Heterogeneous Distributed Barnes-Hut Scheduler:

Project Milestone Report

Zhuoyi Zou (zhuoyiz), Vanessa Lam (yatheil)

WEBSITE URL

https://cchewies.github.io/15418-project.html

SUMMARY

So far, we have implemented four versions of the N-body (galaxy) simulation:

- Serial N^2 naive running on a single thread
- Serial NlogN Barnes-Hut running on a single thread
- MPI single-node NlogN Barnes-Hut running on multiple processors on a single campus machine
- MiniMPI (MMPI) NlogN Barnes-Hut running on multiple campus machines

Since MPI-native options do not work with the password login when attempting to ssh into multiple campus machines, we implemented MMPI, a tailored subset of MPI built on top of TCP. We also have a dynamic load balancer developed, using the calculation runtimes of each node as benchmarks for assigning an optimal amount of work to each node, which also uses Morton ordering to ensure each node gets a spatially-local set of stars.

GOALS AND DELIVERABLES

Plan to achieve:

- ☑ ~~Implement Barnes-Hut (with C++ starter code)~~
- ☑ ~~Enable rank-level parallelism with MPI (single-node + multi-rank + static assignment)~~
- ☑ ~~Build a dynamic load balancer (single-node + multi-rank + dynamic assignment)~~
- ☐ Tuning & performance optimization on single node

- [x] ~~Expand MPI implementation to support node-level parallelism (multi-node + multi-rank + dynamic assignment)~~
- [x] ~~Expand dynamic load balancer to consider node heterogeneity (GHC, ECE, linux.andrew, Hamerschlag) (multi-node + multi-rank + improved dynamic assignment)~~
- [ ] Benchmark performance with different combinations of heterogenous compute

Hope to achieve:

- [ ] Scale to more machines
- [ ] Compare different distributions of machines (ex. 7 GHC vs one of each)

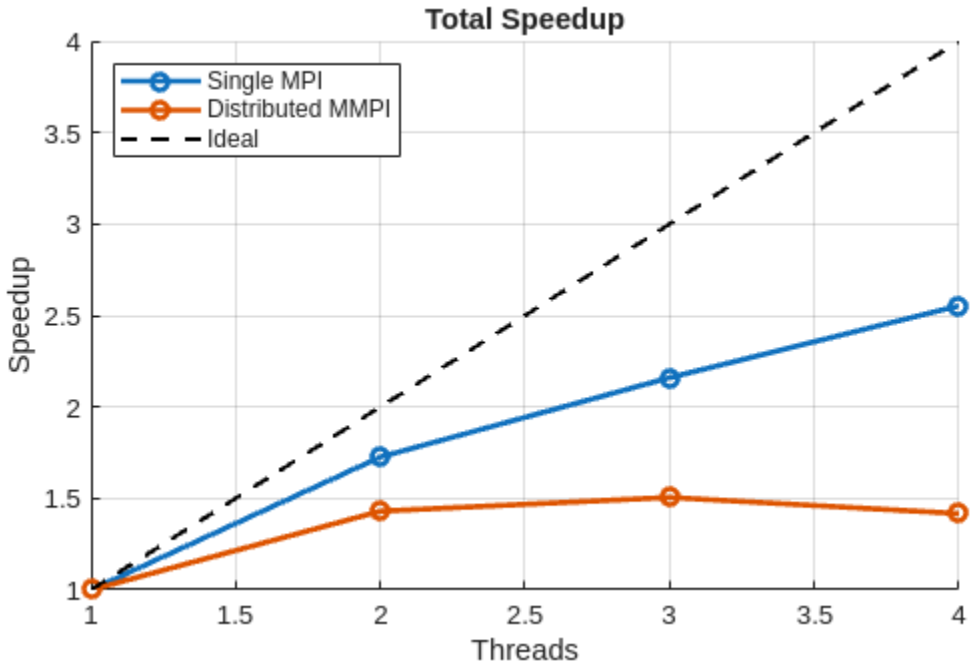
We are ahead of schedule with respect to the original plan. As of today (April 13th), we are starting the "redesign" step of the initial plan, which was scheduled for the latter half of this week.

We also believe that we can achieve both of our "nice to have" goals, since they are primarily focused on benchmarking a complete implementation rather than expanding the implementation itself. Our code is designed to be configurable to various different execution environments.

We did encounter some difficulties with setting up our code to run on the ECE and Hamerschlag machines, as they do not have MPI or SDL2 installed. Compiling separate versions of the code was a sufficient fix, however.

At the poster session, we plan on demoing our code running on some number of different campus machines.

PRELIMINARY RESULTS



Barnes-Hut is a high-communication problem, so we do not expect linear speedup from either the single-node MPI implementation or the distributed MMPI implementation. Both implementations ran a simulation of 50,000 stars on GHC machines only. A serial implementation takes around 100ms per iteration.

Despite the subpar interconnect between separate GHC machines for the distributed implementation, we were surprised to see that it still had potential for speedup, reaching nearly 1.5x on 2 nodes (though single-node MPI reaches 1.7x). Although adding more nodes beyond 2 does not improve speedup, we believe that this is due to our currently naive implementation of allgather, which sends all data to a single node and broadcasts back to everyone else.

CONCERNS

As discussed during our proposal meeting, the interconnect between different campus machines (especially between GHC and linux.andrew) is fairly slow, which is also noticeable in our preliminary results. However, we believe that it should still be fast enough to permit some speedup, though we do not expect it to meet the speedup of a single-node MPI implementation.

SCHEDULE

Week of 4/13, first half: Improved allgather, benchmarked dynamic load balancing
Week of 4/13, second half: General performance optimizations (message size, code structure, tweak parameters)
Week of 4/20: Run final benchmarks, write report, make poster