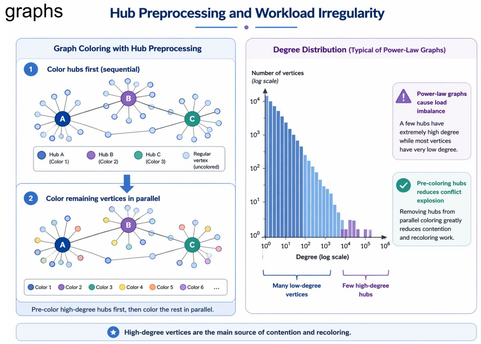
Language / platform: C++17 + OpenMP on the GHC 8-core Intel CPU
Data layout: CSR graph representation with shared colors[] array and per-thread used[] array
Baseline: sequential greedy first-fit coloring
Parallelization strategy: speculative coloring with conflict resolution
- color active vertices in parallel
- detect adjacent same-color conflicts
- reset losing vertices and recolor in the next round

CPU Optimizations:
Hub preprocessing:
- sequentially color very high-degree vertices first
- greatly reduces conflicts on power-law graphs

Largest-Degree-First (LDF) ordering:
- prioritize high-degree vertices before speculative coloring

Adaptive scheduling:
static for regular graphs
dynamic,256 for irregular graphs

Faster preprocessing:
- precomputed degree array
- counting sort for LDF
- skip sorting on regular graphs



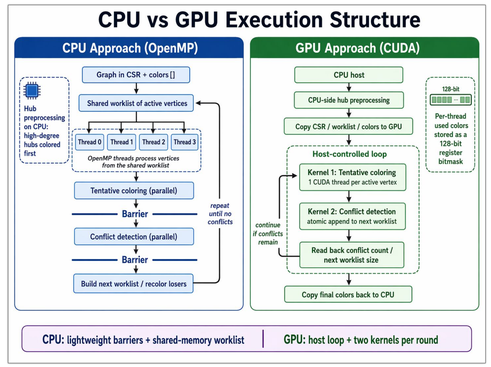
GPU:

Language / platform: CUDA C++ on the NVIDIA RTX 2080
Data layout: same CSR graph format as CPU
Mapping: one CUDA thread per active vertex in the worklist
Per round: two kernels
- tentative coloring
- conflict detection / next-worklist compaction

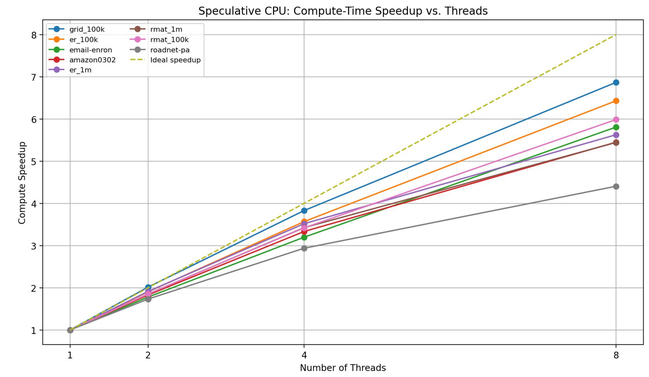
Kernel 1: Tentative Coloring
- One thread per active vertex
- Scans the vertex's neighbors
- Marks neighbor colors
- Assigns the smallest available color

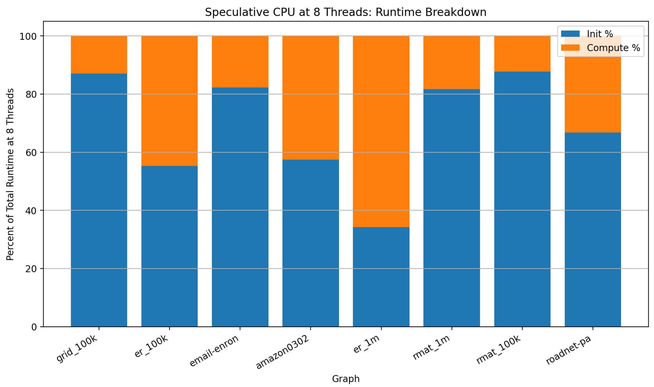
Kernel 2: Conflict Detection
- One thread per active vertex
- same as GPU
- Conflicting vertices are appended to the next worklist using atomics

