Leveraging Parallelism in Swarming and Synchronization Models
Final Presentation

by Benton Liang, Michael Yue, and Huopu Zhang

May 2020

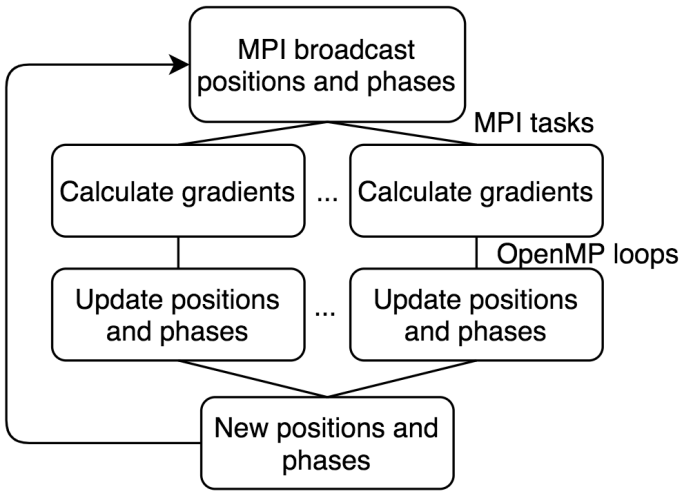

Exact Pairwise Simulation

O'Keefe Model

$$\dot{\mathbf{x}}_i = \frac{1}{N}\sum_{j\neq i}^{N}\left[\frac{\mathbf{x}_j - \mathbf{x}_i}{|\mathbf{x}_j - \mathbf{x}_i|}\left(1 + J\cos(\theta_j - \theta_i)\right) - \frac{\mathbf{x}_j - \mathbf{x}_i}{|\mathbf{x}_j - \mathbf{x}_i|^2}\right]$$

$$\dot{\theta}_i = \omega_i + \frac{K}{N}\sum_{j\neq i}^{N}\frac{\sin(\theta_j - \theta_i)}{|\mathbf{x}_j - \mathbf{x}_i|}$$

- $O(n^2)$ updates computed via OpenMP parallelized for-loops within communicating MPI processors that divide up the points equally
- Very fine-grained parallelism that ideally minimizes synchronization overhead, with majority of overhead coming from communication of updated positions and phases among MPI tasks



