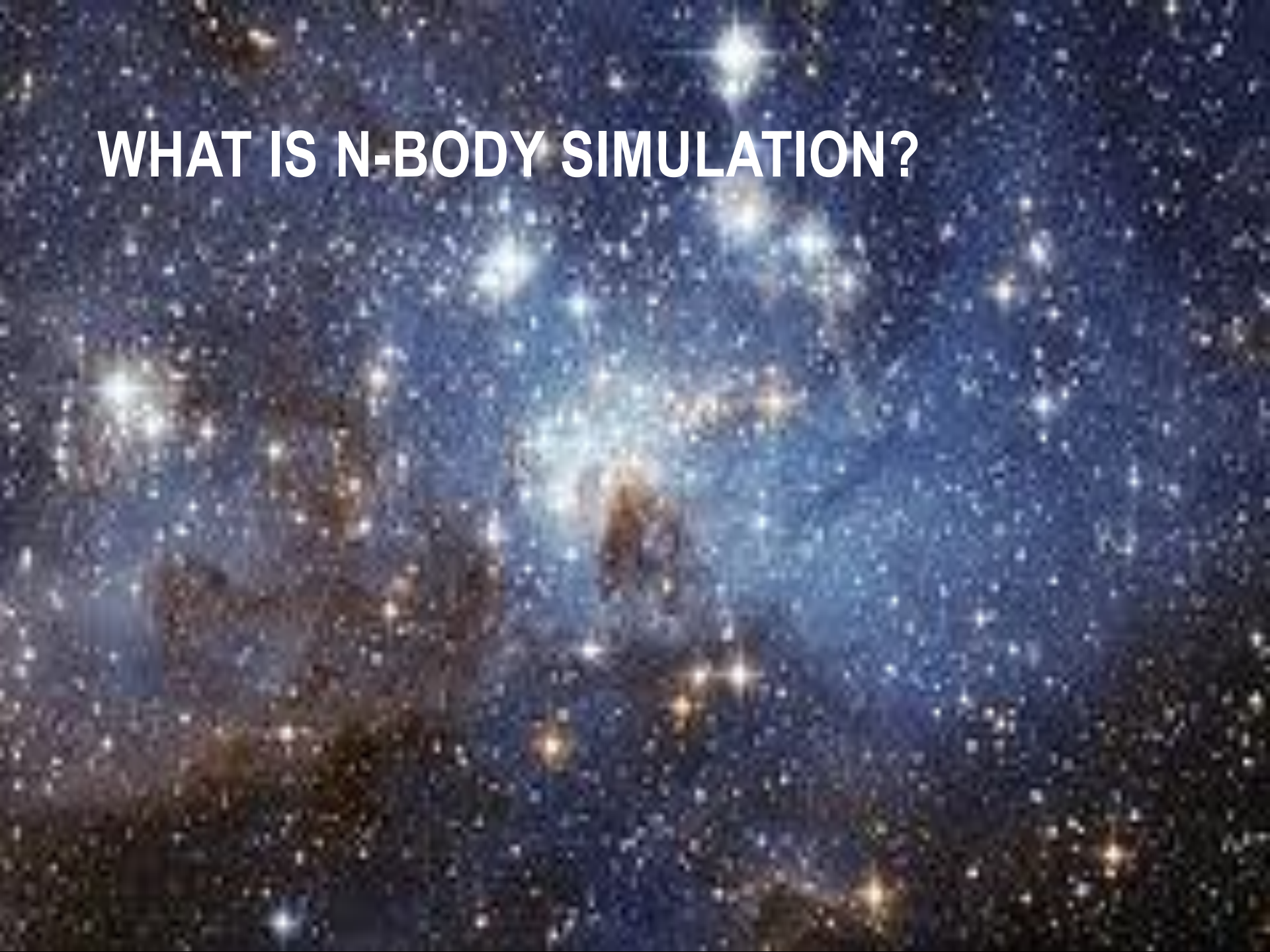
N-BODY SIMULATION

CSE-633 Fall 2012

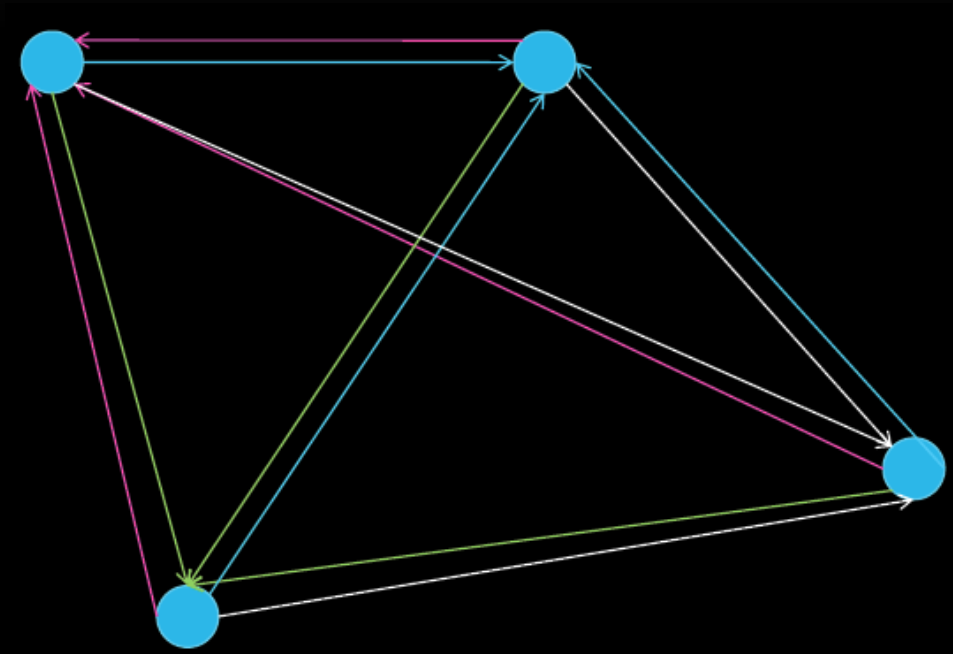
Sanjeev Pandey


WHAT IS N-BODY SIMULATION?