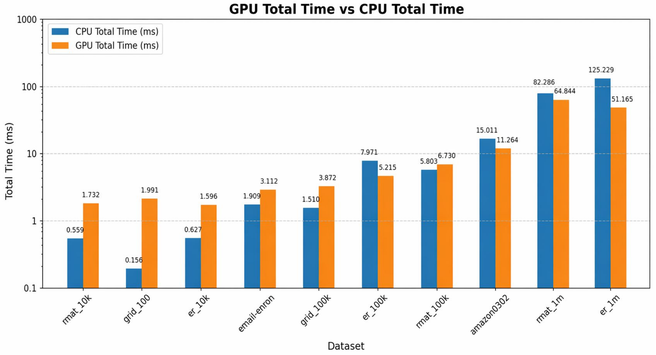
Execution Flow
- CPU launches Kernel 1 and 2 in a loop
- After each round, the CPU reads back the conflict count / next worklist size
- If conflicts remain, another round is launched
- Stops when worklist empty

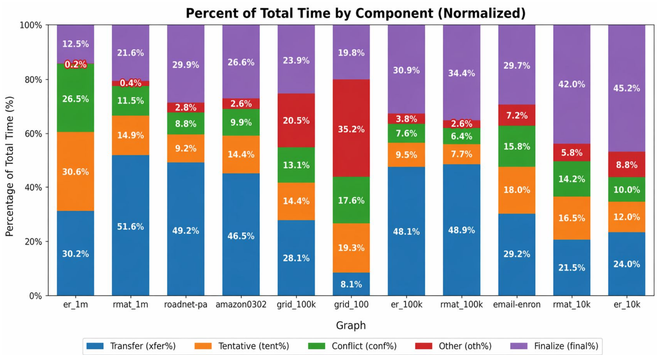


Results









References

- Gebremedhin, A. H., and Manne, F. "Scalable Parallel Graph Coloring Algorithms." Concurrency: Practice and Experience, 12(12): 1131–1146, 2000.
- Jones, M. T., and Plassmann, P. E. "A Parallel Graph Coloring Heuristic." SIAM Journal on Scientific Computing, 14(3): 654–669, 1993.
- Luby, M. "A Simple Parallel Algorithm for the Maximal Independent Set Problem." SIAM Journal on Computing, 15(4): 1036–1053, 1986.
- Çatalyürek, Ü. V., Feo, J., Gebremedhin, A. H., Halappanavar, M., and Pothen, A. "Graph Coloring Algorithms for Multi-core and Massively Multithreaded Architectures." Parallel Computing, 38(10–11): 576–594, 2012.
- Deveci, M., Boman, E. G., Devine, K. D., and Rajamanickam, S. "Parallel Graph Coloring for Manycore Architectures." In IEEE IPDPS, 2016.
- Grosset, A. V. P., Zhu, P., Liu, S., Venkatasubramanian, S., and Hall, M. "Evaluating Graph Coloring on GPUs." In ACM PPoPP, 2011.
- Welsh, D. J. A., and Powell, M. B. "An Upper Bound for the Chromatic Number of a Graph and Its Application to Timetabling Problems." The Computer Journal, 10(1): 85–86, 1967.
- Brélaz, D. "New Methods to Color the Vertices of a Graph." Communications of the ACM, 22(4): 251–256, 1979.
- Buluç, A., Fineman, J. T., Frigo, M., Gilbert, J. R., and Leiserson, C. E. "Parallel Sparse Matrix-Vector and Matrix-Transpose-Vector Multiplication Using Compressed Sparse Blocks." In ACM SPAA, 2009.
- Erdős, P., and Rényi, A. "On Random Graphs I." Publicationes Mathematicae, 6: 290–297, 1959.
- Chakrabarti, D., Zhan, Y., and Faloutsos, C. "R-MAT: A Recursive Model for Graph Mining." In SIAM SDM, 2004.
- The R-MAT generator used for power-law graphs (rmat_10k, rmat_100k, rmat_1m).
- Leskovec, J., and Krevl, A. "SNAP Datasets: Stanford Large Network Dataset Collection." http://snap.stanford.edu/data, 2014.
- Leskovec, J., Adamic, L. A., and Huberman, B. A. "The Dynamics of Viral Marketing." ACM Transactions on the Web, 1(1): 5, 2007.
- Leskovec, J., Lang, K. J., Dasgupta, A., and Mahoney, M. W. "Community Structure in Large Networks: Natural Cluster Sizes and the Absence of Large Well-Defined Clusters." Internet Mathematics, 6(1): 29–123, 2009.
- Dagum, L., and Menon, R. "OpenMP: An Industry-Standard API for Shared-Memory Programming." IEEE Computational Science & Engineering, 5(1): 46–55, 1998.