The Barnes-Hut Algorithm

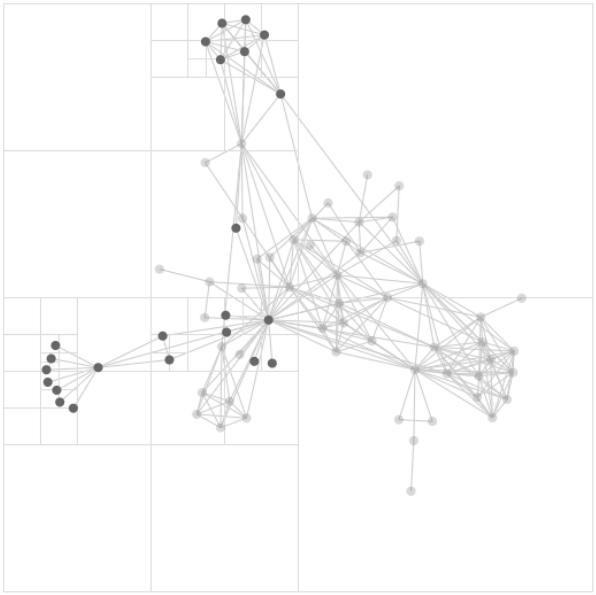


Figure: 1.Quadtree

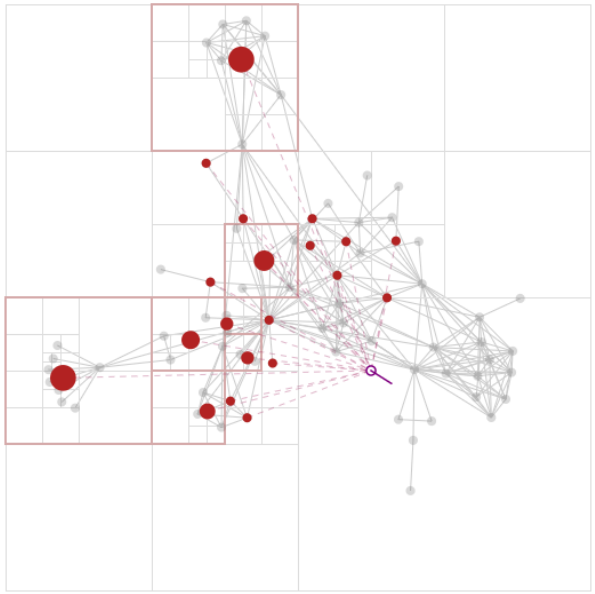


Figure: 2.θ Threshold

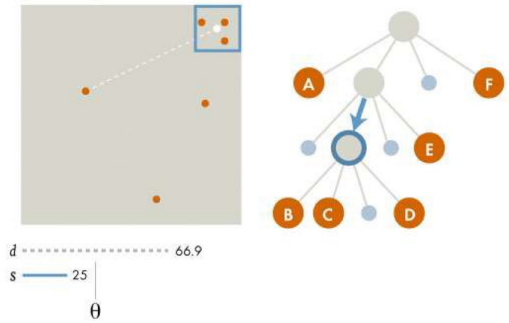


Figure: 3.Traversing

The Barnes-Hut Algorithm is often used in simulating N-body systems. It is based on the assumption that the force from many far-away bodies can be approximated with a single object located at the center of mass and carries the total mass of those bodies.

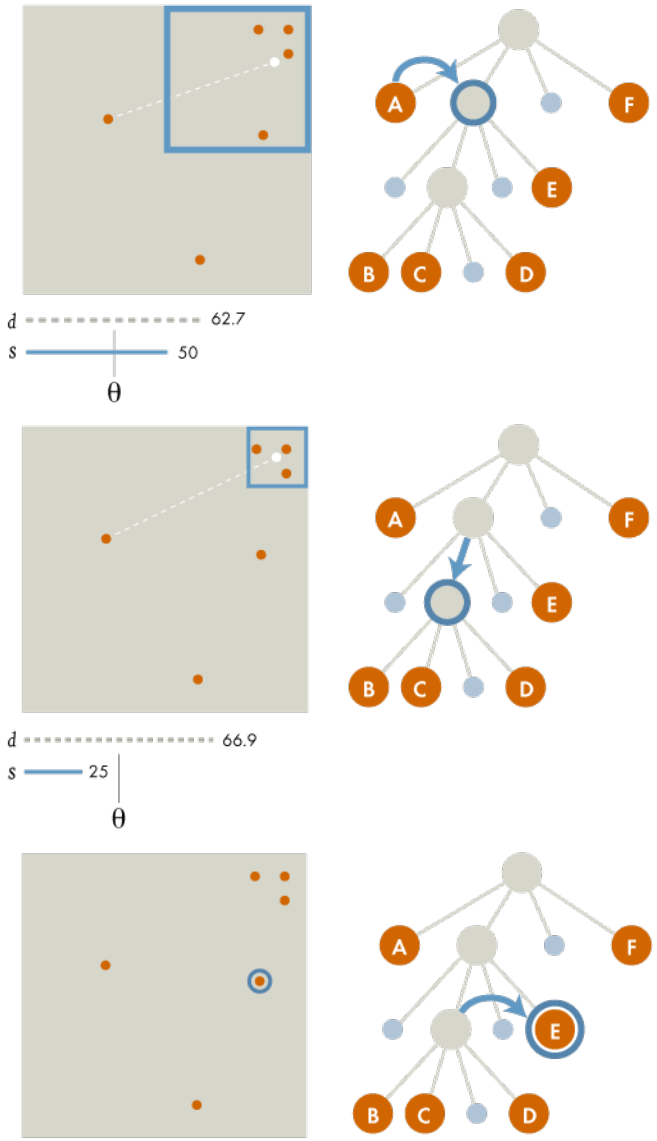
- Quadtree Data Structure - In quadtree, each internal node has 4 children. As shown in the first figure, when inserting particles into the tree, quadrants are recursively subdivided into 4 nodes at each level, until each leaf in the tree contains at most one particle.
- θ Threshold and Traversal - This process is carried out by traversing the tree and evaluating the distance parameter against a threshold of choice.

