CSE 633 Spring 2014

N-Body Simulation

Devanshu Mukherjee
Munish Mehra

What is N-body Simulation?

Simulation of a dynamical system of particles, usually under the influence of physical forces, such as gravity.

$$F= G*m1*m2/(r^2)$$

Objective

- Simulate the gravitational forces acting between a number of bodies in space.
- Barnes-Hut Tree algorithm for optimization of the force calculation.
- Implementation of the project using MPI.
- Comparison of different approaches.

bodies have positions (x1, y1) and (x2, y2), and masses m1 and m2

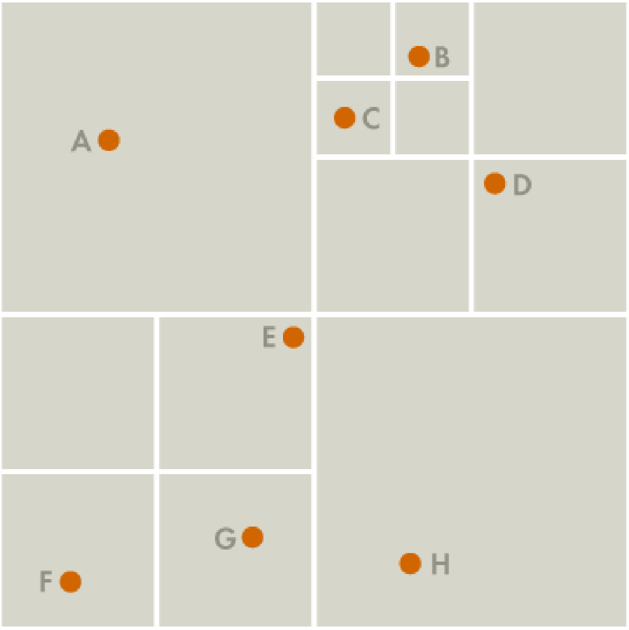
The Barnes-Hut Algorithm

- Speeding up the brute force n-body algorithm is to group nearby bodies and approximate them as a single body.
- If the group is sufficiently far away, we can approximate its gravitational effects by using its *center of mass*.
- Two bodies (x1, y1) of mass ‘m1’, and (x2,y2) of mass ‘m2’.

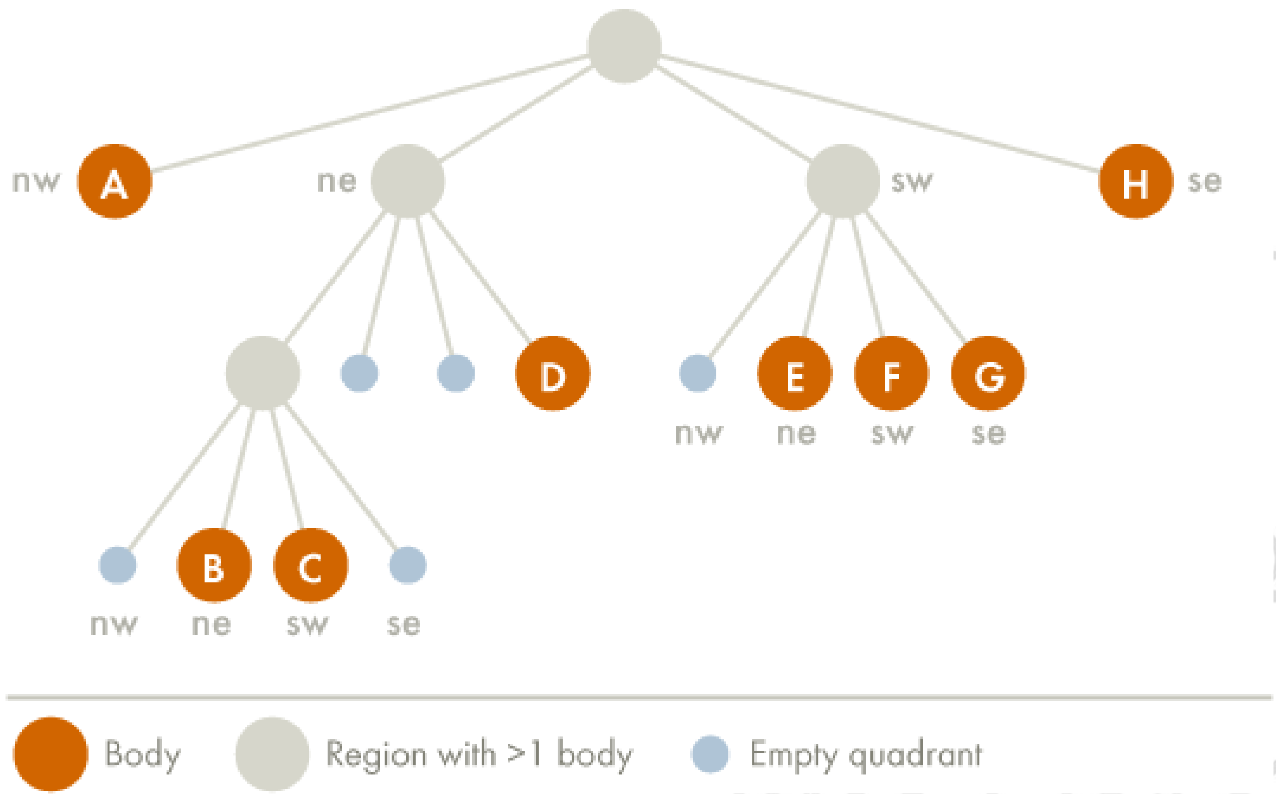
$$m = m1 + m2$$
$$x = (x1*m1 + x2*m2) / m$$
$$y = (y1*m1 + y2*m2) / m$$

The Barnes-Hut Algorithm

- It recursively divides the set of bodies into groups by storing them in a *quad-tree*.
- The topmost node represents the whole space, and its four children represent the four quadrants of the space.