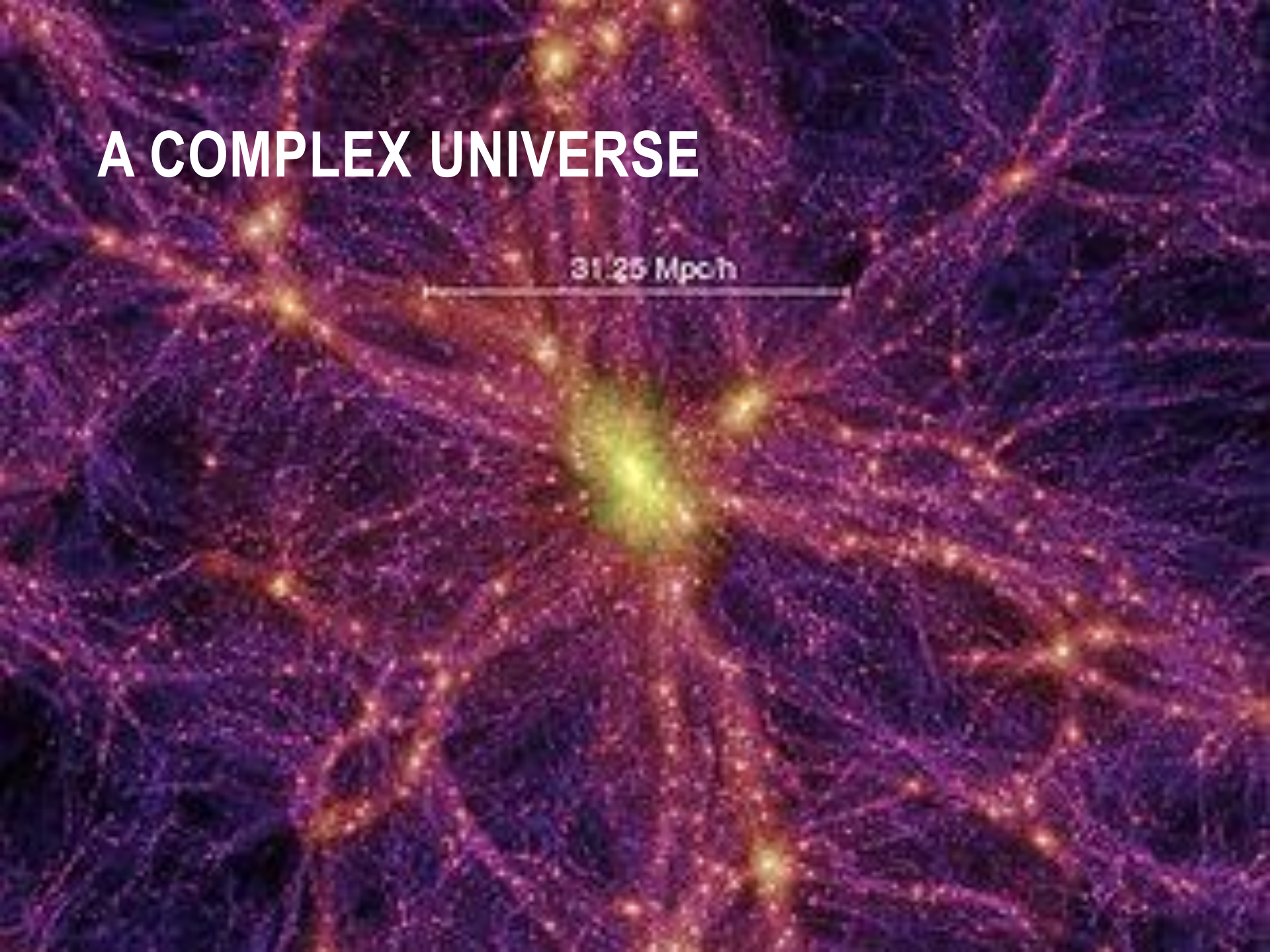
PROJECT

- A program to simulate the gravitational forces acting between a number of bodies in space.
- Newtonian laws, do the gravitational force calculation for each body in the space and calculate their position and velocity at a given time in future.
- Barnes-Hut Tree algorithm for optimization of the calculation.
- Implementation of the project using MPI.
- Comparison with sequential code.

SIMPLE N-BODY SCENARIO WITH 4 BODIES



A COMPLEX UNIVERSE



THE FORCE CALCULATION

The force exerted by partcle j on particle i is given by Newton's Law of Gravity.

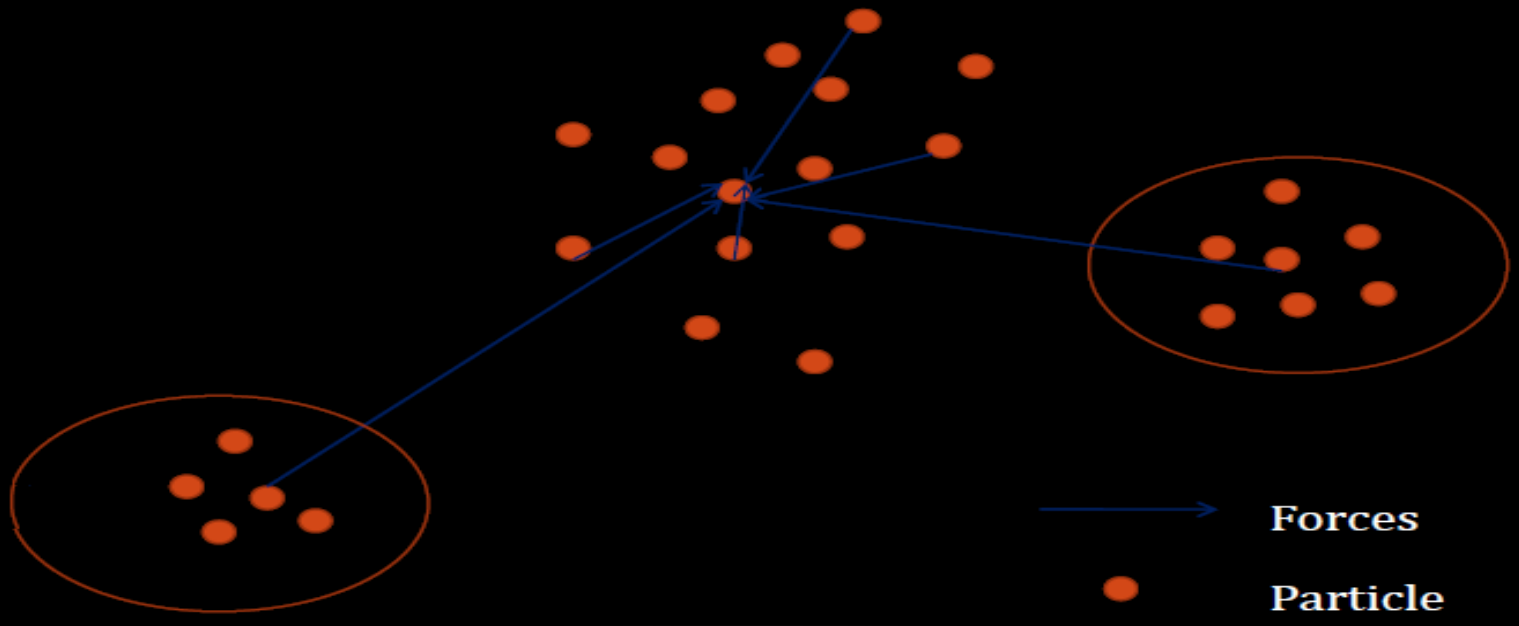
$$F_{ij} = -Gm_i m_j \frac{r_i - r_j}{\left\| r_i - r_j \right\|^3}$$