Traversing the Quadtree

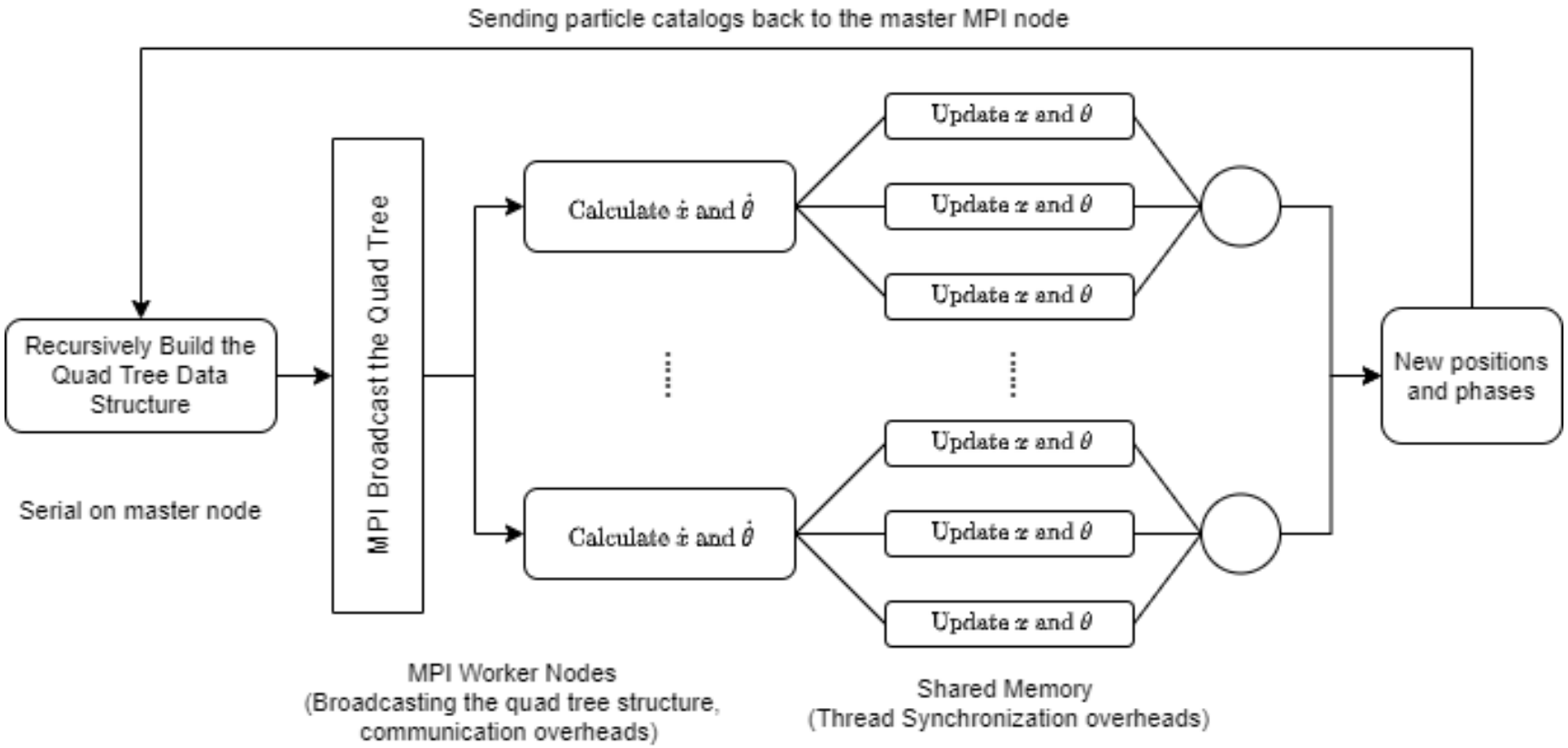
To calculate the overall time derivatives of spatial position and phase of particle b, use the following recursive procedure, starting with the root of the quad-tree:

1. If the current node is an external node (and it is not particle b), calculate the influence from the current node on b, and add this amount to b's net force.
2. Otherwise, calculate the ratio s/d. If $s/d < \theta$, treat this internal node as a single body, and calculate its influence on particle b, and add this amount to b's net force.
3. Otherwise, run the procedure recursively on each of the current node's children.

In the above computation, we use the same time derivative formulae as in the naive algorithm. Overall, the Barnes-Hut algorithm has a time complexity of $O(n \log n)$ as both the build tree and traversal process (n particles, n traversals) has a time complexity of $O(n \log n)$.



