Heterogeneous Distributed Barnes-Hut Scheduler

Zhuoyi Zou (zhuoyiz), Vanessa Lam (yatheil)

WEBSITE URL

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

SUMMARY

We plan on implementing a dynamic load balancer for a distributed Barnes-Hut simulation running on a heterogeneous computing cluster. The load balancer and simulation will be built using MPI, and we will also implement a version to run across various campus computers.

BACKGROUND

Barnes-Hut is an N-body simulation algorithm that computes the net force on each body by summing pairwise interactions (e.g., gravitational or electrostatic forces) from all other bodies, using their masses, positions, and distances to determine the resulting acceleration and motion in each iteration.

Parallelized workloads generally have one of three work assignment techniques: static, semi-static, or dynamic. Semi-static is particularly well-suited for Barnes-Hut, since workloads for each body are cheap to compute and the spatial distribution of the bodies changes slowly. A well-balanced work distribution is likely to remain well-balanced for several iterations, reducing the amount of synchronization required for assignment.

Existing dynamic scheduler implementations, such as CUDA Dynamic Parallelism (CDP) and MPI-based distributed schedulers, enable runtime adaptation of work allocation*. While CUDA primarily focuses on intra-device parallelism, multi-GPU or multi-node execution requires additional host-side coordination, as device-side kernels cannot directly manage work across nodes; concurrency is limited to the scope of a single device, and memory coherence is guaranteed only within the device's global memory. In contrast, MPI-based dynamic schedulers, such as the one proposed in the paper, explicitly target multi-node

environments by assigning tasks with the goal of minimizing overall completion time based on predicted convergence and scaling efficiency. The scheduler dynamically adjusts the number of workers assigned to each job using heuristics driven by marginal performance gains, enabling adaptive resource distribution at runtime. However, this approach assumes homogeneous worker performance, modeling training speed primarily as a function of worker count rather than node-specific characteristics. As a result, it improves utilization and scalability in distributed settings but remains limited in its ability to optimally balance workloads in heterogeneous clusters.

*References: CUDA, MPI Scheduler for Distributed Training

This project aims to explore the possibility of achieving effective scaling beyond a single machine to perform larger simulations and improve throughput, as well as accurately measuring the heterogeneity of machines to improve resource utilization.

CHALLENGE

Both the amount of work per body and the communication patterns in the Barnes-Hut algorithm are non-uniform. Additionally, the communication is fine-grained, which is an especially inefficient pattern for distributed computing. Although Barnes-Hut has some amount of locality, body movements eventually require a redistribution of work assignments.

Campus computers are also highly non-uniform in terms of computing power. The ECE computing cluster runs on datacenter-level GPUs (Tesla T4) while linux.andrew is virtualized and has no GPU access. Another factor contributing to the heterogeneity is the different real-time user activity, hence varying utilization of the cores.

RESOURCES

We will start our project based on the distributed Barnes-Hut starter code at https://arxiv.org/abs/2203.08966)

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
- Expand MPI implementation to support node-level parallelism (multi-node + multi-rank + dynamic assignment)
- 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)

The system would be capable of fairly distributing work among a heterogeneous cluster of computers while dynamically measuring each one's performance.

Performance Goal: Given that the network between the campus machines (even between GHC machines) is very slow, our first target would be to achieve any speedup above 1.0x when running across multiple machines. Ideally, our final goal would be to get as close to linear speedup as possible. If one ECE machine is roughly as powerful as two GHC machines, we should expect a 3x speedup of ECE + GHC over GHC alone.

PLATFORM CHOICE

We plan on implementing this project using MPI on various campus computers. MPI provides a clean interface for specific communication protocols we will use to communicate between the computers. We will write an implementation for MPI that targets distributed communication between campus computers.

Our choice of computing environment is intentionally suboptimal, as our focus will be on developing a load balancing strategy that efficiently makes use of heterogeneous computers physically located fairly far from each other.

SCHEDULE

Week of March 23: Project proposal, read paper
Week of March 30: Initial Barnes-Hut MPI implementation based on paper
Week of April 6: Distributed MPI implementation
Week of April 13: Benchmark, identify bottlenecks, redesign
Week of April 20: Final Report
April 24th: Poster presentation during last lecture slot