PARALLEL N-BODY SIMULATION

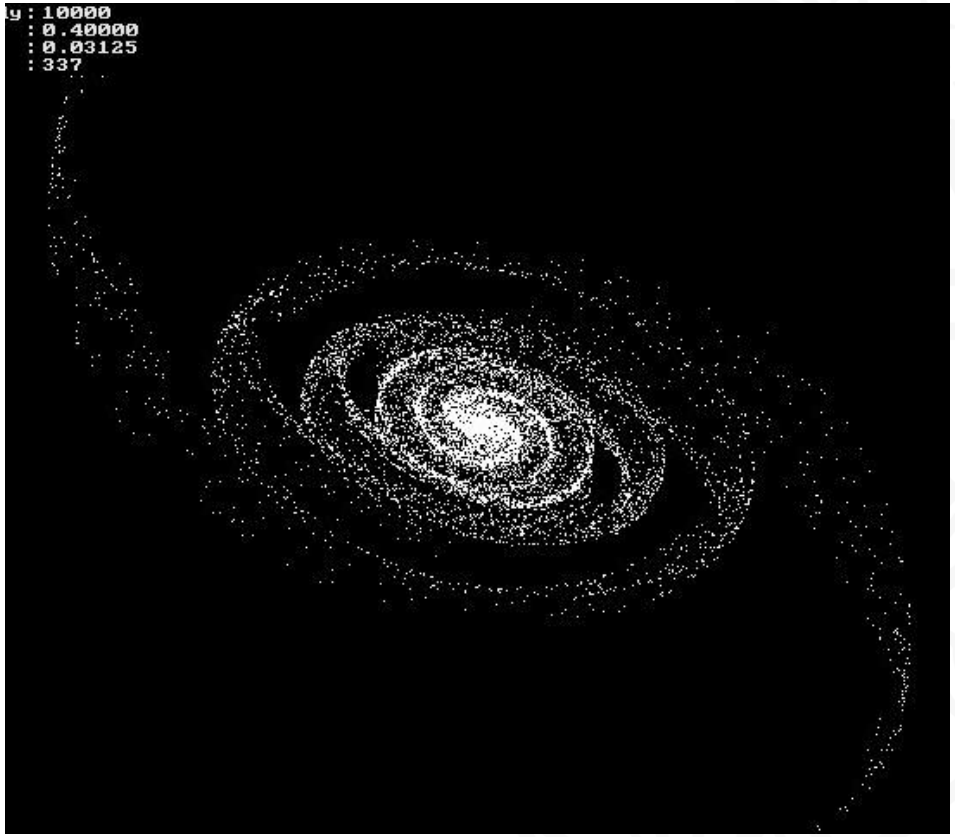
CSE-633 Parallel Algorithms

Anthony Huynh

University at Buffalo The State University of New York

What are N-Body Simulations?

- N Body Simulations are a way of modeling the interaction between some force and objects or bodies over a period of time to observer their behavior
- In this case we are modeling the effects of gravity on n-number of bodies