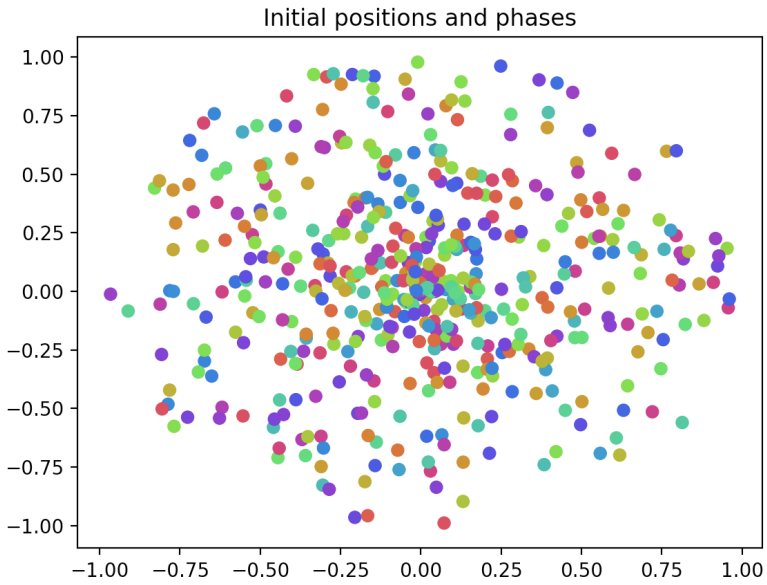
Barnes-Hut Parallelization



- The program consists of 3 parts: 1. The recursive algorithm for tree construction is implemented sequentially. 2. We broadcast the tree data structure with MPI and parallelize the for-loop for computing $\dot{x}$ and $\dot{\theta}$. 3. Since updating the particles does not need all the data in the tree, we further parallelize this for-loop using OpenMP.
- Main computation overheads include the communication cost in broadcasting the tree to the worker nodes and sending back the new positions and phases to the master node. The synchronization of threads in the shared memory part also has overheads.

Implementation Details

- Used the `odeint` package from Boost with default fourth-order Runge-Kutta integration, and consequently Boost-MPI packages needed to be used rather than MPI
- All swarms are initialized with random phases and positions within the unit desk with natural frequencies $\omega_i = 0.1$ for 50 seconds with a timestep of 0.1 second
- Our Big Compute solution: OpenMP and MPI hybrid model on AWS MPI cluster of 2 t2.2xlarge EC2 instances
- Compiler-optimized code with -Ofast flags



runtime (s)	-O0	-O1	-O2	-O3	-Ofast
naive, 1 thread	25.323304	13.306423	12.608864	12.749575	12.008113
naive, 8 threads	4.029078	1.864422	1.780282	1.782941	1.705384
naive, 4 tasks	13.92434	8.369722	8.126015	7.982621	7.763066
Barnes-Hut, 1 thread	159.39276	17.363245	17.149313	16.11532	15.458164
Barnes-Hut, 8 threads	24.266724	3.412126	3.36841	3.204692	3.192476