and the total force acting on particle i is

$$F_i = \sum_{j=1, j \neq i}^{N} F_{ij}$$

- This calculation suggests that the complexity is $O(n^2)$.

OPTIMIZATION

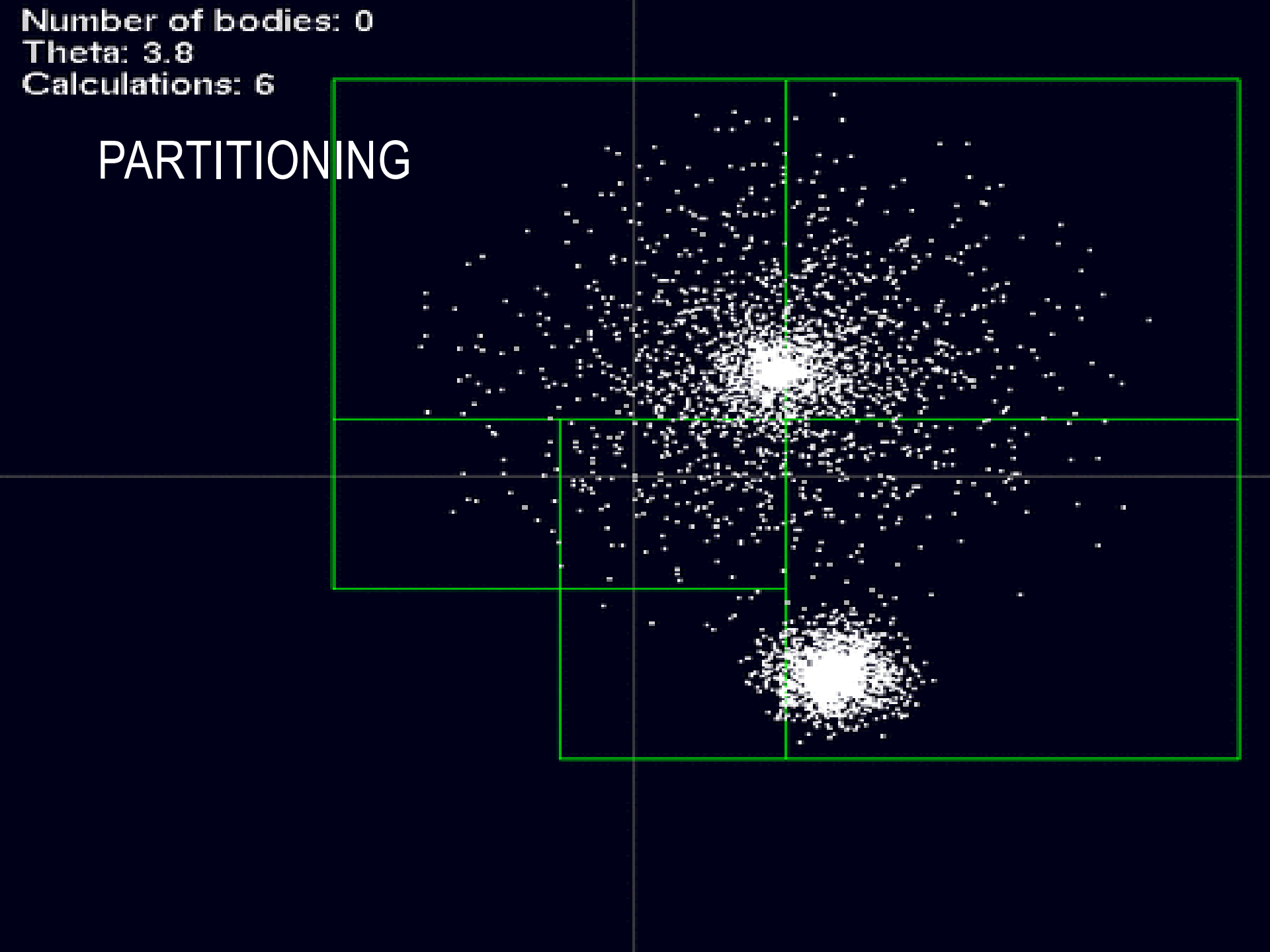


BARNES-HUT-TREE ALGORITHM

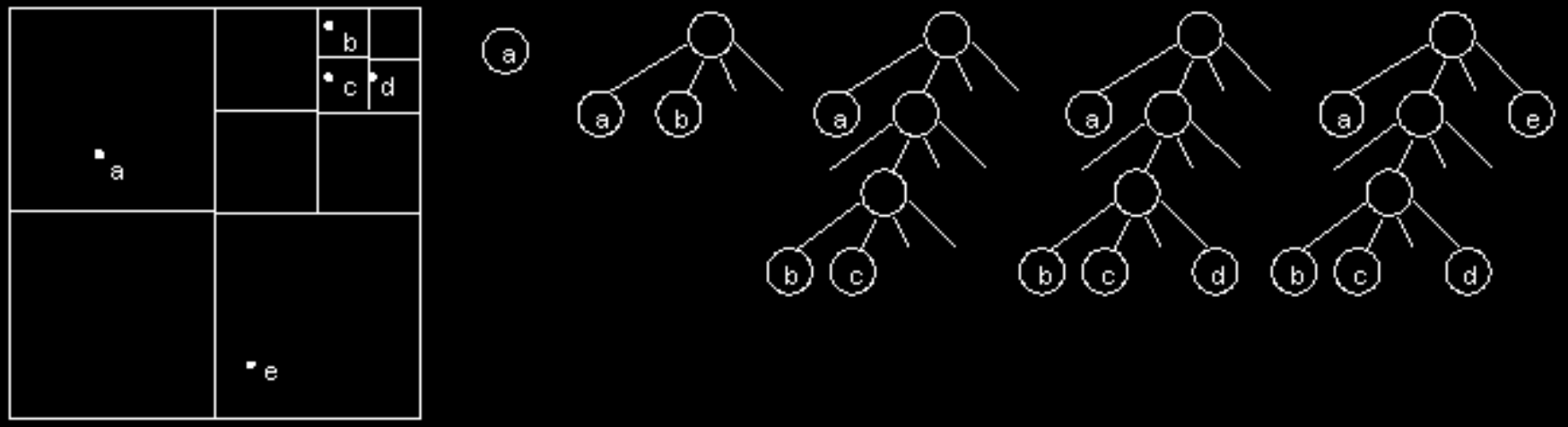
- Applying this fundamental insight, Barnes and Hut proposed a faster N-Body algorithm involving computations of O(N log N). The approximation made can be expressed as :