The Barnes-Hut Algorithm



The Barnes-Hut Algorithm

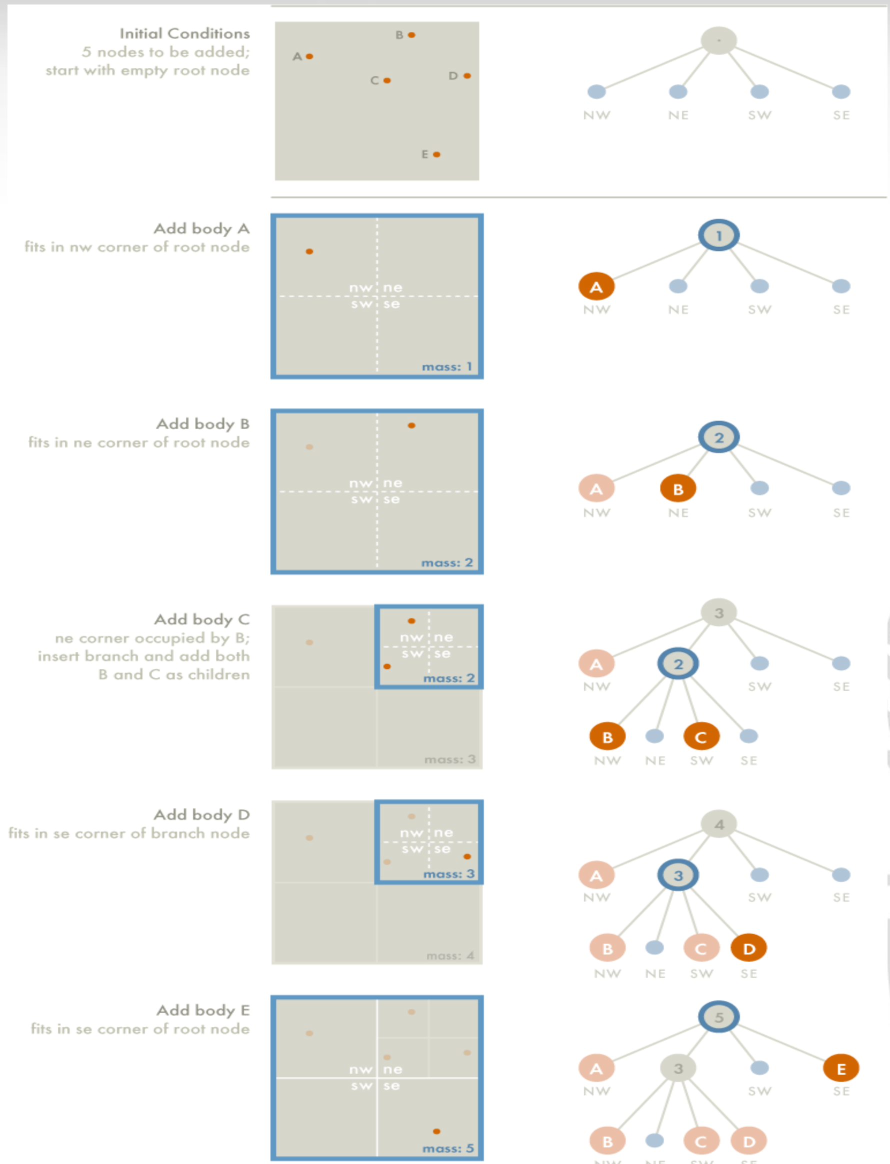
- Determine if $(s / d) < \Theta$
- *s* is the width of the region represented by the internal node,
- *d* is the distance between the body and the node's center-of-mass
- Θ can change the speed and accuracy of the simulation. Typically, 0.5.

The Barnes-Hut Algorithm

Constructing the Barnes-Hut tree :

To insert a body *b* into the tree rooted at node *x*, use recursive procedure:

- *If node x does not contain a body, put the new body b here.*
- *If node x is an internal node, update the center-of-mass and total mass of x. Recursively insert the body b in the appropriate quadrant.*
- *If node x is an external node, subdivide the region further by creating four children. Then, recursively insert both b and c into the appropriate quadrant(s).*
- *Finally, update the center-of-mass and total mass of x.*

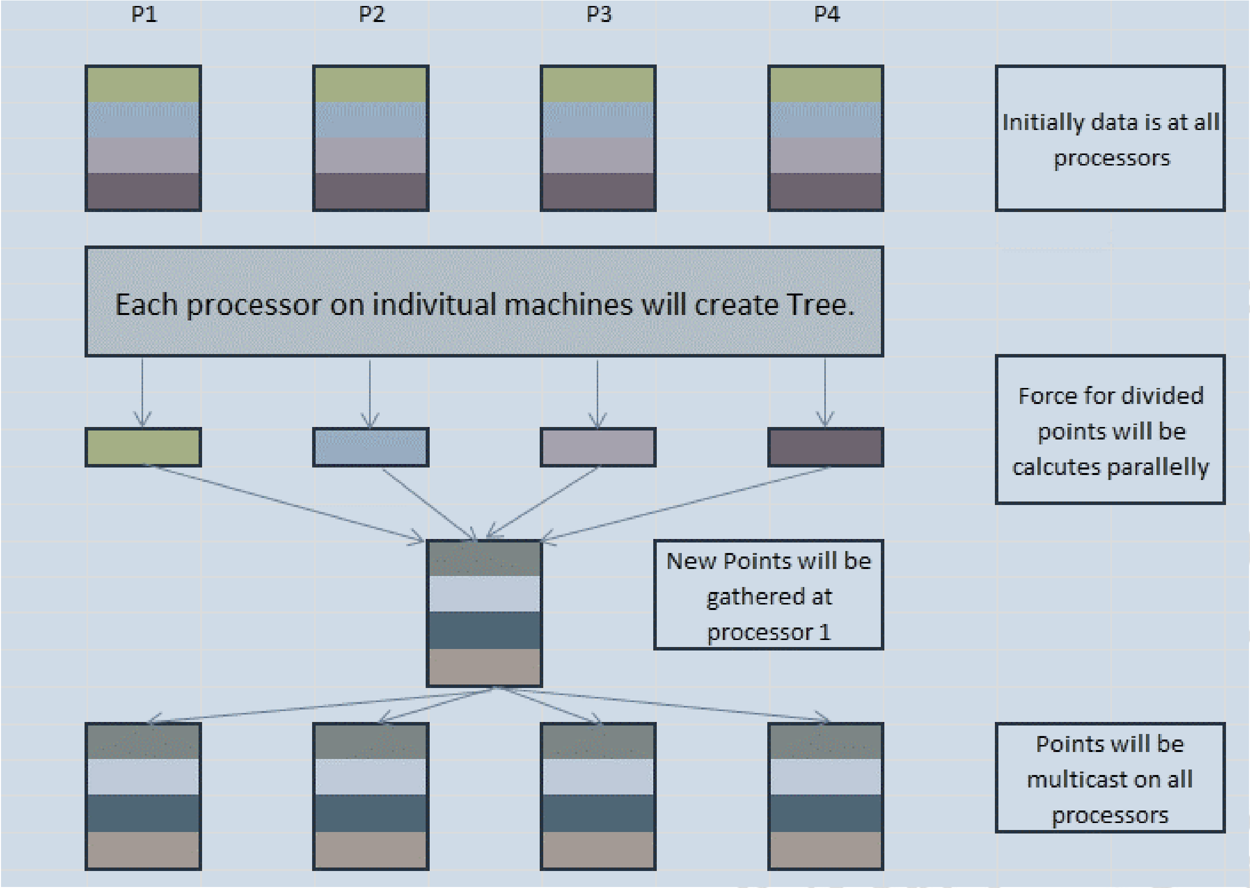

Initial Conditions
5 nodes to be added;
start with empty root node
A
B
C
D
E
NW
NE
SW
SE
Add body A
fits in nw corner of root node
nw ne
sw se
mass: 1
Add body B
fits in ne corner of root node
mass: 2
Add body C
ne corner occupied by B;
insert branch and add both
B and C as children
mass: 3
Add body D
fits in se corner of branch node
mass: 4
Add body E
fits in se corner of root node
mass: 5

Our Attempt

1. Master – Worker Configuration:

- Parallel Tree Formation
  - Every node reads data from input file.
  - Formation of quad-tree at all nodes in parallel.
- Parallel Force Calculation
  - Every processor selects bodies from input file based on its rank.
  - Calculate force on the selected bodies and there new position due to the force.
- Merge Partial Results
  - Merge the partial results from all the nodes at master node to get the final result.
  - Broadcast the new dataset to all nodes.