Model Correctness

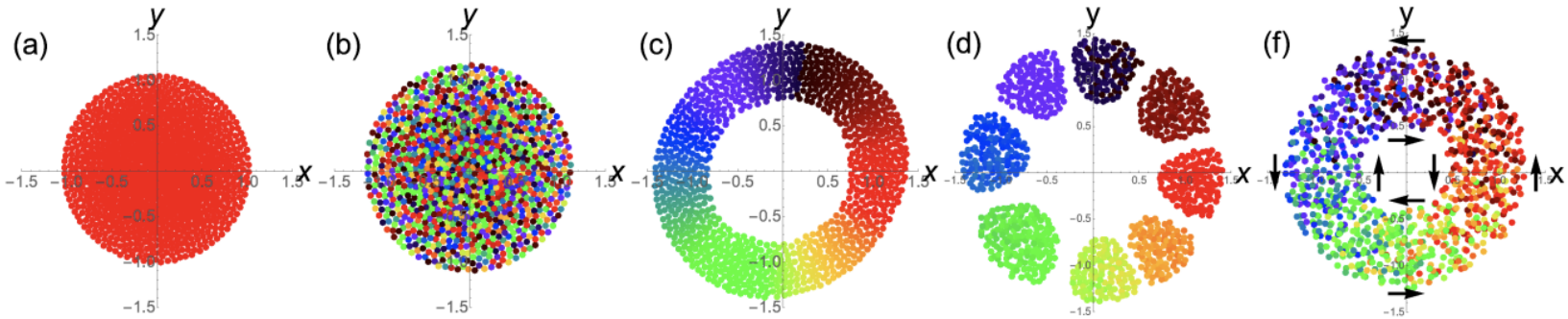
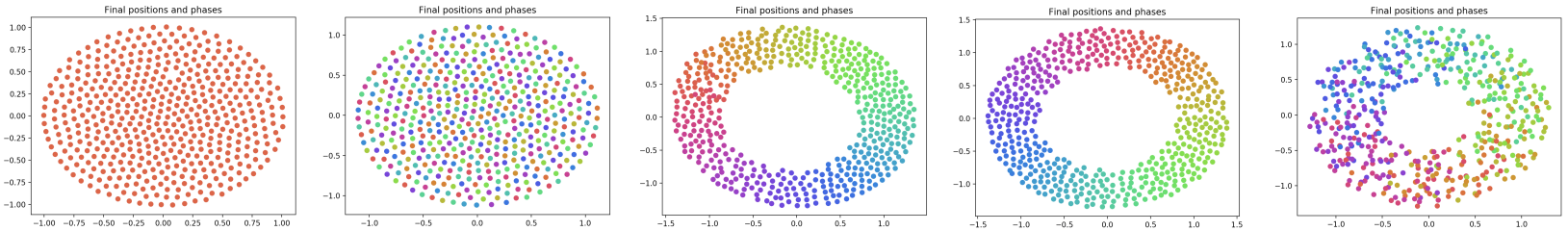
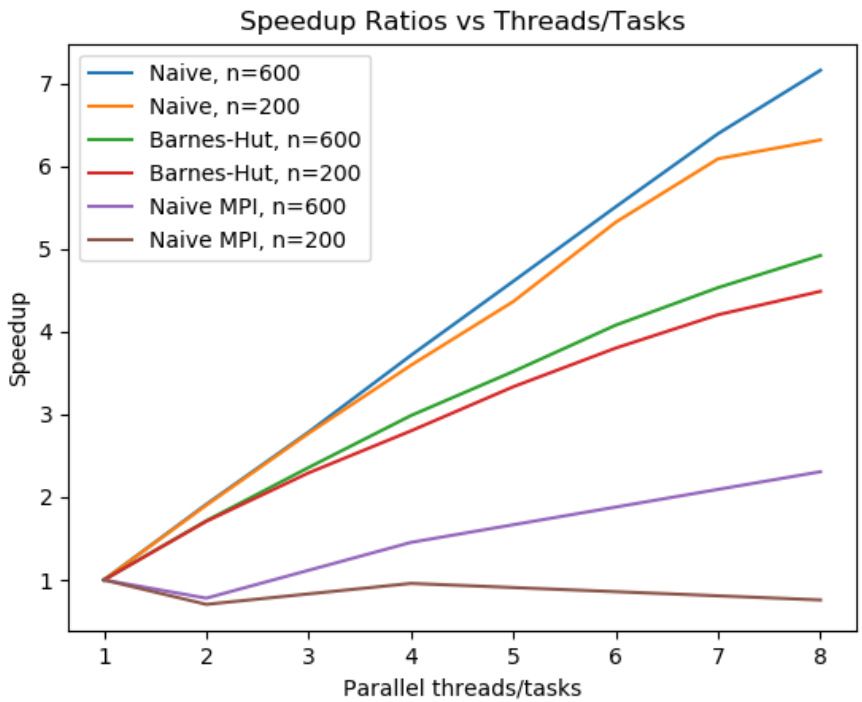


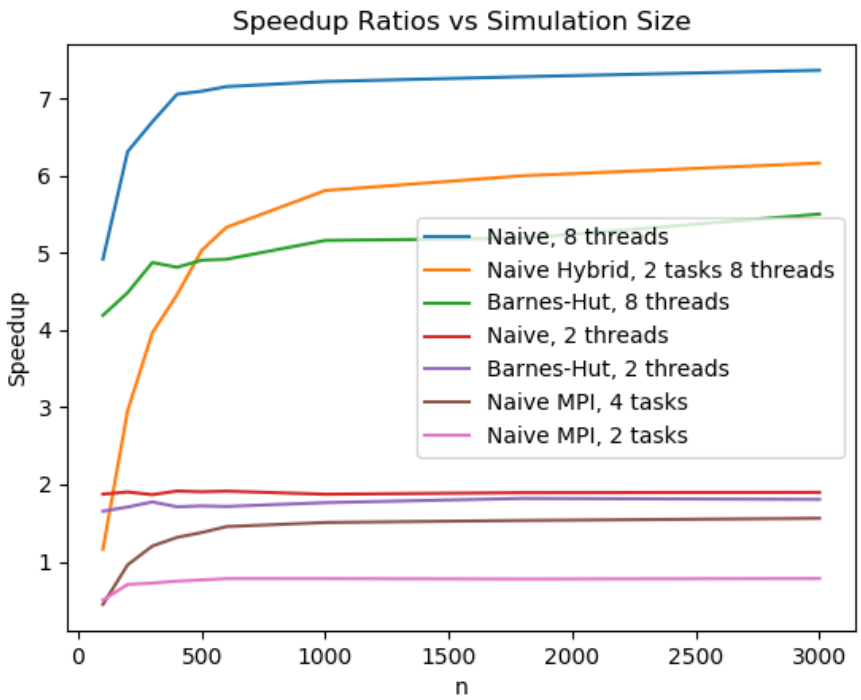
FIG. 1: **Swarmalator states**. Scatter plots in the (x, y) plane, where the swarmalators are colored according to their phase. (a) Static sync for $(J, K) = (0.1, 1)$. (b) Static async $(J, K) = (0.1, -1)$. (c) Static phase wave $(J, K) = (1, 0)$. (d) Splintered phase wave $(J, K) = (1, -0.1)$. (e) Active phase wave $(J, K) = (1, -0.75)$.