$$\sum_j \frac{Gm_j \vec{d}_{ij}}{\left|d_{ij}\right|^3} \approx \frac{GM\vec{d}_{i,cm}}{d^3{}_{i,cm}}$$

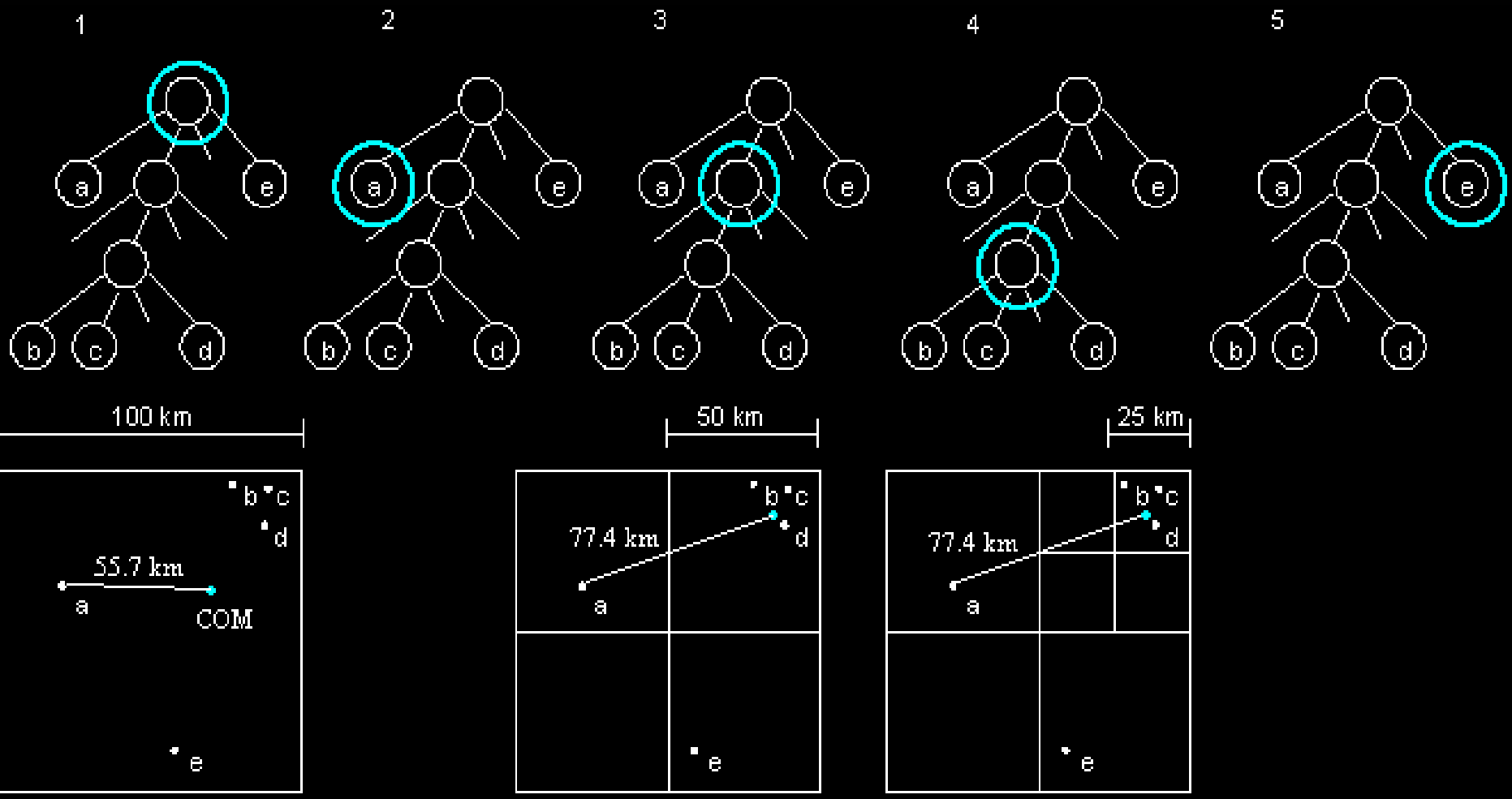
- 2 steps of implementation:
  - Tree Construction
  - Force Calculation


Number of bodies: 0
Theta: 3.8
Calculations: 6
PARTITIONING

TREE CONSTRUCTION



FORCE CALCULATION



PARALLELIZATION

- Tree Construction : Sequential

  The tree construction is done by the master process. After every force calculation step the tree construction happens again.