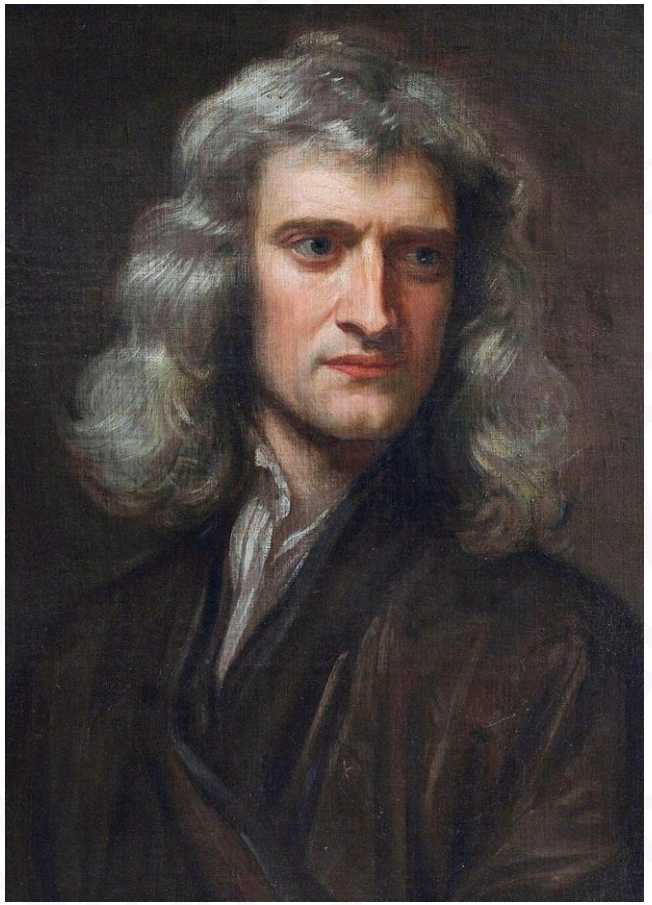


Background: Gravitational Force

- The gravitational force being applied from one particle to another particle can be calculated by the following:

$$- F = \frac{G m_1 m_2}{r^2}$$

  - Where G is the gravitation constant, m_1 and m_2 are the masses of the two objects and r is the distance between the two objects



Sequential Solution

- Sequential Algorithm
  - For each body calculate the force being applied to it by all other bodies, then calculate the displacement of the body
- Runtime: O(n^2) since we're for each body we're iterating through all other bodies to calculate the force
- While not very efficient, it is the most accurate since there is not much approximation happening

Faster Sequential Solution

- The parallel solution takes advantage of the inverse square law the (r^2 component of the previous force equation) , which translates to the force being applied to objects as distance diminishes exponentially so further bodies are less significant
- The faster solution can be divided into two parts, calculating the long-range forces being applied and the short-range forces

Computing Long Range Forces

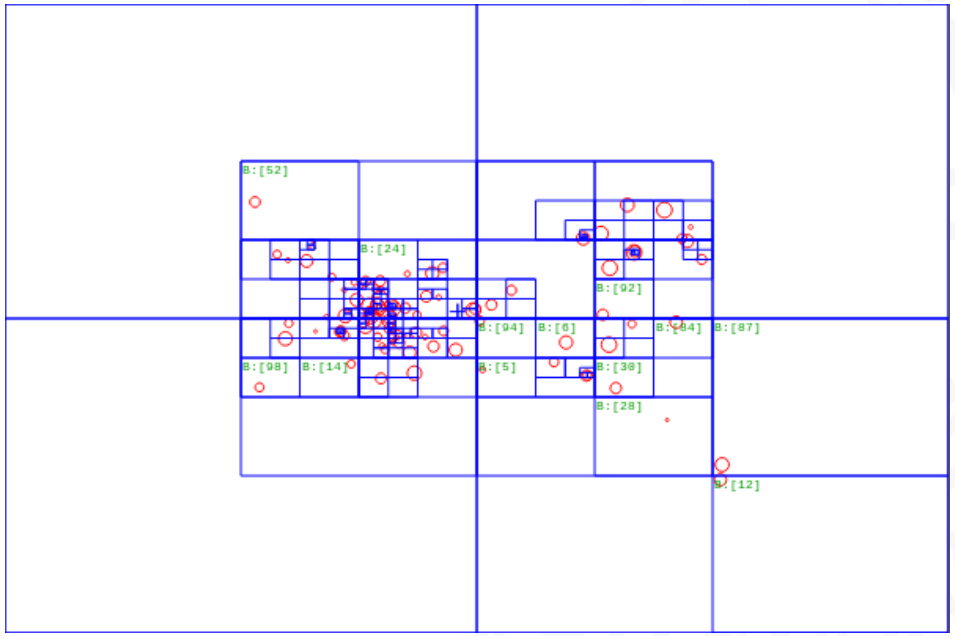
- Particle Mesh Method:
  - Overlay simulation space with grid
  - Assign each particle's mass to the nearest point on the grid
  - Solve Poisson's equation to get the potential field which has a scalar value for every point in our grid that represents the force being applied to that point of the grid
  - Apply that force from the grid point back to the particles assigned to that point
- Roughly (O g*log(g)), where g is the amount of grid points