- Fineness of (d) result limited by `odeint` integrator, but overall correctness of behavior preserved with OpenMP and MPI

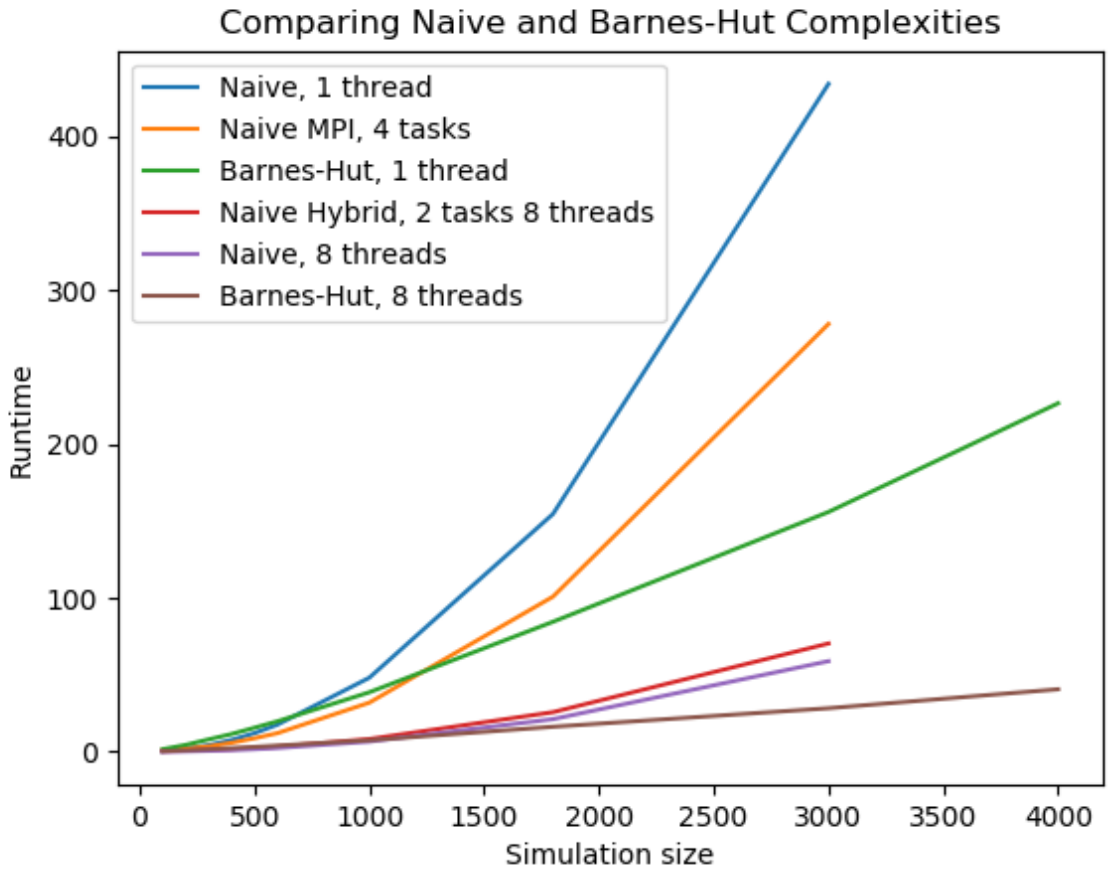
Parallelization Speedup





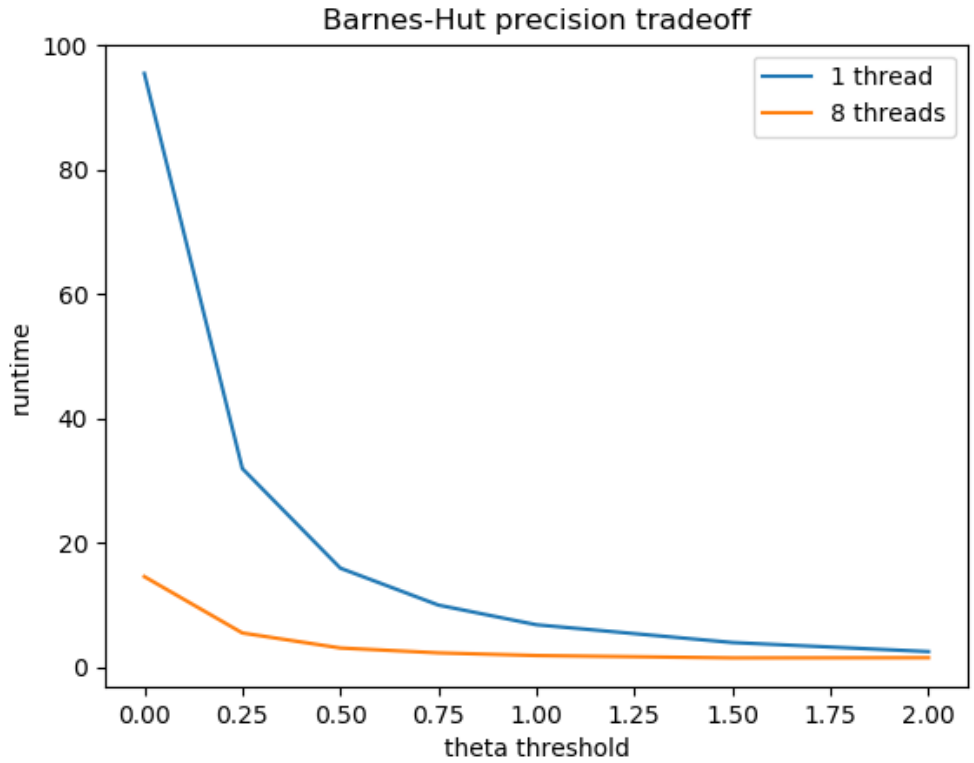
- Better speedup found for higher problem sizes, approaching theoretical maximum
- Less of naive algorithm is serial work, so speedup is