Parallel GPU Max Clique Finder

Ryan Lin and Matt Hegi
URL: https://github.com/ryalin/clique

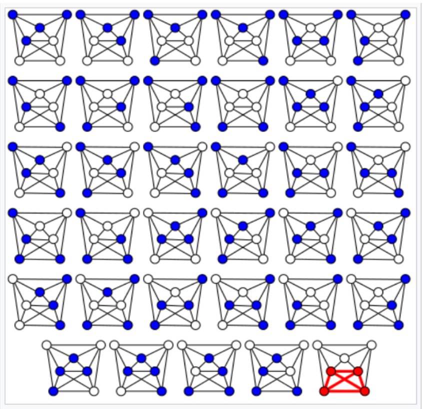
Summary:

In this project, we will implement a parallelized algorithm using OpenMP and CUDA to determine the existence of a clique of size n given an undirected, unweighted graph.

Background:

A clique, or a complete subgraph, is a subset of vertices in a graph such that all the vertices are adjacent to each other. This problem has several real-life applications, such as determining mutual connection groups in a social network.

There are many clique problems, for example, finding the max clique, listing all maximum cliques, finding all cliques greater than a size, etc. We will focus on the clique decision problem, which is NP-Complete. The clique decision problem is to find if a clique of size n exists in the graph or not.


The brute force algorithm finds a 4-clique in this 7-vertex graph (the complement of the 7-vertex path graph) by systematically checking all C(7,4) = 35 4-vertex subgraphs for completeness.

From https://en.wikipedia.org/wiki/Clique_problem

The goal of our project is to accelerate the solution to the problem using CUDA and OpenMP. This will benefit from parallelization because a singular thread or core will have to scan all the node combinations by itself. With multiple threads we can cut down on the search time significantly, either by assigning a thread to each node, breaking the graph into smaller subgraphs for computation, or something else.

Challenges:

Dependencies among nodes:

The clique problem relies on finding subgraphs with nodes all adjacent to each other. A common algorithm is for every new node, check if it is adjacent to all the nodes in the existing clique. This will introduce dependencies that will make it difficult to parallelize.

Workload imbalance:

Since each node has a different number of neighbors, we can't just give each thread a certain number of nodes to check. We don't want one thread to work a lot longer than others. Also managing overhead in assigning tasks will be a challenge.

Memory:

With large subgraphs, there will be a large amount of memory access and storage. We need to determine a way to parallelize the algorithm without mass amounts of intermediate memory. In our CUDA implementation, we will be constrained by device and shared memory, resulting in an even more challenging task of parallelization. One other thing is that since the computation is generally very low (just checking for node adjacency) compared to memory access.

Resources:

We will be using the GPUs in the GHC clusters for our CUDA implementation and the PSC machines for OpenMP (if the resources are available, otherwise GHC also), so we can benefit from the computation power not found on our personal devices.

We will start from scratch to implement the best non-parallel algorithm for this problem, with preliminary research into the Bron-Kerbosch Algorithm (Wikipedia). There may be other algorithms available that will require further research.

Testing is a very important pillar of our project. We will need to figure out how we are going to get the test graphs, either making them ourselves or finding an external source.

Resources used to learn about the problem:
https://en.wikipedia.org/wiki/Clique_problem
https://theory.stanford.edu/~virgi/combclique-ipl-g.pdf

Goals and Deliverables:

If our project goes slower than expected:
We at least expect to implement a sequential clique finding algorithm and also an OpenMP version with noticeable speedup.

Plan to achieve:
Parallelizing the algorithm using OpenMP and CUDA, achieving at least ½ * core_count speedup. Development of a simple testing suite.

Hope to achieve:
Parallelizing the algorithm like above, with at least 0.75 * core_count speedup, development of a comprehensive testing suite for all edge cases. Try MPI if time permits.

Demo and hope to learn:
For our demo, we plan to describe the speedup(s) obtained through speed-up graphs as well as the steps we took to obtain it. We want to measure speedup for different numbers of threads, implementations, and different values of n (size of clique). Also want to find answers to our challenges above.

Platform Choice:

We will use the GHC and PSC (if applicable) machines because of their relatively available uptime and computational power. With this we plan to use C++ (with OpenMP) and CUDA to parallelize our algorithm due to our prior experiences working with them.

Schedule:

Dates	To-Do
Week 1 11/11 - 11/17	- Research problem and solutions - Work on project proposal
Week 2 11/18 - 11/24	- Start working on sequential algorithm and start on parallelization with OpenMP - Write test cases
Week 3 11/25 - 12/1	- Finish week 2 tasks and start parallelization with CUDA - Start working on milestone report
Week 4 12/2 - 12/8	- Finish all parallelization, wrap up and benchmark solutions - Fix any extraneous issues - If time permits, improving parallelization and MPI
Week 5 12/9 - 12/15	- Write project report - Make poster and continue benchmarking