- Force Calculation : Parallel

  The force calculation has to be divided among different processing elements. This division of computation should make sure that each node gets equal amount of work. This will be made sure by assigning equal number of nodes to the available processing elements. Although this does not guarantee equal distribution of computation among the processing elements, but it is required to get the correct result.

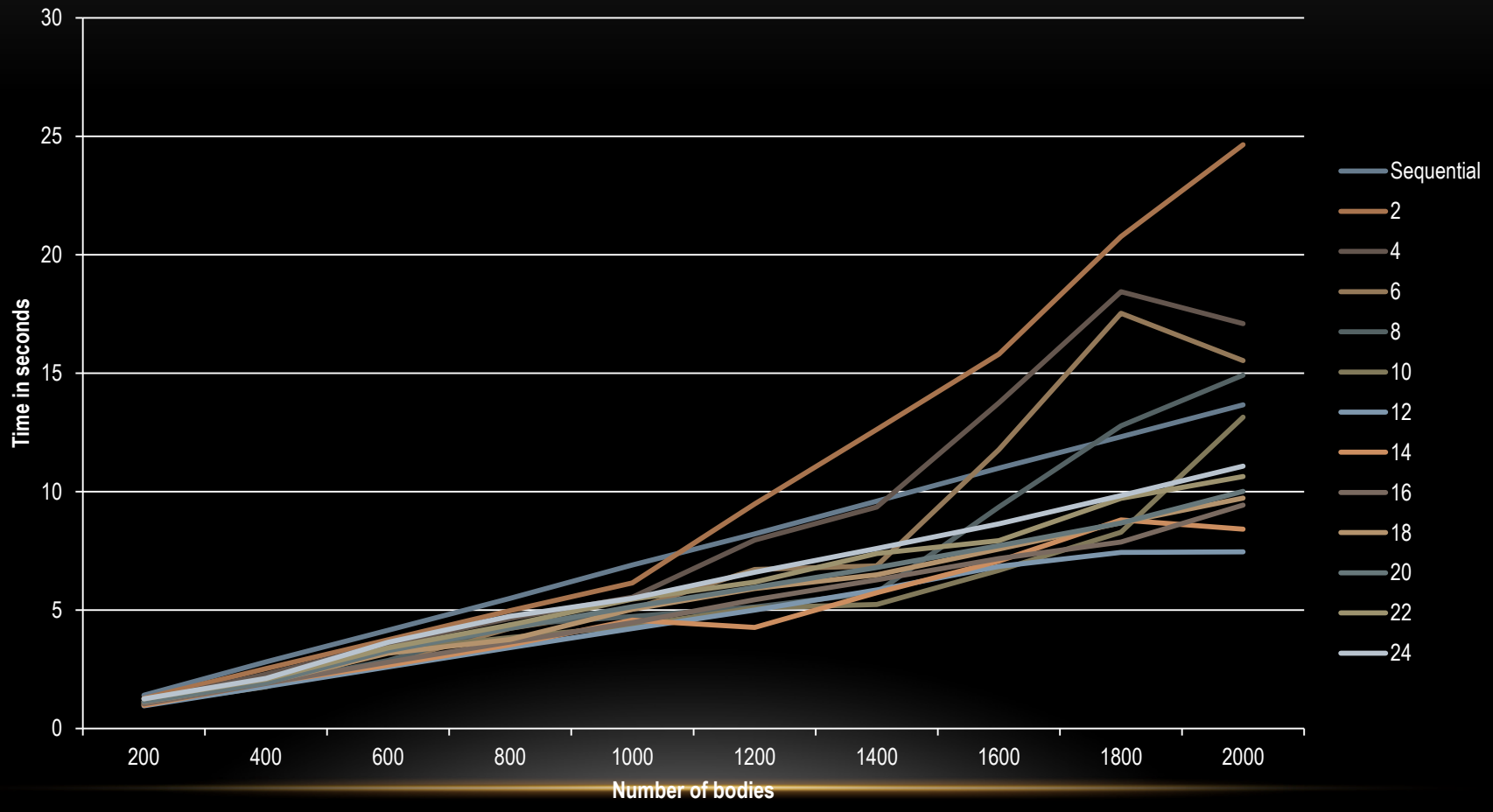
IMPLEMENTATION & TESTING DETAILS

- The code is implemented using C++.
- The parallelization is achieved using MPI.
- Testing of correctness is done using a simple body-to-body force calculation implementation in Matlab.
- Number of processors : 1 – 12 processors with 2 – 12 cores each
- Size of test data : 200, 400, 600, 800, 1000, 1200, 1400, 1600, 1800, 2000
- Objective : To find the new position and new velocity of the bodies after 1000 seconds.
- Number of runs for each processor-core-data : 2 – 5

TEST RESULT (USING ONLY 2 CORE PROCESSORS)

Number of cores used	Number of bodies in the system									
	200	400	600	800	1000	1200	1400	1600	1800	2000
Sequential	1.407775	2.808078	4.146703	5.492879	6.901948	8.205867	9.588082	10.99016	12.31352	13.65686
2	1.276178	2.523217	3.742116	4.973791	6.145523	9.478352	12.61787	15.79857	20.76073	24.6472
4	1.143109	2.286766	3.565634	4.639205	5.53449	7.949234	9.345573	13.74467	18.43838	17.08232
6	1.041932	1.872796	2.795644	4.226675	5.026368	6.708911	6.857607	11.76521	17.52122	15.52658
8	1.051868	1.868747	2.870389	4.375543	4.720748	5.120825	5.739536	9.351284	12.77664	14.91671
10	1.011577	1.754942	3.281019	3.834161	4.325293	5.094597	5.226812	6.678875	8.28365	13.13418
12	0.956534	1.773133	2.599147	3.412487	4.213007	4.999543	5.842502	6.834224	7.429297	7.45747
14	0.980419	1.919019	2.664115	3.529601	4.566682	4.267481	5.733323	7.068776	8.796899	8.408275
16	1.030482	1.898737	2.796701	3.652319	4.465114	5.430321	6.273997	7.164943	7.868132	9.430788
18	1.120092	1.948308	3.173143	3.748781	5.032132	5.906262	6.486563	7.577517	8.682746	9.725414
20	1.124623	1.926533	3.285092	4.225042	5.134918	5.950564	6.794716	7.711284	8.671826	10.01314
22	1.246033	2.078045	3.404853	4.366002	5.451053	6.177333	7.386048	7.932652	9.71643	10.63003
24	1.250754	2.108383	3.640148	4.733134	5.488904	6.593642	7.594441	8.633278	9.830037	11.06872