better, though overhead of MPI communication severely limits speedup

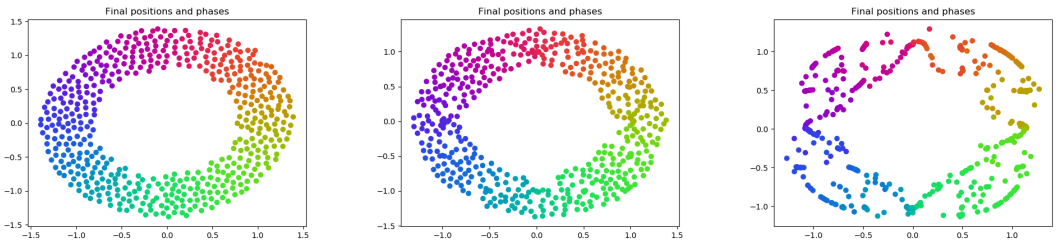
Strong Scaling and Algorithmic Complexity



- Tradeoff between efficiency ($O(n^2)$ vs. $O(n \log n)$) and accuracy (exact vs. approximation)

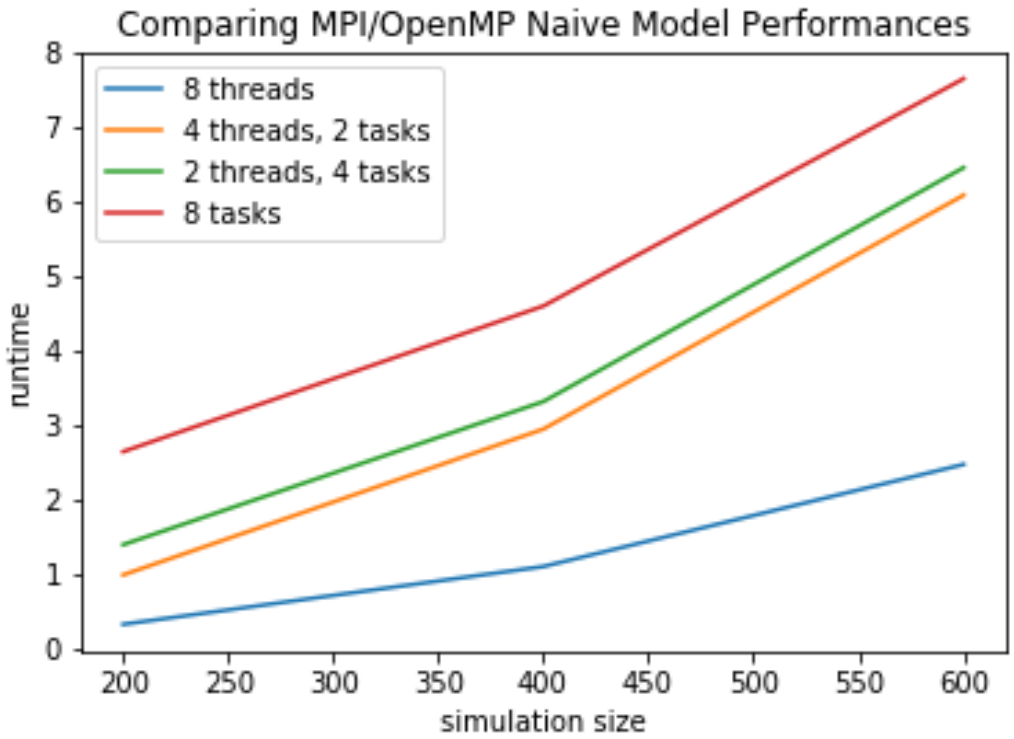
Evaluating the Barnes-Hut Approximation





Showing simulations of $\theta = 0, 1.5, 2$.

