15-418 Project Milestone

Gary Gao, Mingxuan Li
wgao2@andrew.cmu.edu, mingxua3@andrew.cmu.edu

Updated Website

Website URL: https://garygao33.github.io/ParallelFlow/

Progress

We reviewed the relevant lecture notes and papers about the maximum flow problem and Dinic's algorithm. We also searched for academic papers for parallelized maximum flow algorithms to find a baseline for comparison, but we have only found papers analyzing theoretical asymptotic bounds, which have little value to our analysis in this course. We therefore decide not to have a specific absolute target, but rather analyze the speedups (as well as the imperfection in speedups) of our parallelized algorithm vs. our sequential algorithm.

We implemented a correct and efficient sequential version of Dinic's algorithm in C++ (including the I/O and timing components) that is suitable for use as the sequential baseline. We also implemented testing framework that can generate random input files based on specified number of vertices, the range of edge capacities, and the degree of vertices. This allows us to create basic suites of test inputs of varying graph sizes and graph density.

We also have some initial trials on the parallelized versions:

- We began by experimenting with parallelizations of the blocking flow computation phase. The main idea is to allow for multiple threads to search for augmenting paths in parallel. However, our basic implementations have correctness issues due to contention for edge occupancy updates. We are currently tuning the algorithm and data structure to allow for correct handling of overlapping augmenting paths. We also tested with different granularities of synchronization and atomicity, and the results show that we need relatively fine edge-level granularity on the updates to allow for a good speedup, which demands better designs that can account for the correctness issue.
- We also have several designs on the phase of building leveled residual graphs (not yet implemented). It will mainly be parallelizing the breath-first searches, which should be more intuitive than the phase of finding blocking flows. We may need to focus on this phase, if the solutions for the blocking flow computation phase do not allow for greater speedups.

Goals

We are following our original schedule and on track to complete the basic goals. We anticipate no significant change in our original goals. However, we feel that implementing the parallelization for these two phases would be sufficiently time-consuming and require arduous debugging. Therefore, we are unlikely to do the extra "nice to haves" of implementing with other frameworks and experimenting with other algorithms.

We reiterate the set of goals we plan to achieve here:

- Implement a successful and efficient sequential version of the Dinic's algorithm.
- Implement parallel versions of the algorithm using OpenMP that can achieve reasonable speedup. The parallel versions will utilize parallelizations of one or both phases.
- We will run experiments on GHC and PSC machines to analyze the performance of the best parallel version. The experiments will include inputs of differing characteristics.
  - For the analysis component, our aim is to understand how well the algorithm adapts to parallel execution on a shared-memory system using OpenMP. Specifically, the questions are: how do the two phases of the algorithm, namely level graph construction and blocking flow computation, scale with increasing thread count, and which parts become bottlenecks due to synchronization or load imbalance. We will analyze how graph structure (e.g., density, degree distribution, size, and shape) affects parallel performance and whether certain graph types benefit more from parallelism than others.

Deliverables for Poster Session

For the poster session, we plan to have a poster that contains the following items:

- Background information, as well as illustrations, for the maximum flow algorithm and Dinic's algorithm.
- A detailed explanation and pseudocode of our parallel implementations.
- Our design of test suites.
- Experiment results and analysis of the performance of our parallel algorithm. Specifically, various speedup graphs with respect to the number of threads and graph characteristics (for the sensitivity studies) will be included.

Issues

Currently, the major issue is to correct the implementations that parallelize the blocking flow computation phase. We would need significant debug and tuning time to achieve this. We also need to implement the parallelization for the level graph construction phase, for which we expect less hindrances.

Another minor issue is that the current implementations are very fast on small inputs, making it inaccurate and difficult to compare speedups. We have therefore generated large inputs with millions of edges, but their file size are relatively large (up to several hundred MBs), so we need to manage our cloud storage more carefully.

Updated Schedule

We make an updated version of our schedule here, as required. The letters in parentheses indicate the persons responsible.

- 4/18
  - Implementation of the level-graph construction phase. (G)
  - Correction of coarse synchronization version of parallelized blocking flow computation phase. (M)
- 4/21
  - Optimize the parallelization of the level-graph construction phase. (G)
  - Make an efficient version of the parallelization of the blocking flow computation phase. (M)
- 4/25
  - Complete the building of all test inputs. (G)
  - Tune the parallelized versions with different atomicity and synchronization granularities. (G and M)
- 4/28
  - Finish all experiments and make all graphs. (G and M)
  - complete the poster and the report. (G and M)