Computing Short-Range Forces

- Using Barnes-Hut to calculate short range forces using a tree structure:
  - Root node is a box that contains all particles in the simulation space
  - For any node that has more then one particle subdivide box into 4 child boxes, if it contains 0 particles do not sub dive further, if it contains 1 particle is becomes a leaf node, we also calculate the center of mass of each populated node on the tree
  - For each particle we iterate through the tree, at each level we check if the distance of particle / center of mass of node < threshold we calculate force using G (mass of particle * total node mass) / distance to center of mass ^ 2 if > threshold we go into the children nodes
- O(nlog(n)) since we walking the tree (log(n)) at most n times

Combing Particle Mesh and Barnes-Hut

- We combine the two methods by calculating the long-range force first on the entire domain and then refining that further by performing the Barnes-Hutt on a sub-domain



Parallel Implementation MPI Details

- MPI Details:
  - Using Mpi_Dims_Create – we create a grid based on the number of nodes we have and the dimensions of our simulation
  - Mpi_Cart_Create – maps mpi communicator to grid
    - Each node now has its assigned portion of the simulation domain and knows its four neighbors, in addition each rank also holds a number of rows from the grid space for later calculation

MPI Details Cont.

- After each node assigns its particles to points on its local grid
- To compute the Poisson's equation on the whole grid we use the following Intel MKL library we do the following:
  - We create a Intel_DFT descriptor, each node then calls MKL_Cluster_DFT_ComputeFoward and MKL_Cluster_DFT_ComputerBackward which gives each node a portion scalar vector, we then use MPI_AlltoAllv so each node has full copy of the scalar.

MPI Details Cont.

- After the final displacements are calculated using Barnes-Hut each node checks which particles should be transferred to either of its 4 neighbors (up, down, left, right) we then use MPI_Neighbor_alltoallv to send and exchange particles
- MPI_Barrier is used to synchronize at the end to ensure all nodes are in the same time step

OpenMP

- Parallelize local operations such as:
  - Performing the initial particle grid assignments
  - Walking the tree during Barnes-Hut to calculate short range forces