1. Master – Worker Configuration:



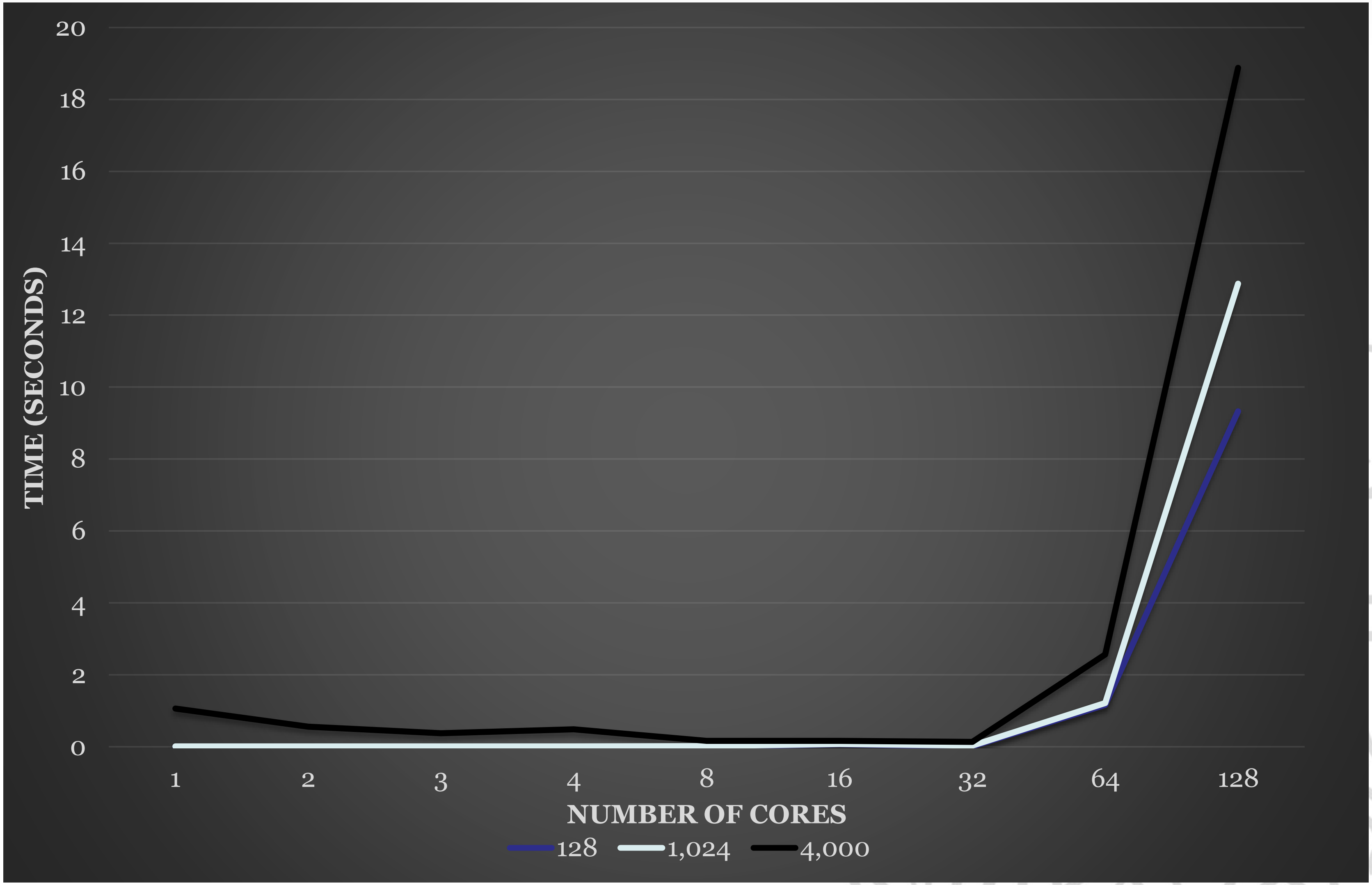
1. Master – Worker Configuration:

Number of Bodies vs Number of Cores (8 Cores/Node)

	1	2	3	4	8	16	32	64	128	256
128	0.00415	0.0034	0.00379	0.00412	0.00658	0.069261	0.007926	1.164081	9.328586	15.090703
1024	0.007804	0.006331	0.006141	0.006481	0.00813	0.01089	0.024016	0.054247	3.547129	9.35108
4,000	1.05145	0.546004	0.367569	0.47367	0.148349	0.084429	0.14190	1.348655	6.092180	10.570122
10,000	7.204272	3.585363	2.414663	1.829134	1.721207	3.879273	11.297868	16.450013	33.001744	70.34019
20,000	28.87630	16.664297	11.29883	9.454216	7.055522	12.517156	13.48926	14.109527	42.71600	91.23014
40,000	125.70631	63.20303	42.38242	33.10332	25.78558	27.124613	36.43922	48.252101	79.216301	114.675226

