15-618 S25
Parallelizing Maximum Flow using CUDA Milestone Report

Names: Alena Lu, Ankita Chatterjee
Andrew IDs: alenalu, ankitac
`https://achstar.github.io/max-flow-gpu/`

1 Updated Schedule

- Week 1: Read research papers to understand sequential and parallel versions of push-relabel algorithm in their entirety. Brainstorm ways to approach parallelization and outline a solution through pseudocode.
- Week 2: Write and test sequential implementation. Develop test cases, which will be varying sized input graphs, and define expected results.
- Week 3: Continue to develop test cases, and write first iteration of CUDA implementation. Evaluate performance and debug. Outline findings and roadblocks in milestone report.

Up to now, we have been pretty much on track with our proposed schedule. We have correct sequential and CUDA implementations of the push-relabel algorithm, and our work from here is to continue iterating on the CUDA implementation to improve its speedup. We'll break up the work for the next two weeks into half-week increments.

- Week 3.5: Continue to debug and improve parallel implementation.
  For Ankita: change the CUDA implementation to not use an adjacency matrix to store residual capacities so that we don't run out of memory when trying to test with hundreds of thousands of edges.
  For Alena: Look into ways we can reduce the amount of data transfer between device and host, since that currently seems to be our bottleneck. Potentially explore the gap relabeling technique instead of global relabeling.
- Week 4: Continue to debug and improve parallel implementation.
  For Ankita: Implement the graph coloring version of the parallel push-relabel algorithm and compare performance between that and the asynchronous method.
  For Alena: Continue to improve upon the asynchronous parallel push-relabel algorithm if performance is still poor compared to the sequential version. If performance

is satisfactory, work on our nice-to-have, which is the OpenMP parallel max flow implementation.

- Week 4.5: Finalize parallel implementations and collect profiling results. Both of us will work on any last minute debugging and tuning to improve performance, and both of us will put together graphs comparing the implementations against each other.
- Week 5: Both of us will analyze and explain our findings, and complete the final report and poster. At this point our parallel implementations should be finalized.

2 Current Progress

So far, we have compiled a variety of test cases representing actual maximum flow problems, most obtained from the University of Waterloo's "Maxflow Problem Instances in Vision" [1]. We also wrote a script to generate smaller test traces for easier debugging. We have also completed a sequential and CUDA implementation of the push-relabel algorithm, both of which are correct according to the tests we have run; we need to continue iterating on the CUDA implementation to improve performance.

3 Updated Goals

We are mostly on track to meet the goals and deliverables we stated in our proposal, however the performance of our CUDA implementation is significantly worse than expected so we may need to spend more time iterating on it than we initially accounted for. As a result we may not be able to implement our nice-to-haves, like the OpenMP implementation of push-relabel. Our updated goals:

1. A working (correct) sequential push-relabel max flow algorithm. This is done.
2. A working CUDA (correct, with non-negligible speedup) parallelization of the push-relabel max flow algorithm. This is partially done; we do observe non-negligible speedup on some smaller test cases, but due to some flaws in our implementation we are unable to test on larger graphs (we run out of memory). We also observe that the CUDA implementation is actually slower than the sequential implementation for many of the smaller test cases (typically those that require more iterations to converge on a maximum flow).
3. Compare execution time and compute speedup across implementations. Evaluate the performance of both implementations on different graph inputs (sparse vs. dense) and evaluate scalability across different graph sizes and varying number of threads per block. This is also in progress.
4. Explain all findings from the above evaluation using concepts discussed in course material or other factors determined from additional research. We want to determine

what kinds of workloads most benefit from GPU parallelization and how well we are able to scale with larger inputs. This is also in progress.

4 Plans for Demo

We plan to demonstrate our achieved speedup for the CUDA implementation compared to the sequential implementation using graphs. Alongside those, we want to evaluate which workloads benefit the most from GPU parallelization, and record how well performance scales with larger graphs (which can be defined by more nodes, higher capacities, and more edges.)

5 Preliminary Results

6 Points of Concern

Our primary point of concern at the moment is that our CUDA implementation is often slower than our sequential implementation. This is obviously not ideal, but we hope to work on this by implementing other techniques described by Wu et al. [2] that are suggested to have better results, as well as play around with how we are handling shared data (to potentially reduce the number of times we copy from host to device and vice versa.) We also need to be able to test on larger test cases to get more meaningful speedup results, especially since the paper we are referring to tests on millions of nodes, and since our CUDA implementation is unable to run those tests we need to change our current approach, which is to use an adjacency matrix to keep track of residual capacities.

References

[1] Jiadong Wu, Zhengyu He, Bo Hong, Chapter 5 - Efficient CUDA Algorithms for the Maximum Network Flow Problem, Editor(s): Wen-mei W. Hwu, In Applications of GPU Computing Series, GPU Computing Gems Jade Edition, Morgan Kaufmann, 2012, Pages 55-66, ISBN 9780123859631, https://doi.org/10.1016/B978-0-12-385963-1.00005-8.

[2] https://vision.cs.uwaterloo.ca/data/maxflow