Milestone Report

Team Members: Jerry Cheng, Julie Wu

URL: https://jwjulie.github.io/15418-betweenness-centrality/milestone-report

WORK COMPLETED SO FAR

Since the proposal, we have successfully established a performance baseline and explored two parallelization strategies:

- **Sequential Baseline:** Implemented Brandes' Algorithm in C++ using both an adjacency list and compressed sparse row format. It can handle different types of data that we've found, including undirected graphs that only store the edges in one direction, data that store the edges multiple times, and data that has extraneous comments.
- **Test file generator:** While we've found decent test data on power-law graphs in the Stanford Large Network Dataset Collection from SNAP, we didn't find many high diameter graphs (things like road networks) that were easy enough to use for testing. We created our own test file generator that generates a noisy grid of nodes (a fully connected grid with nodes and edges randomly removed) to simulate road networks.
- **Coarse-Grained Implementation:** We implemented a root-parallel version where available threads each conduct BFS traversals starting from different root nodes.
- **Fine-Grained Implementation:** We developed an initial version of a frontier-parallel strategy that parallelizes within a single BFS. So far, we have developed three iterations of our synchronization model (the following metrics are obtained on 8 cores):
    1. **Lock-based:** High contention, resulting in a poor 0.1-0.2x speedup.
    2. **Lock-free:** Replaced locks with compare and swap-style code, improving speedup to 0.3-0.5x.
    3. **Optimization:** Mitigated fork-join and memory allocation overheads, currently reaching 0.4-0.85x speedup.

PRELIMINARY RESULTS

- **Coarse-Grained Scaling:** On small graphs (4k nodes), scaling is near-linear on the GHC machines. However, on a 50k-node graph, scaling peaks at 4 cores and drops at 8 cores. As we suspected in the proposal, the thread-private metadata arrays for running 8 concurrent runs of Brandes' exceeds cache capacity when the number of nodes in the graph is high.
- **Fine-Grained Bottlenecks:** Despite moving to lock-free code, we are still seeing sub-serial performance. The current profiling we've done suggests that there are still portions of the forward pass with high contention, but the primary slowdown is from the backward pass, which takes up a larger portion of the runtime on higher thread counts.

Speedup vs. Num threads	Coarse (small, power-law graph)	Fine (small, power-law graph)	Coarse (large, high-diameter graph)	Fine (large, high diameter graph)
2	1.975	0.888	1.848	0.751
4	3.848	0.791	3.079	0.606
8	7.088	0.819	1.353	0.418

REVISED GOALS & DELIVERABLES

We are maintaining our original goals and deliverables since we're about on track with what we originally planned out. Given that we have a better understanding of the challenges involved in refining the parallel Brandes' with BFS implementations we have fleshed out the goals with more details:

PLAN TO ACHIEVE (Success)

- A working sequential baseline for Brandes' algorithm (complete).
- A working multi-core OpenMP implementation (in progress/needs improvement)
    - Coarse-grained version where multiple BFSs are assigned to different cores
    - Fine-grained version where multiple cores operate on the same BFS
        - Comparison of more refined synchronization strategies for frontier updates on the forward pass
        - Comparison of different update methods to speed up backwards pass

HOPE TO ACHIEVE (Reach Goals)

- Implementation on weighted graphs using Dijkstra's for the SSSP algorithm on the forward pass (likely not possible by the poster session)
- CUDA-based GPU implementation and comparison with CPU implementation

POSTER SESSION PLAN

Our posters will include diagrams depicting our coarse and fine-grained approaches to parallelizing Brandes'. We will present a series of graphs analyzing our the parallel implementations in comparison to the sequential Brandes' baseline:

- Speedup across different numbers of threads on GHC machines.
- Runtime/speedup on power-law graphs and high-diameter graphs.
- Total number and per thread cache misses.
- PSC speedup across different numbers of threads and comparison to GHC results.

CONCERNS & UNKNOWNS

While we currently think that replacing a high-contention critical section with a parallel scan, switching from a vertex-parallel to an edge-parallel approach in both the forward and backward pass, and counting successors in the backward pass rather than maintaining strict level-parallelism can improve our fine-grained implementation, we aren't sure if the changes we have in mind will be sufficient enough to improve the speedup by a significant margin. We are also considering ways to decrease the memory requirement for the coarse-grained implementation and will likely need to look into some research to do that.

Revised Schedule

Week	Task	Status	Assignee
Mar 26 - Apr 2	Study betweenness centrality algorithms and related papers on parallelizing graph algorithms. Implement a sequential Brandes' baseline and draft CSR representation. Find datasets for testing.	Done	Both
Apr 3 - Apr 9	Implement coarse-grained OpenMP implementation. Make the test file generator. Test initial speedup on GHC machines.	Done	Julie
Apr 10 - Apr 16	Implement initial fine-grained OpenMP implementations. Test speedup on GHC machines. Work on milestone report due Apr 15.	Done	Both
Apr 17 - Apr 20	Implement the current forward pass improvements we have in mind for the fine-grained OpenMP implementation to improve speedup.	In progress	Jerry
Apr 17 - Apr 20	Implement the current backward pass improvements for the fine-grained OpenMP implementation.	In progress	Julie
Apr 21 - Apr 23	Research further improvements to the coarse- and fine-grained implementation. Evaluate feasibility of djikstra's and cuda implementation (reach goals)	Not started	Both
Apr 24 - Apr 27	Collect data from PSC machines. Create graphs for GHC and PSC performance metrics.	Not started	Both
Apr 28 - Apr 30	Write up the final report and finish the presentation poster.	Not started	Both