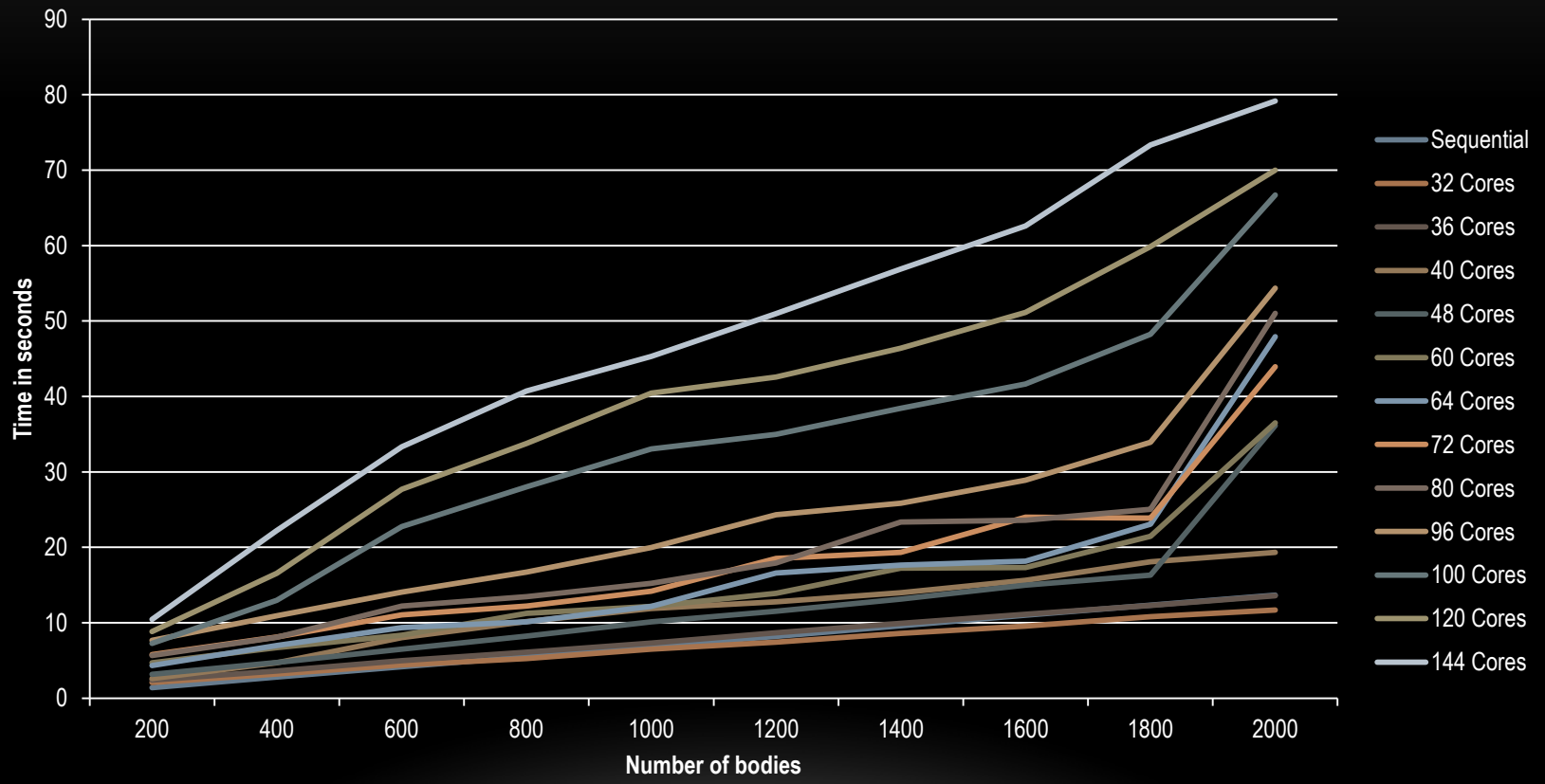
PERFORMANCE (2 – 24 CORES)