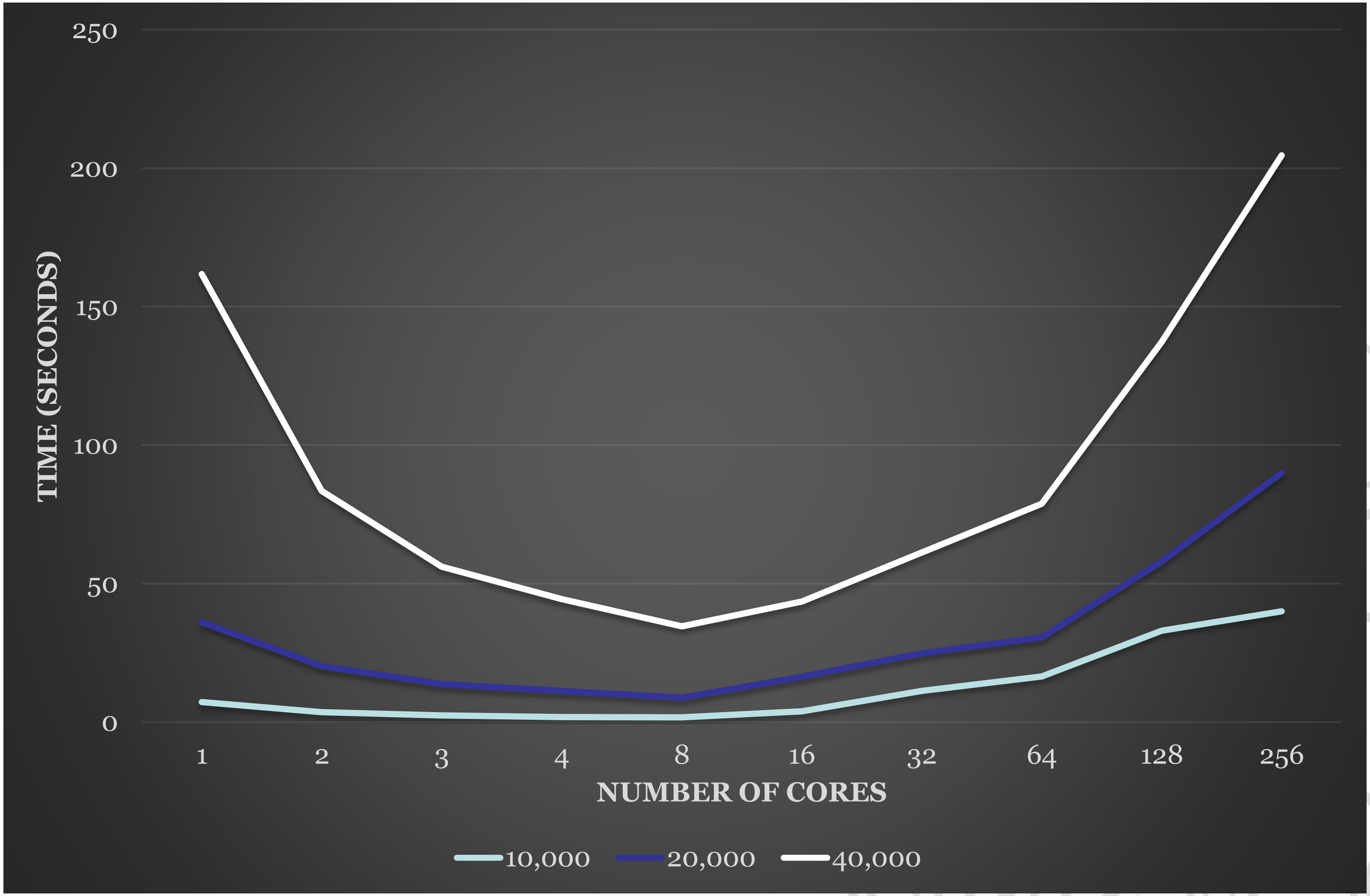
1. Master – Worker Configuration:

Time vs No of Cores (8 Cores/Node)



1. Master – Worker Configuration:

Time vs No of Cores (8 Cores/Node)

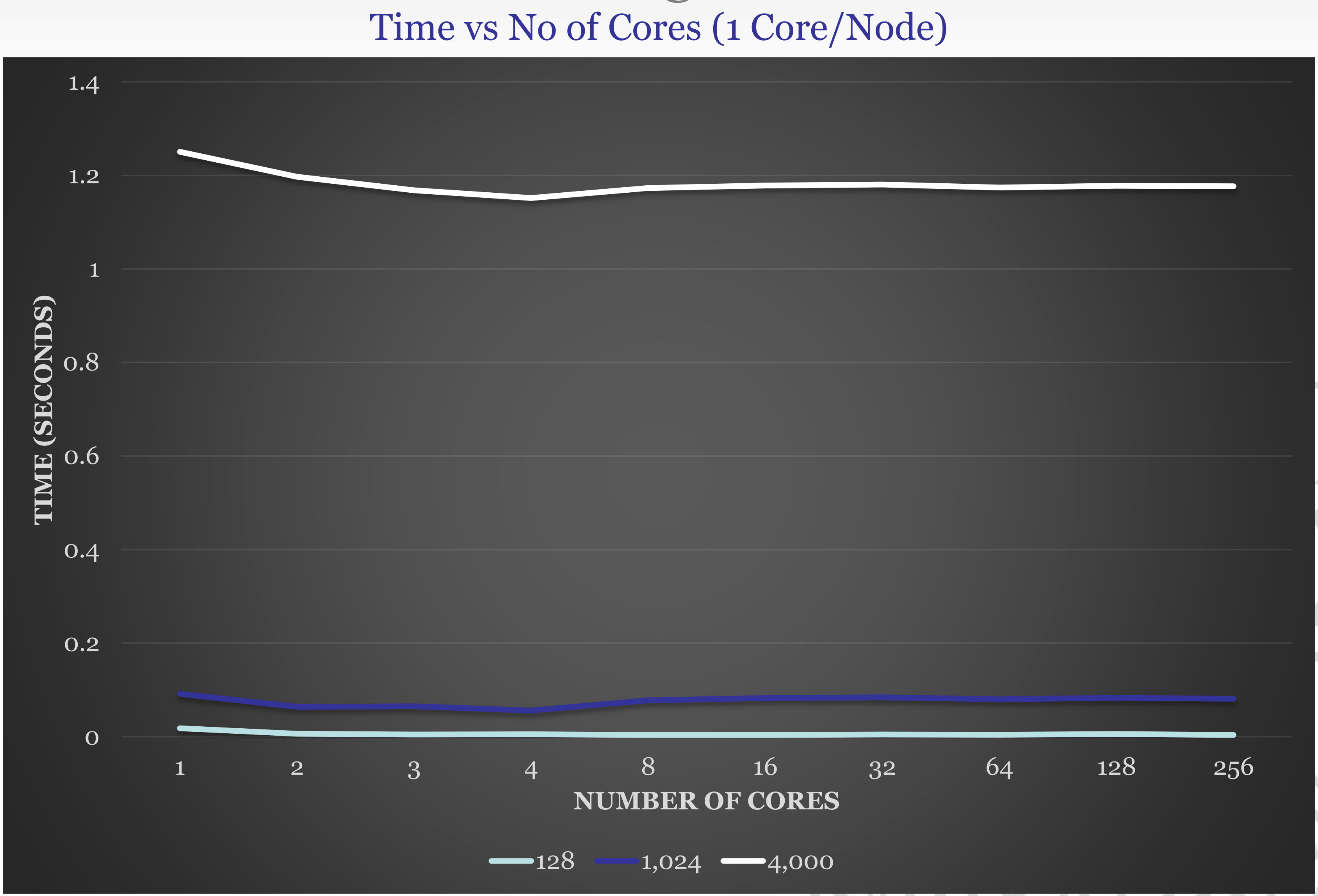


1. Master – Worker Configuration:

Number of Bodies vs Number of Cores (1 Core/Node)