Evaluating the MPI/OpenMP Hybrid Model



- Splitting into MPI tasks is worse than using OpenMP threads since each processor must receive all data from all other processors each iteration of the naive algorithm

Challenges

- Applying complex parallelization stragies to code heavily featuring external libraries like `odeint`
- Complexity of our Barnes-Hut tree structure did not lend itself to simple MPI communication schemes

Potential MPI Barnes-Hut Solution

- Serialize the Quadtree and send it under a MPI_Type_contiguous defined MPI Datatype. Or using Boost.MPI dependencies serialization::access to create a friend class of the Quadtree, functioning in a similar manner to the first approach.
- We also need to make sure MPI Barriers are placed properly before broadcasting the tree to worker nodes.

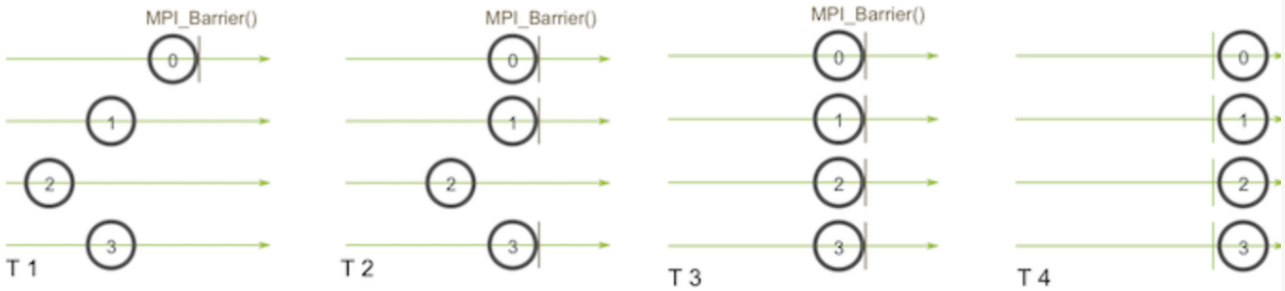


Figure: 1.MPI Barriers

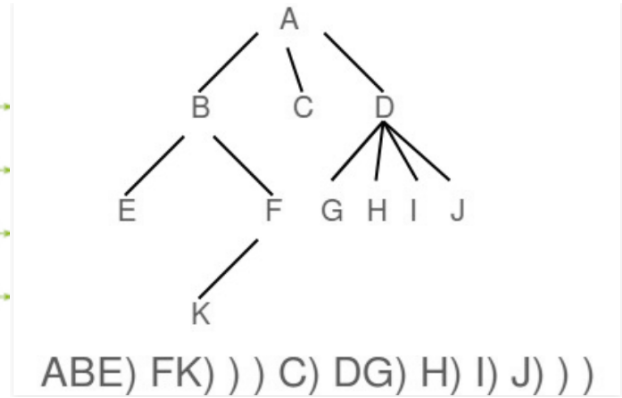


Figure: 2.Tree Serialization

Discussion and Summary

- We observed a tradeoff between efficiency (Barnes-Hut's lower algorithmic runtime) and accuracy (its approximations)
- We also observed a tradeoff between algorithmic and parallelization efficiency, as Barnes-Hut had lower speedups in general
- High MPI communication overhead due to the non-locality of the O'Keefe model led to lowered performance of MPI and OpenMP/MPI hybrid models compared to pure OpenMP models

References

- O'Keeffe and Bettstetter. *A review of swarmalators and their potential in bio-inspired computing*, https://arxiv.org/pdf/1903.11561.pdf. 2019.
- O'Keeffe, Hong, and Strogatz. *Oscillators that sync and swarm*, https://www.nature.com/articles/s41467-017-01190-3. 2017.
- Gan and Xu. *Efficient Implementation of the Barnes-Hut Octree Algorithm for Monte Carlo Simulations of Charged Systems* https://arxiv.org/pdf/1305.1825.pdf. 2013.

See https://github.com/myue2020/swarming-synching for more!