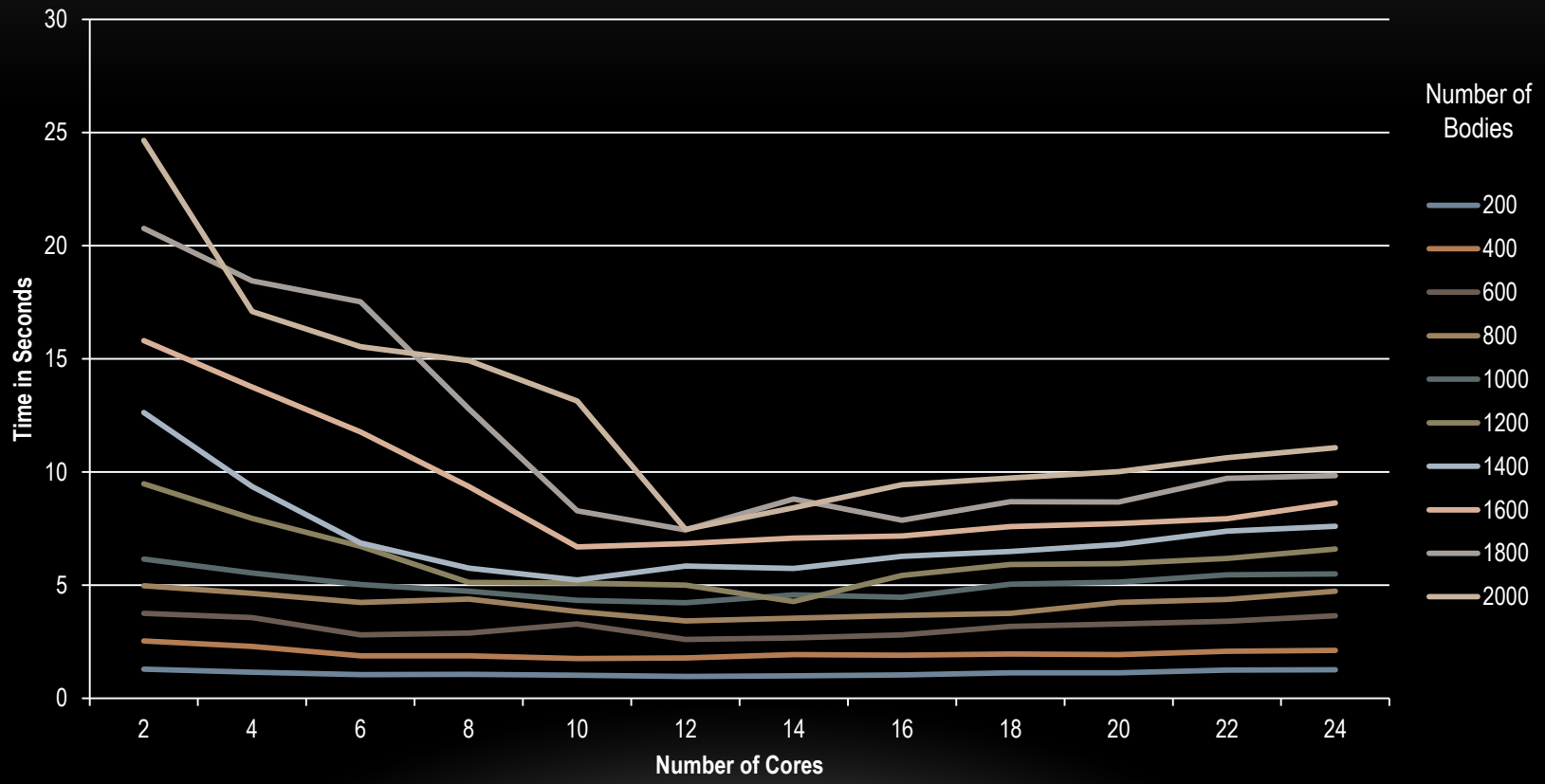
TEST RESULT CONT…
(USING 2 – 12 CORE PROCESSORS)

Number of cores used	Number of bodies in the system									
	200	400	600	800	1000	1200	1400	1600	1800	2000
Sequential	1.407775	2.808078	4.146703	5.492879	6.901948	8.205867	9.588082	10.99016	12.31352	13.65686
32	2.085415	3.194262	4.407054	5.272604	6.498258	7.416389	8.597305	9.575059	10.79097	11.67462
36	2.383148	3.584971	4.97983	6.087345	7.285348	8.664208	9.898121	11.13336	12.28171	13.55981
40	2.614461	4.67082	8.041525	10.16314	11.90424	12.7907	13.9767	15.62855	18.08354	19.33125
48	3.16292	4.740578	6.496666	8.233161	10.11402	11.52716	13.14517	14.97198	16.31785	36.12609
60	4.67569	6.706849	8.346394	11.25229	12.15455	13.89799	17.25654	17.32813	21.4477	36.49074
64	4.346687	6.996563	9.373251	10.11677	12.16597	16.60774	17.6199	18.16131	23.08055	47.91956
72	5.77313	8.108859	11.0614	12.22787	14.19273	18.53947	19.32937	23.98289	23.89063	43.95134
80	5.655985	8.070414	12.22401	13.43794	15.21979	17.91896	23.35271	23.58371	25.02088	51.00766
96	7.679689	10.93534	14.0798	16.73281	19.98828	24.32782	25.83515	28.91602	33.91711	54.35409
100	7.258991	12.95844	22.73295	28.00667	33.02196	34.97957	38.43564	41.63451	48.23355	66.68993
120	8.839344	16.55249	27.69451	33.757	40.45978	42.57086	46.39405	51.14697	59.87619	69.98934
144	10.45524	22.24125	33.34236	40.70483	45.32077	50.99073	56.8834	62.58797	73.34034	79.18707

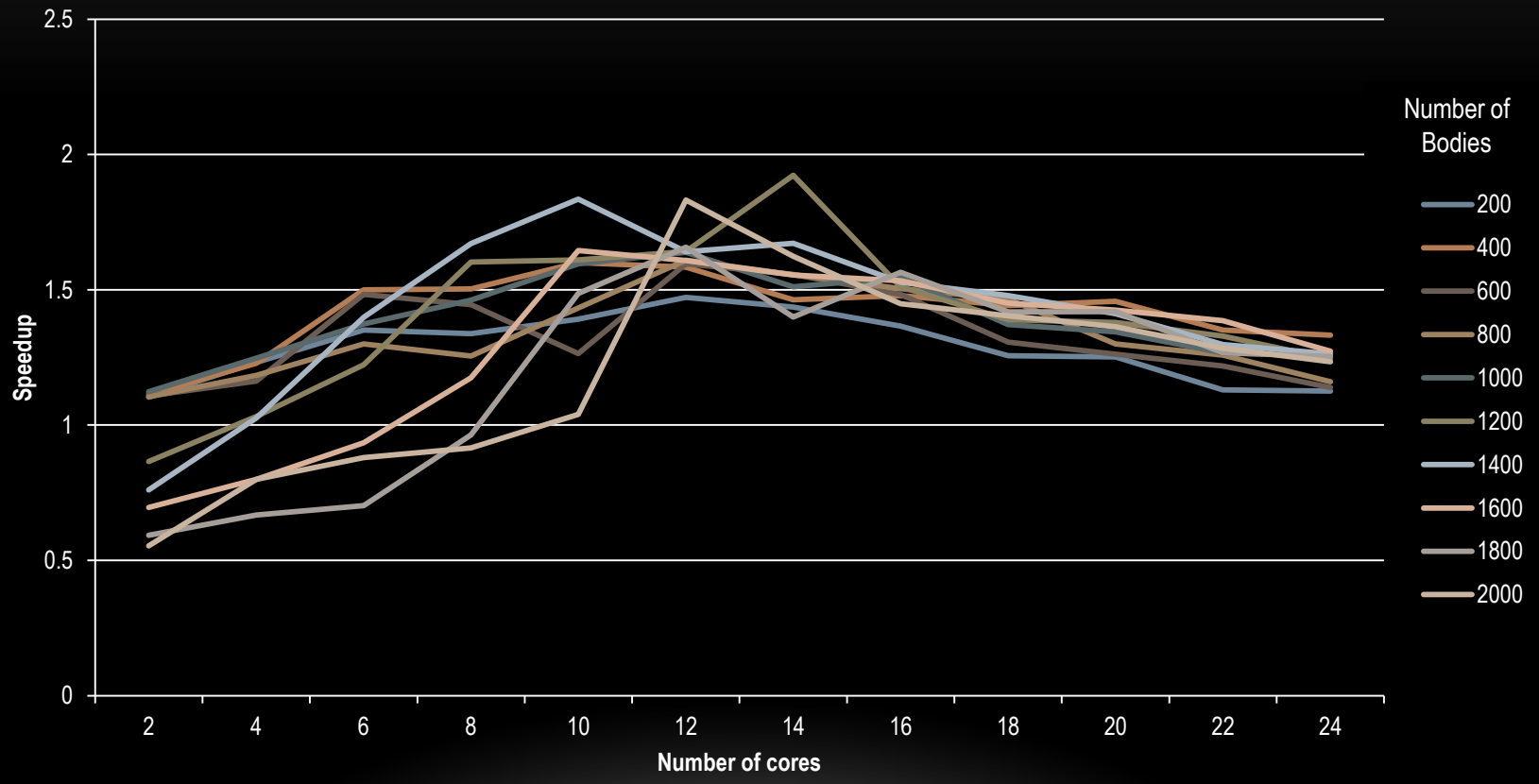
PERFORMANCE (32 – 144 CORES)