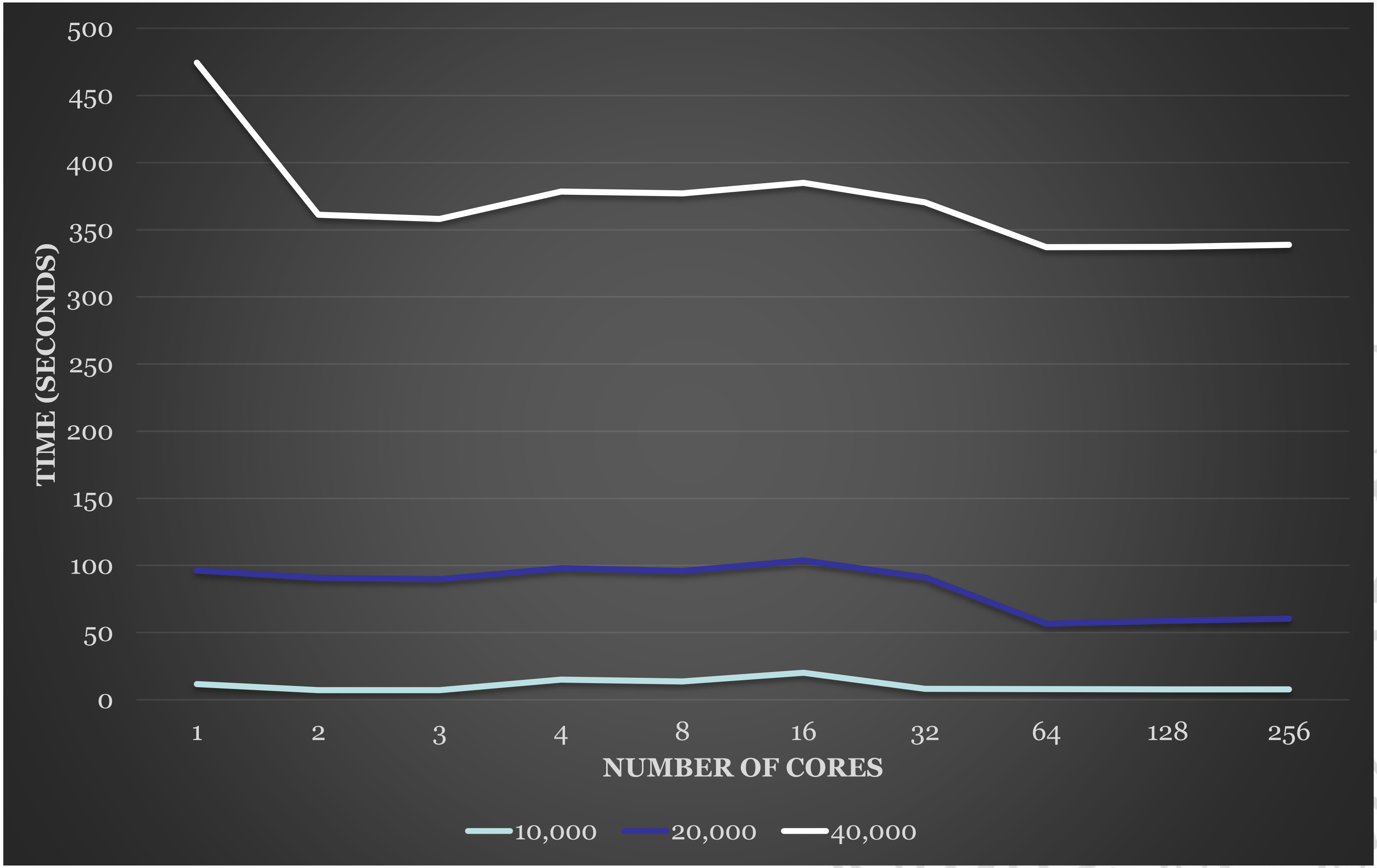
	1	2	3	4	8	16	32	64	128	256
128	0.01806	0.00616	0.00452	0.00532	0.00325	0.00369	0.00456	0.00379	0.0058	0.00359
1024	0.07366	0.05778	0.0604	0.05033	0.07467	0.07871	0.07971	0.07617	0.07723	0.07723
4000	1.15893	1.13307	1.10315	1.09626	1.09492	1.09624	1.09624	1.09402	1.09466	1.0961
10000	11.6091	7.187	7.1992	14.9769	13.6214	20.1081	8.23125	8.02496	7.73448	7.81518
20000	84.6857	83.6755	82.6016	83.0337	82.21	83.5951	82.9271	48.6264	51.0667	52.4524
40000	378.088	270.424	268.216	280.575	281.261	281.266	279.194	280.503	278.525	278.534

1. Master – Worker Configuration:



1. Master – Worker Configuration:

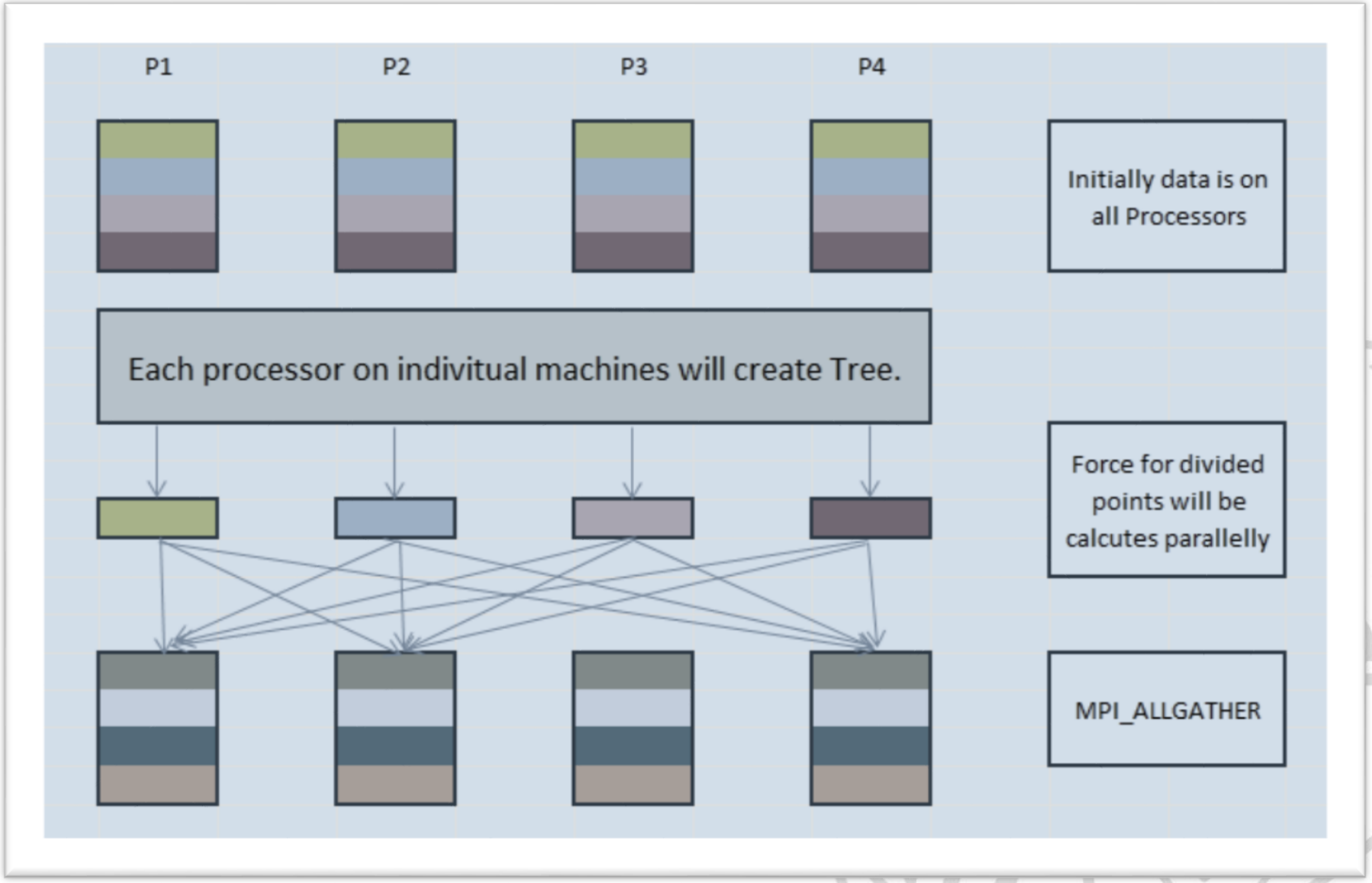
Time vs No of Cores (1 Core/Node)