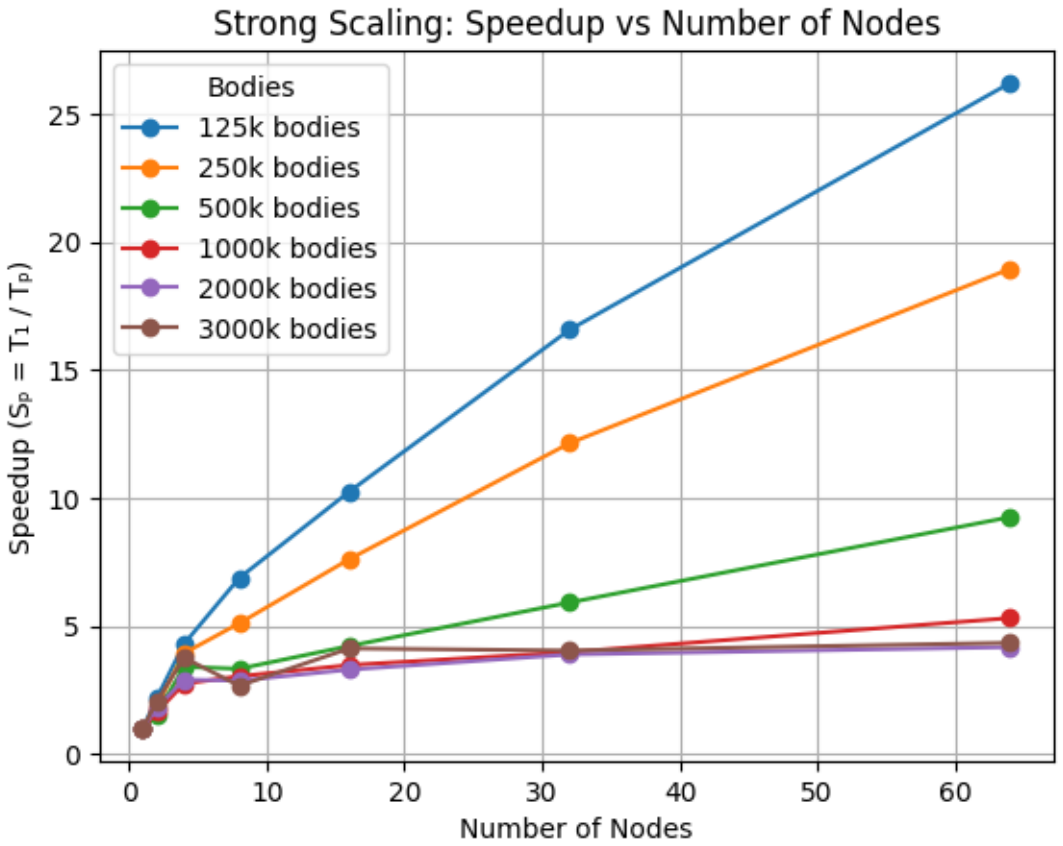
Benchmark Setup

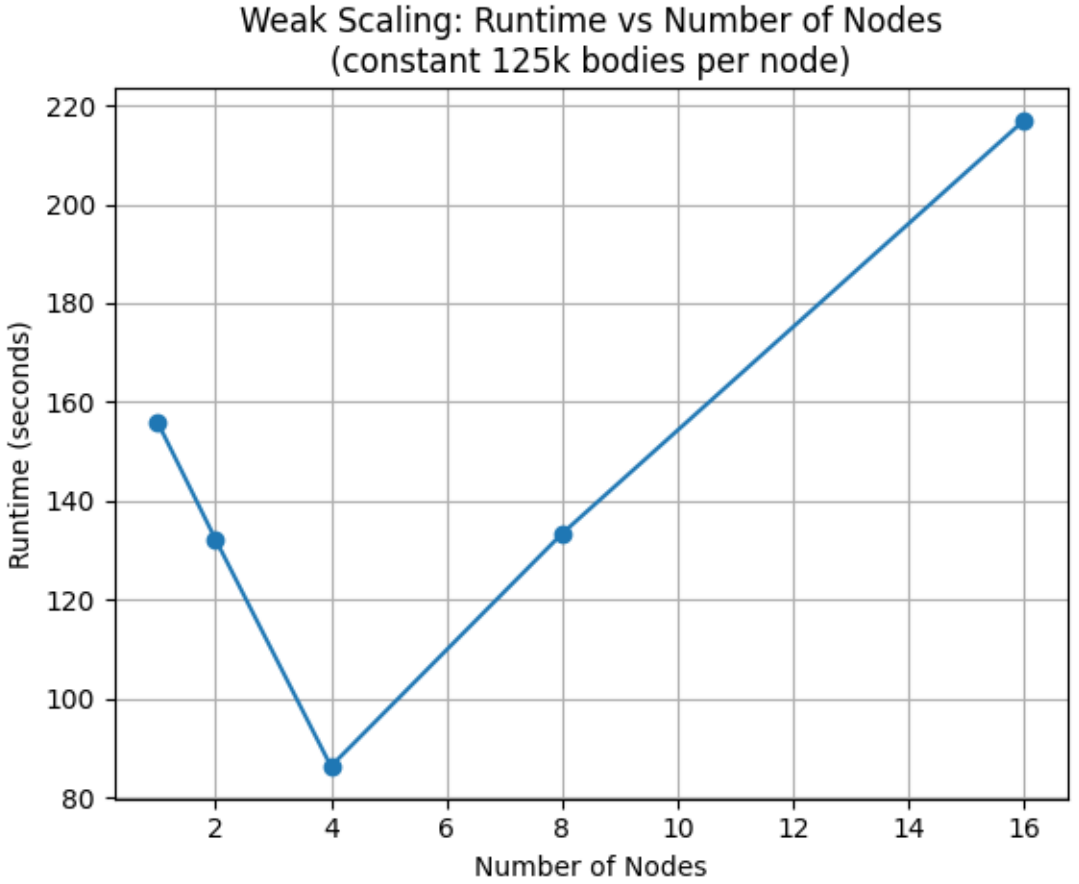
- Tested on 1,2,4,8,16,32,64 Nodes
- Node Specifications
  - 32 CPU Cores Intel Xenon Gold 6230
  - 32 GB Memory
  - InfiniBand Network
- Parameters:
  - N, Number of Bodies
  - 100 steps

