Parallel Linear Programming Solver Using MPI

Yuan(Alex) Guan and Yixuan(Ivan) Zheng
https://github.com/IvanLenn/15-418-Project

March 28, 2024

1 Summary

Summarize your project in no more than 2-3 sentences. Describe what you plan to do and what parallel systems you will be working with. Example one-liners include (you should add a bit more detail):

- We are going to implement a parallel version of a parallel linear programming solver with MPI. We will parallelize Simplex algorithm in order to speed up the computation of linear programming solver. We plan to experiment on the GHC Clusters and PSC machines and perform analysis on the results. We'll probably study the parallelization of other LP algorithms such as interior point algorithms and Seidal's algorithm.

2 Background

If your project involves accelerating a compute-intensive application, describe the application or piece of the application you are going to implement in more detail. This description need only be a few paragraphs. It might be helpful to include a block diagram or pseudocode of the basic idea. An important detail is what aspects of the problem might benefit from parallelism and why?

- Linear programming(LP) is a technique for the optimization of a linear objective function. The development of better and faster LP algorithms is an interesting ongoing area of research. In standard form, an LP can be written as:

  $$\text{For a variable vector x: maximize } c^T x$$

  $$\text{subject to } Ax \leq b$$

  $$x \geq 0$$

  The Simplex algorithm is a common method to solve LP. Starting with a corner of the feasible region, we repeatedly look at all neighboring corners of our current position and go to the best one. When we reach a position without better neighboring corners, it means we have found the optimal solution because the feasible region is convex. This algorithm can be greatly customized by large numbers of parameters and tolerances. We wish to speed up the algorithm using parallel techniques with message-passing.

3 The Challenge

Describe why the problem is challenging. What aspects of the problem might make it difficult to parallelize? In other words, what to you hope to learn by doing the project?

- Data Partitioning: Divide the constraints and objective function coefficients among MPI processes. Use data distribution techniques to distribute the data in a balanced way and minimize communication overhead. Locality can be a problem for LP algorithms dependent on the input constraints.

- Load Balancing: Implement load balancing work assignment methods to distribute the computational workload among MPI processes. This may involve dynamically redistributing data or adjusting the workload of each process based on its progress during the algorithm's execution. We'll experiment with different levels of dynamic assignment from static to fine-grained.
- Communication Optimization: Minimize communication overhead by aggregating data exchanges between processes using techniques such as batches and reducing the frequency of artifactual communication where possible.
- Parallelization Strategy: Explore different parallelization strategies, such as parallelizing different stages of the Simplex algorithm or parallelizing multiple independent Simplex iterations concurrently. We'll probably study the parallelization of other LP algorithms such as interior point algorithms and Seidal's algorithm.

4 Resources

Describe the resources (type of computers, starter code, etc.) you will use. What code base will you start from? Are you starting from scratch or using an existing piece of code? Is there a book or paper that you are using as a reference (if so, provide a citation)? Are there any other resources you need, but haven't figured out how to obtain yet? Could you benefit from access to any special machines?

- Type of computers: GHC for ≤ 8 cores. PSC for multi-cores experiments.
- We will probably use some existing code for the serial version implementation.
- We are going to implement the parallel version with MPI.

5 Goals and Deliverables

Describe the deliverables or goals of your project. This is by far the most important section of the proposal!

- Basic Goal:
  A working serial version of the Simplex algorithm, a benchmark script to measure the performance, as well as a working serial version of the Simplex algorithm using MPI.
- Ideal Goal:
  Optimize our parallel version by using proper parallel strategy and techniques after experiments and analysis. We wish to achieve a 10x speedup on PSC machines with high thread counts. (Past research using the shared memory approach has reached 19x speedup)
- Extra Goal:
  If we've achieved the ideal goal, we'll try to further optimize our parallel version of the Simplex algorithm if possible. We'll probably utilize efficient (perhaps parallel lock-free) data structures to help us speed up the program as well.

6 Platform Choice

Describe why the platform (computer and/or language) you have chosen is a good one for your needs. Why does it make sense to use this parallel system for the workload you have chosen?

- We'll use GHC machines for 8-core CPU testing for experiments for our basic implementation.
- We'll also use PSC machines for experiments with multi-nodes.
- We'll C++ for this task, which is also appropriate since C++ has excellent support for MPI interface for parallel programming with MPI.

7 Schedule

Produce a schedule for your project. Your schedule should have at least one item to do per week. List what you plan to get done each week from now until the parallelism competition in order to meet your project goals. Keep in mind that due to other classes, you'll have more time to work some weeks than others (work that into the schedule). You will need to re-evaluate your progress at the end of each week and update this schedule accordingly. Note the intermediate milestone deadline is April 16th. In your schedule we encourage you to be precise as precise as possible. It's often helpful to work backward in time from your deliverables and goals, writing down all the little things you'll need to do (establish the dependencies).

Time	Plan
-3/27	Brainstorming Project Ideas and Proposal
3/28-3/30	Finalize Project Idea
3/31-4/12	Implement both serial and parallel version, do various experiments
4/13-4/14	Milestone Report
4/15-4/24	Compare different approaches of parallelism and do scalability analysis
4/25-5/5	Extra optimization, Final report