2. All to All Configuration:

- Parallel Tree Formation
  - Every node reads data from input file.
  - Formation of quad-tree at all nodes in parallel.
- Parallel Force Calculation
  - Every processor selects bodies from quad-tree based on its rank.
  - Calculate force on the selected bodies and there new position due to this force.
- Merge Partial Results
  - Every node gathers partial result from all the other nodes using **MPI_Allgather.**

2. All to All Configuration:

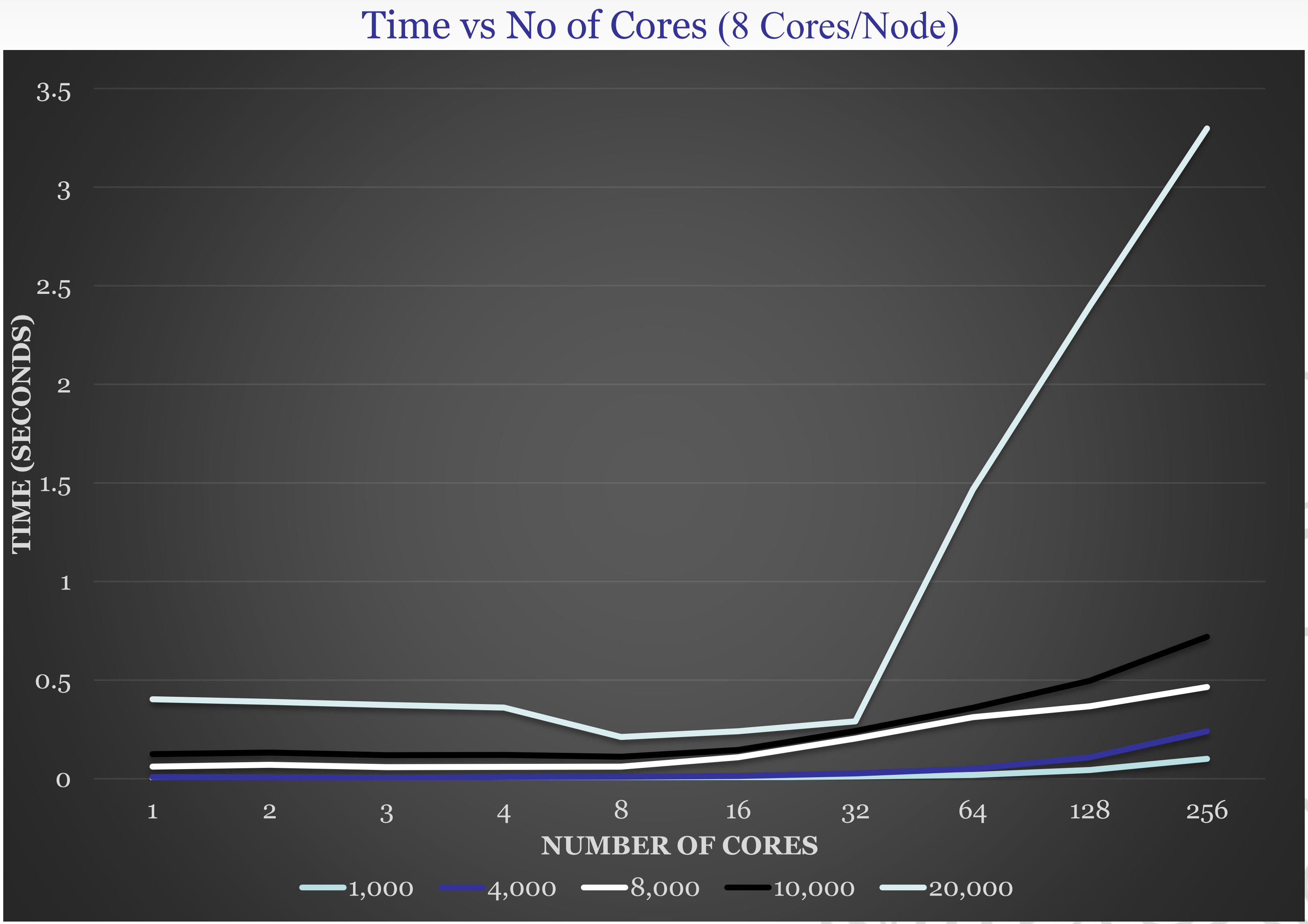


2. All to All Configuration:

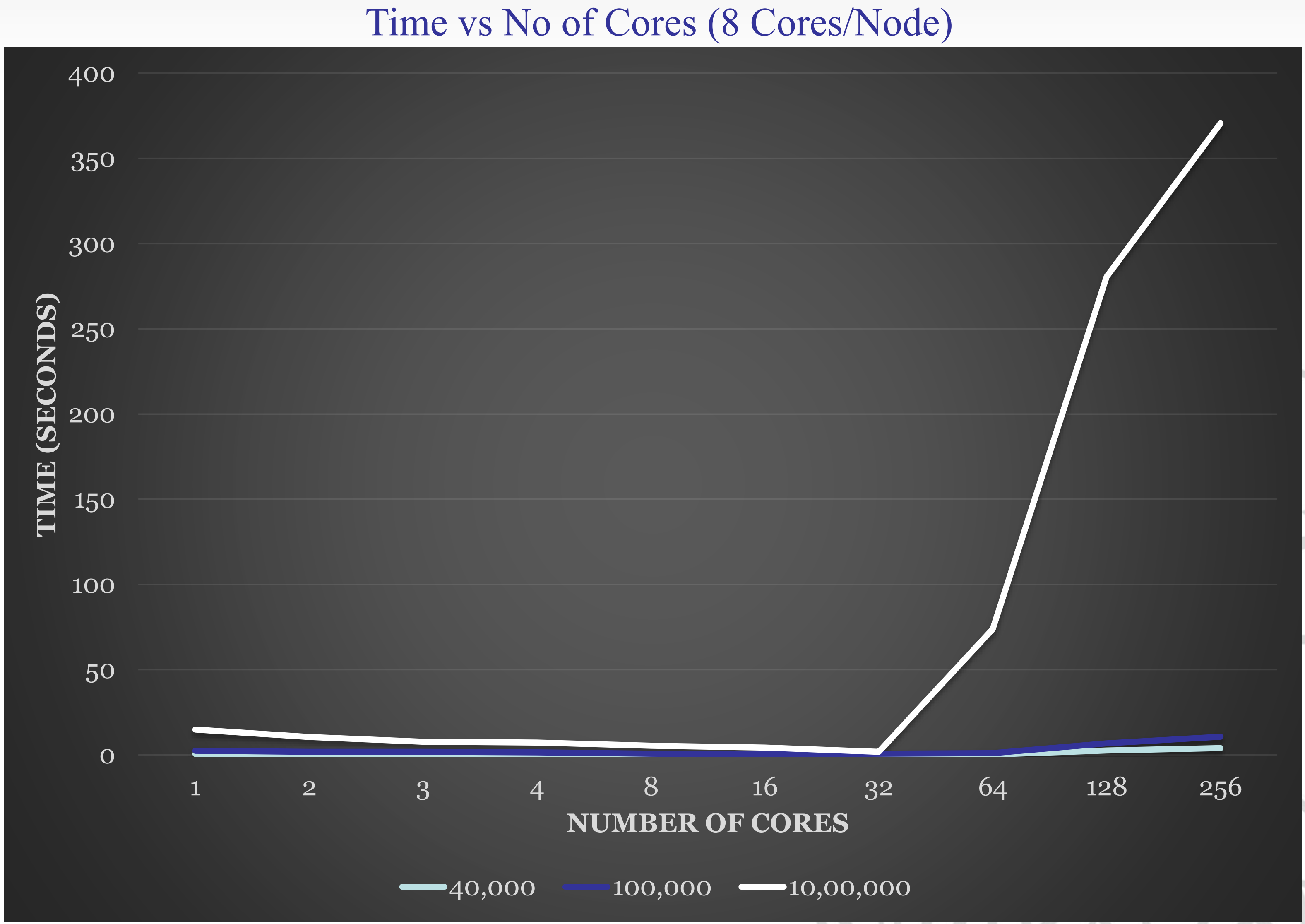
Number of Bodies vs Number of Cores (8 Cores/Node)