Experiment Results

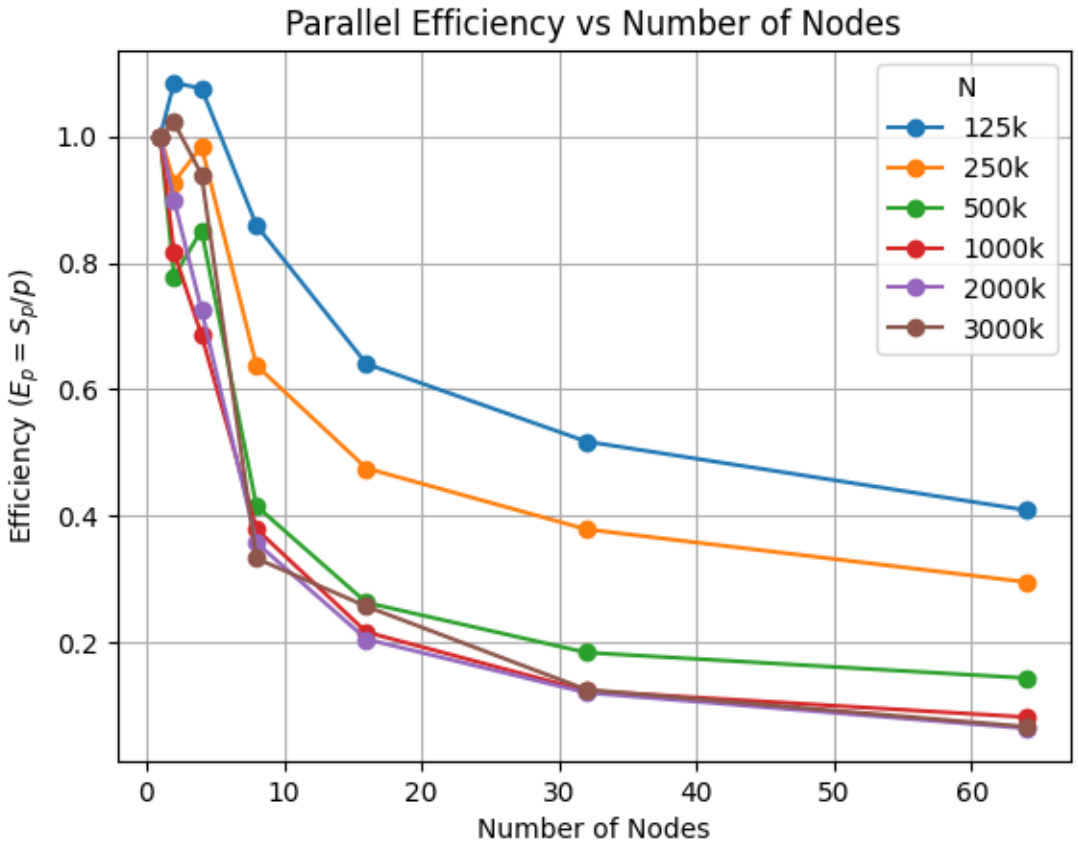
Number of Bodies (N)	1 Node (s)	2 Node (s)	4 Node (s)	8 Node (s)	16 Node (s)	32 Node (s)	64 Node (s)
125 000	155.902	71.931	36.2426	22.6844	15.2158	9.40864	5.94864
250 000	245.307	132.289	62.332	48.0661	32.237	20.2089	12.9458
500 000	294.209	189.13	86.394	88.3982	69.7557	49.7291	31.7995
1 000 000	404.993	248.128	147.532	133.393	116.593	101.843	76.2678
2 000 000	714.644	397.465	246.046	249.56	216.933	183.834	171.256
3 000 000	1117.14	546.766	297.292	418.905	270.986	276.456	257.279

