Final Project Proposal

fmautner, emuchnik @ andrew.cmu.edu

April 8 2024

1 Title

Parallel Route Planning With Time-Dependent Graphs

2 URL

Project website or `https://fzmautner.github.io/418-FinalProject/`

3 Summary

In the realm of traffic routing (in a way similar to Google Maps) there isn't currently an open sourced implementation that takes advantage of parallel computing to establish routing for multiple vehicles while taking traffic and road limitations into account, especially with new cars entering the road network.

4 Background

While parallel algorithms are well studied for static graphs, there is far less literature concerning dynamic graphs. This is in part due to the inherent challenge of implementing a parallel solution in a problem where dependencies span not only space/ branches of execution but also time. As such, the research in this topic is limited and algorithms are scarce, although we must highlight the existence of adaptations of Dijkstra's and A* pathfinding algorithms for time-dependent graphs. However, these are not explicitly parallel, only find a single route, and do not account for any changes to the graph due to additional cars (meaning traffic).

Below are some of the papers we will be using to guide in the first step of making a time-dependent implementation of Dijkstras and A*.

https://i11www.iti.kit.edu/extra/publications/dw-tdrp-09.pdf
https://link.springer.com/article/10.1007/s41019-019-00105-0

5 The Challenge

Due to the complicated dependencies of dynamic graphs which are not only spatial but also temporary, we expect synchronization to be challenging to achieve. As such, we foresee that our main challenges with achieving good parallel performance will be with maintaining data structures and procedures that can run as independently as possible while maintaining acceptable quality results. On top of that, figuring out a parallel structure to account for additional traffic with the introduction of new cars while not expending too much extra computation running our algorithms will prove to be a challenge. Thus, we expect that there will be some necessary tradeoffs between performace and quality of results, a situation similar to that seen in HW3 and HW4, and common throughout this area of study.

6 Resources

We intend to write code that can be run in single-node multi-core machines, and for that we would want to use the GHC machines and their GPU's. In addition, we believe that certain calculations can in large part be parallelized to the extent that PSC's 128 core machines may have use.

We believe this wide range of machines will give us insight into which algorithms and approaches for parallelization are best.

7 Goals and Deliverables

7.1 Plan to achieve

Our goal is to successfully implement a parallel path assignment algorithm that can route multiple vehicles though a network while minimizing the time spent by each vehicle in each route. This would initially be done on a small graph representing a small map of a town with a simple road network.

7.2 Hope to achieve

Time allowing it, we would like to explore and increase the scalability of our solution, possibly using new frameworks such as MPI to perform distributed parallel work on a large graph. Another goal we hope to achieve is to be able to read real world maps and compare our results to those observed in real life and our routes with those recommended by popular routing apps such as Google Maps, Waze, Apple Maps and etc.

7.3 Poster session/ demos

We plan on displaying diagrams that illustrate how standard time dependent algorithms for graphs work, as well as how we've parallelized them. We also want to exhibit charts and graphs of our analysis and performance debugging process, highlighting non-trivial, key insights that made the final product possible.

8 Platform Choice

MPI is likely the best choice - at least for the initial implementation only because some parts of Dijikstras may not be parallelized and will be run on one thread while other completely separate parts of the algorithm and approaches will be run on other threads. The amount of control MPI has over what code each thread executes will prove useful in the initial implementation.

Eventually, as Dijkstra itself may be parallelized frameworks like CUDA/OpenMP as the underlying algorithms get parallelized themselves.

9 Schedule

- Week 2 (April 10^{th}):
  - Implement simple graph representation and parsing.
  - Implement sequential Dijkstra's for a time dependent graph.
- Week 3 (April 17^{th} - April 16 Milestone checkpoint):
  - Implement parallel version of Dijkstra's for a time dependent graph using MPI by implementing various parts of our approach and not necessarily each individual algorithm via MPI.
  - Start testing and performance debugging.
- Week 4 (April 24^{th}):
  - Continue to refine parallel solution by parallelizing the underlying algorithms which will enable testing on the PSC.
  - Measure and analyze solution in different platforms such as OpenMP and CUDA, continuing to improve where needed
  - Start writing the final write-up and poster
- Week 5-6 (March 5^{th}):
  - Attempt to integrate our algorithm directly with google maps data and testing the scalability of our different approaches
  - Finalize write-up and poster, present