	1	2	3	4	8	16	32	64	128	256
1000	0.00314	0.0020	0.00171	0.00199	0.00282	0.004832	0.010829	0.018858	0.043453	0.167657
4000	0.00517	0.0042	0.00339	0.005657	0.00627	0.008842	0.016211	0.031286	0.064436	0.139774
8000	0.05313	0.0640	0.05303	0.052184	0.051377	0.095169	0.176234	0.261428	0.25843	0.225114
10000	0.06300	0.06113	0.06096	0.060151	0.05019	0.036377	0.037995	0.046847	0.130437	0.254299
20000	0.27814	0.2573	0.25487	0.24071	0.10119	0.095223	0.048549	1.10658	1.90124	2.579095
40000	0.56598	0.51989	0.50833	0.50402	0.20842	0.170834	0.19661	0.243217	2.532225	3.993428
100000	1.798	1.30278	1.209313	0.98268	0.42435	0.230418	0.471686	0.759693	4.263693	6.618865
1000000	12.4243	8.60192	6.00739	5.697782	4.69039	3.914725	1.193584	72.843632	273.71508	
10000000	71.4634	43.5849	29.95473	21.36642	12.5039	46.167598				

2. All to All Configuration:



2. All to All Configuration:

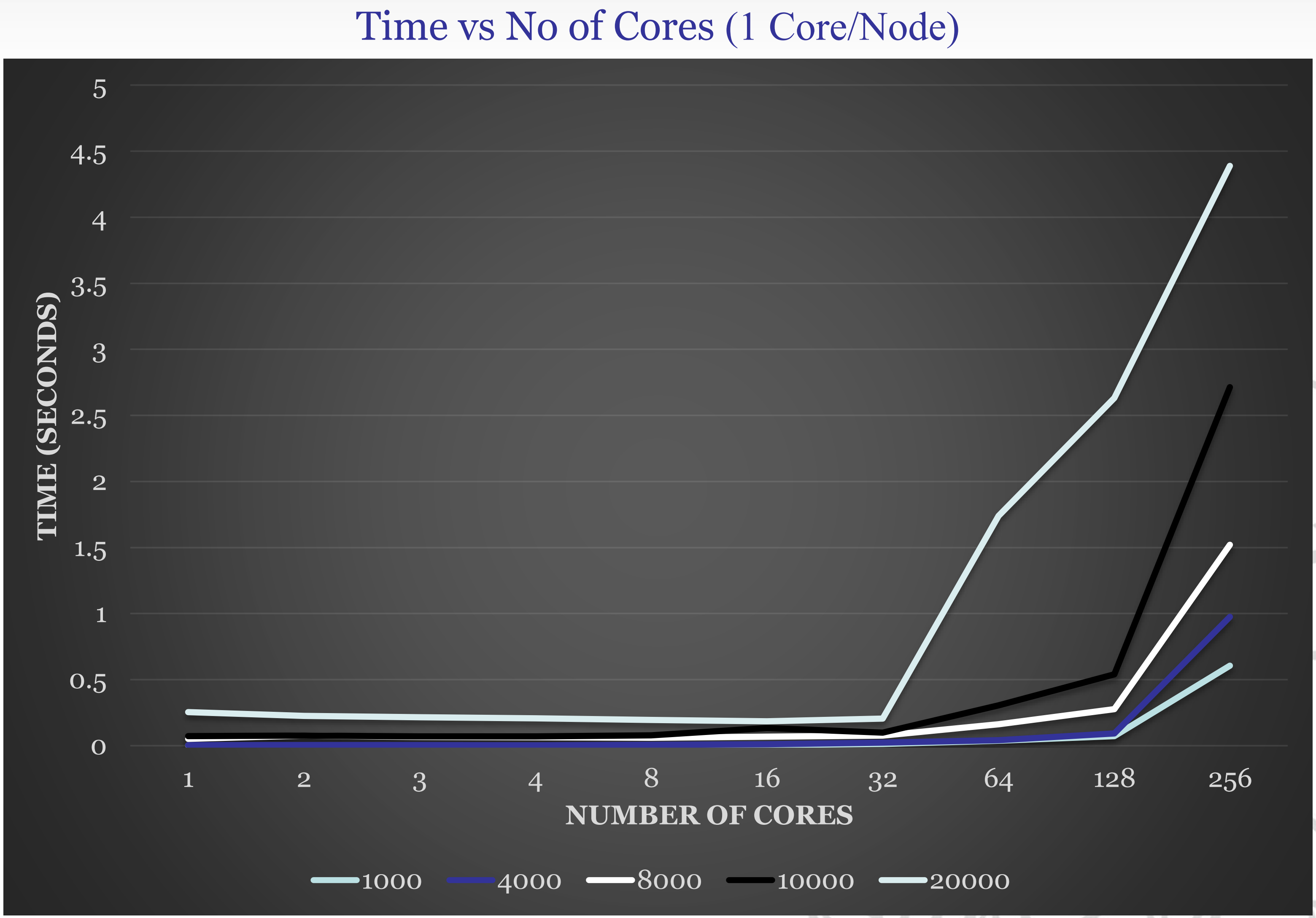


2. All to All Configuration:

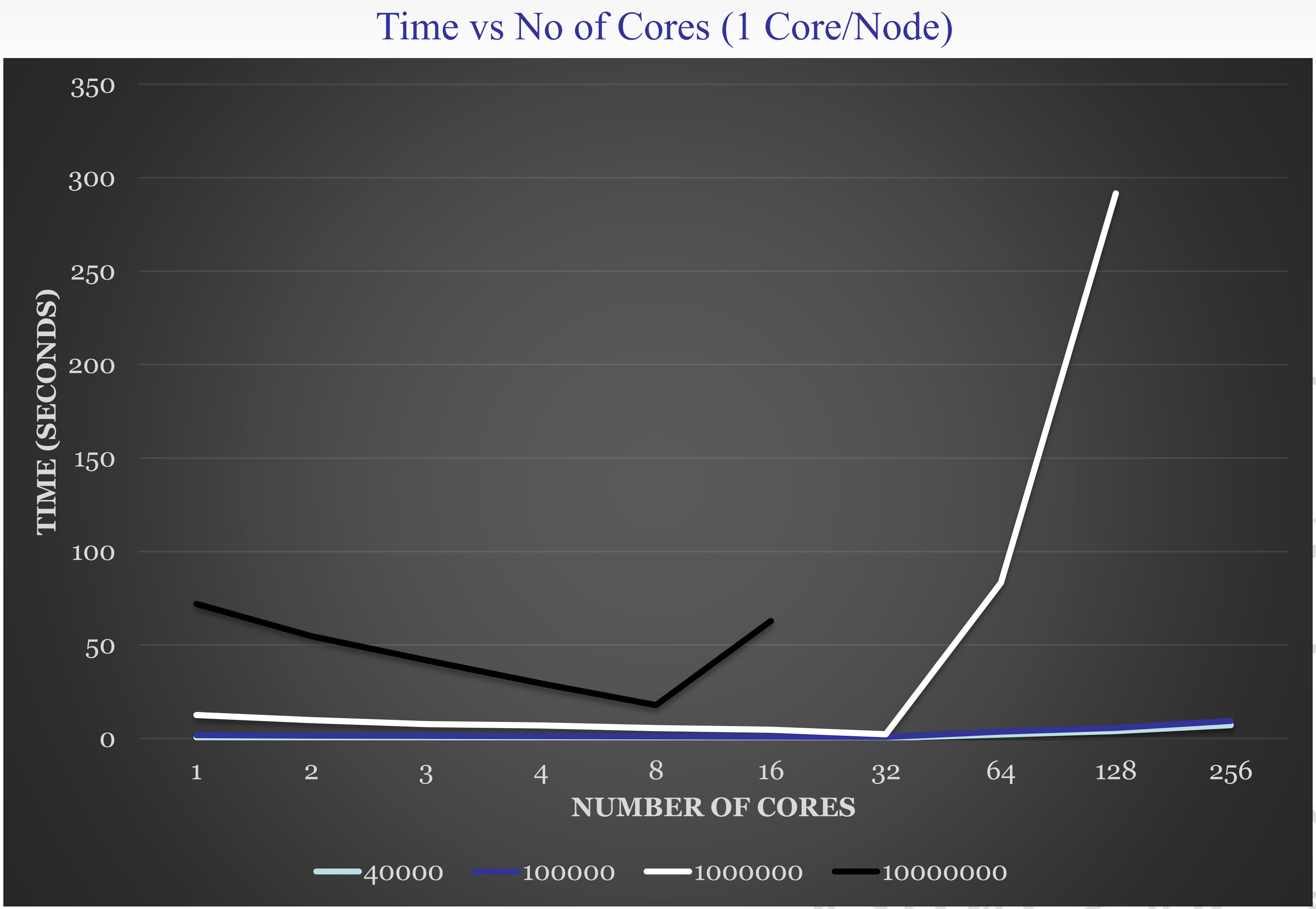
Number of Bodies vs Number of Cores (1 Core/Node)

	1	2	3	4	8	16	32	64	128	256
1000	0.00322	0.0029	0.0021	0.0020	0.00367	0.005291	0.012746	0.037138	0.07351	0.60512
4000	0.00497	0.0044	0.00413	0.003879	0.007364	0.010942	0.02389	0.042596	0.092569	0.97594
8000	0.05302	0.07591	0.06739	0.064371	0.06075	0.06753	0.0760231	0.162841	0.277403	1.52179
10000	0.07310	0.07589	0.07089	0.070012	0.07918	0.133816	0.0984027	0.3053662	0.539712	2.71329
20000	0.25491	0.22671	0.21561	0.20683	0.195752	0.184693	0.204915	1.73914	2.6319	4.3891
40000	0.56071	0.53892	0.52593	0.519768	0.37516	0.298305	0.24730	2.13840	3.86032	7.09152
100000	1.8065	1.5491	1.3118	1.1863	1.05293	0.91742	0.76491	3.57921	5.64190	9.42618
1000000	12.5017	9.7859	7.71932	6.89257	5.49581	4.61475	2.14739	83.27491	291.6317	
10000000	72.0049	54.7293	41.89721	29.36642	17.83714	62.81534				

2. All to All Configuration:



2. All to All Configuration:



Observations

1. Master-Worker Configuration:
   - Best result for 8 cores per node is achieved with 4-8 cores.
   - Best results for 1 core per node:
     - For 128 bodies, best result achieved with 3 cores. Increasing cores after that did affect performance much.
     - For 1,024 and 4,000 bodies, best result achieved with 4 cores.
     - For 10,000 and 40,000 bodies, best result achieved with 64 cores.
     - For 20,000 bodies, best result achieved with 2-3 cores.

Observations

2. All to All Configuration:
   - Best results for 8 cores per node :
     - For small datasets (1000-8000 bodies) best result is achieved with 8 cores.
     - For medium datasets (10,000 – 1 Million bodies) best result is achieved with 16 cores.
     - For large datasets (10 Million bodies) best result is achieved with 8 cores.
   - Best results for 1 core per node :
     - For small datasets (1000-10,000 bodies) best result is achieved with 4 cores.
     - For medium datasets (20,000 – 1 Million bodies) best result is achieved with 32 cores.
     - For large datasets (10 Million bodies) best result is achieved with 8 cores.
   - After the best configuration, adding more cores increases running time due to communication overhead.

Conclusion

1. Master-Worker Configuration:
   - Load Distribution: Better than All-to-All configuration as the dataset is distributed for force calculation.
   - Running time more than All-to-All configuration due to communication overhead.
   - Due to sending of whole dataset from the master to other nodes, could not run on datasets having more than 40000 bodies.
2. All to All Configuration:
   - Load Distribution: Worse than Master-Worker configuration as each core processes a subset of tree for force calculation and number of bodies may vary in each part of tree.
   - Running time less than Master-Worker configuration due to less communication overhead as only partial results are sent.
   - Due to less communication overhead, running program with larger datasets was possible.

References

- The Barnes-Hut Algorithm - **TOM VENTIMIGLIA & KEVIN WAYNE** - http://arborjs.org/docs/barnes-hut
- Planar Decomposition for Quadtree Data Structure – **PINAKI MAZUMDER**
- An Effective Way to Represent Quadtrees – **JAMES FOLEY**