TITLE:
The Effectiveness of Multilevel Parallelism on Minimum Spanning Forest and Maximum Independent Set Problems
Connor Clayton and Jacqui Fashimpaur

URL:
http://www.andrew.cmu.edu/user/cbclayto/418%20Final%20Project

SUMMARY:
We are going to implement parallel algorithms for two graph-processing problems, inspired by a new BFS strategy we read about that uses MPI. We will compare this strategy's effectiveness on the different problems.

BACKGROUND:
The two problems we will be implementing solutions for are Minimum Spanning Forest and Maximum Independent Set. The strategy from the paper roughly involves splitting a tree into parts, processing the parts separately, and then putting them back together.

Minimum Spanning Forest asks of a weighted undirected graph: for every connected component in the graph, what is the minimum cost tree that spans every vertex in the component? One common algorithm for solving this problem is Kruskal's algorithm, which makes a list of all edges in the graph in order from lightest to heaviest and iterates through them, adding those that do not create cycles. Though this algorithm is fairly sequential, the problem could benefit from parallelism because (as long as the graph is relatively sparse) different sections of the graph could be processed at the same time, without having major impacts on each other.

Maximum Independent Set asks of an unweighted undirected graph: what is the largest set of vertices in the graph such that none of the vertices in the set are adjacent to any other vertices in the set? This problem is NP-hard, but has algorithms that are faster than the naive brute force method. It would be interesting to see if parallelising the naive brute force algorithm (having each thread try its own subsets) results in something faster than the strategy in the paper (which involves splitting up the graph and processing it in parts).

THE CHALLENGE:
For both problems, there can be a lot of dependencies if the graph is dense. It would be difficult to find the best way to split the graph up into parts such that there are as few edges between the parts as possible. The edges between parts will be very important, since they will require the parts to communicate while they are processing independently--it could be that one of those edges belongs in the full MST, or that one is an edge connecting two vertices that are both supposed to be in the Independent Set. Since the information related to each vertex is minimal, there will be a high communication to computation ratio.

RESOURCES:
We plan to start from scratch (in that we will not use any starter code), but we plan to use two references throughout the project. The first is the Problem Based Benchmark Suite (http://www.cs.cmu.edu/~pbbs/), which provides a C++ implementation of parallel minimum spanning forest (MSF) and maximal independent set (MIS) solvers. We plan to use this as an algorithmic reference (i.e. Kruskal's algorithm) and not an implementation reference, since it uses a different style of parallelism (CILK) than we will use. We also plan to use the serial versions of these solvers provided by the reference for correctness checking.

Our second reference is a paper, cited at the end of this section. The paper describes multi-level parallelization techniques for Breadth-First Search and calculating Betweenness Centrality, and also provides code that implements them. We will use this source as a reference for these techniques, which we will apply toward the MSF and MIS problems.

One interesting aspect explored by the paper is parallelization across multiple GPUs. If there is a machine that has this resource, this is something we would like to explore. As a backup, we plan to test our code on the GHC machines we've been using so far in this class.

M. Bernaschi, M. Bisson, E. Mastrostefano and F. Vella, "Multilevel Parallelism for the Exploration of Large-Scale Graphs," in *IEEE Transactions on Multi-Scale Computing Systems*, vol. 4, no. 3, pp. 204-216, 1 July-Sept. 2018.

GOALS AND DELIVERABLES:
Plan to Achieve

- We plan to complete a correct parallel implementation for each of the two problems mentioned above, inspired by the Bernaschi paper.
    - This can hopefully teach us more about MPI and when/how it is most effective
    - We don't have specific performance goals, because this strategy may not be particularly effective on these problems. We are more focused on exploring the benefits and drawbacks of this strategy.
- We will create speedup graphs demonstrating the effectiveness of our solutions relative to other solutions, relative to other methods we attempted, and relative to each other.
- We will complete a more qualitative analysis of what works and does not work well for the respective problems, and why.
- If we fall behind, we will only complete one problem and discuss how effective the new strategy was on it.

Hope to Achieve

- If we have time, we'd like to implement a parallel implementation for one or both of the above problems that's not related to the Bernashi paper, like one using OpenMP.
- If we do this, we will also create speedup graphs/analyses comparing these solutions to our others.

PLATFORM CHOICE:
We'll be working in C++ using Open MPI and CUDA. Ideally, to resemble the environment of the paper we are using for reference, we want to run our code on a machine with multiple GPUs. Having multiple GPUs will benefit us because it will allow us to partition the graph and run computations on each section in parallel on GPUs.

SCHEDULE:

Week	Dates	Plans
1	10/31-11/05	Begin project; figure out high level ideas for MSF: graph representation, graph partitioning, algorithm choice, CUDA parallelism strategy
2	11/06-11/12	Write the bulk of the code for MSF; this will be the most intensive coding week and likely the most difficult part of the project
3	11/13-11/19	Finish up MSF code: make optimizations, etc.; write checkpoint report; meet with course staff for checkpoint (11/18); study for exam (11/20)
4	11/20-11/26	Begin and complete bulk of work on MIS; this should be less difficult after having done MSF; Connor will be out of town 11/21-11/23
5	11/27-12/03	Thanksgiving break; don't plan on getting too much work done during this time
	12/01-12/03	Finish up MIS code: make optimizations, etc.
6	12/04-12/07	Code should be complete; perform analysis of programs; compile final report; create poster; Jacqui will be super busy with play on December 6 and 7
	12/08-12/09	Get poster printed; make final touches to report; report due 12/09 at 11:59pm