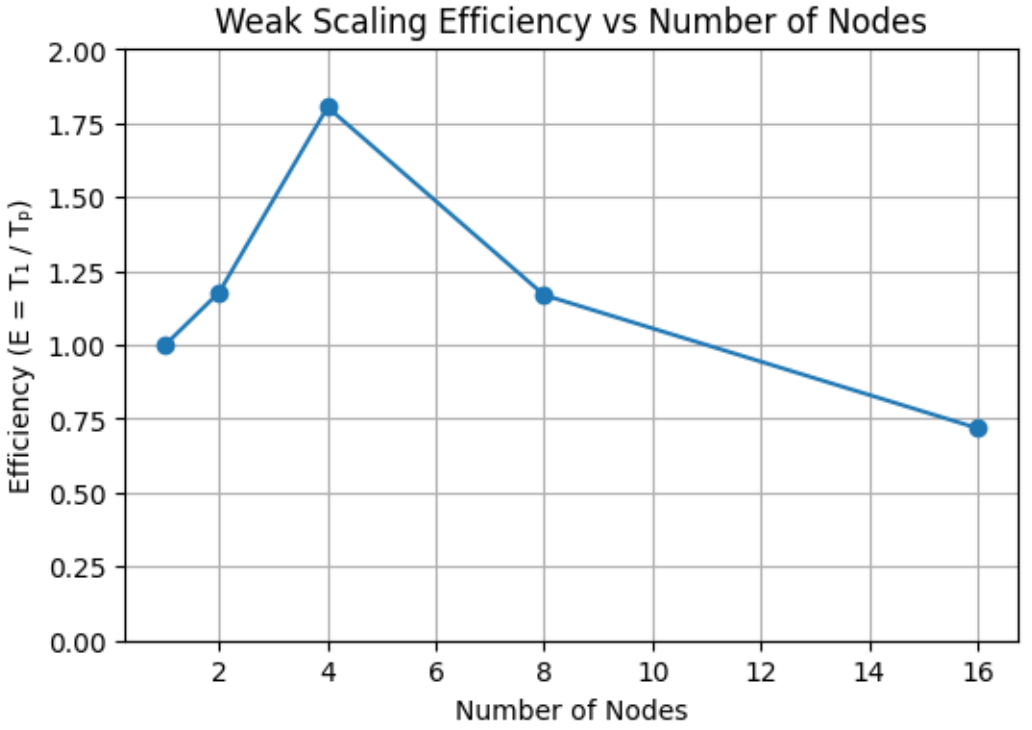
Speedup Results





Efficiency Results





Analysis

- Unfortunately, its not great, below are some potential bottlenecks:
  - Construction of the Barnes-Hut Tree is single threaded
  - Poor Memory Allocation and memory accessing patterns
  - Preprocessing before exchanging particles is not parallelized
  - Grid parameters not optimized leading to workload imbalance across threads in OpenMP
- Good strong scaling up to 8 nodes
- Decent workload balance across nodes
- Stable behavior across different nodes and sizes of n

References

Barnes, J. E., & Hut, P. (1986). A hierarchical O(N log N) force-calculation algorithm. Nature, 324(6096), 446-449. https://doi.org/10.1038/324446a0

Hockney, R. W., & Eastwood, J. W. (2021). *Computer Simulation Using Particles*. https://doi.org/10.1201/9780367806934

Intel Corporation. (2024). Intel® MPI Library Developer Reference for Linux* OS. https://www.intel.com/content/www/us/en/docs/mpi-library/developer-reference-linux/2021-8/overview.html

OpenMP Architecture Review Board. (2024). OpenMP API 6.0 Specification. https://www.openmp.org/specifications/