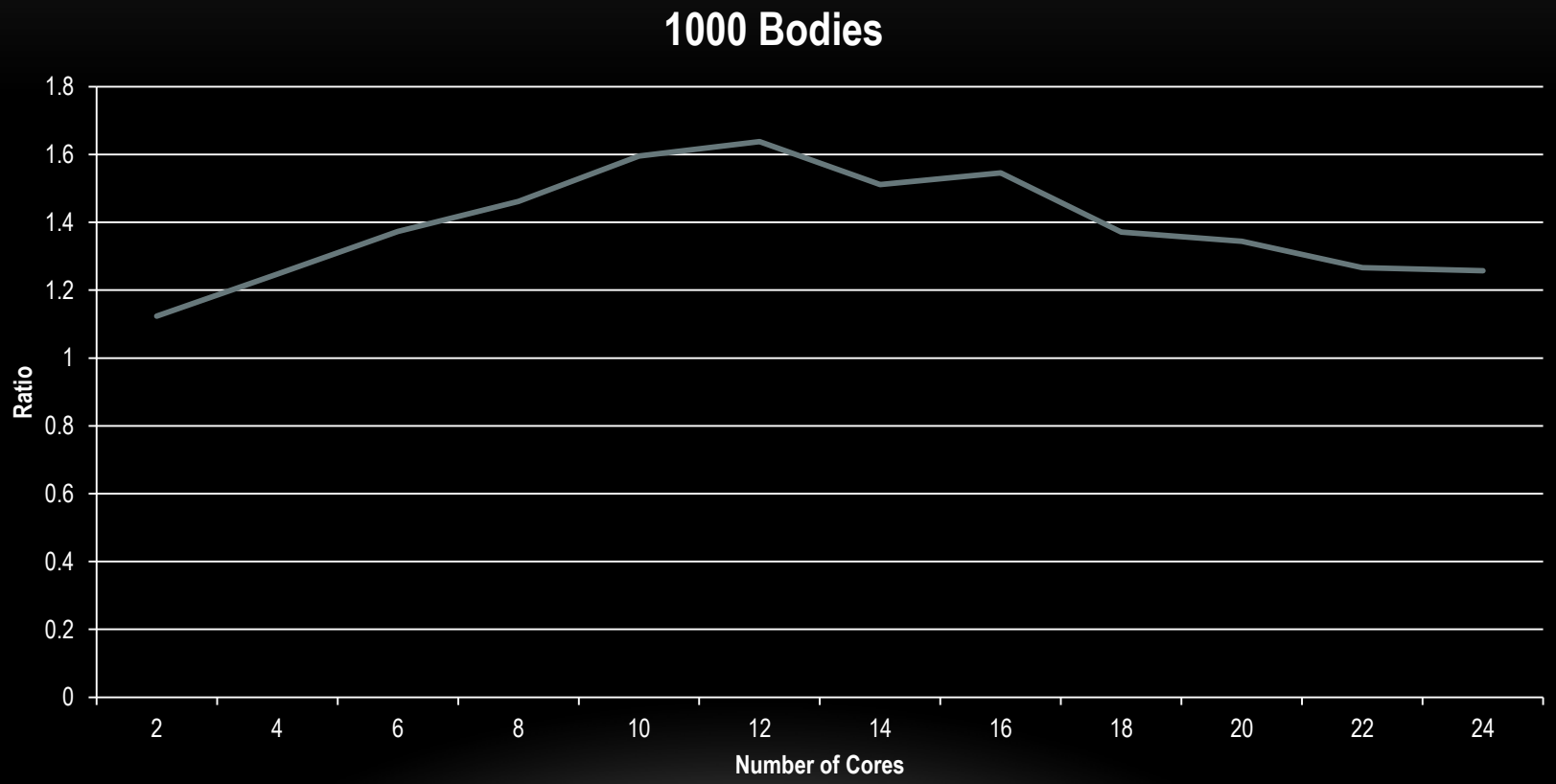
TIME VS CORES



SPEEDUP


2.5
2
1.5
1
0.5
0
Speedup
2
4
6
8
10
12
14
16
18
20
22
24
Number of cores
Number of Bodies
200
400
600
800
1000
1200
1400
1600
1800
2000

SPEEDUP FOR 1000 BODIES



FINDINGS / OUTCOMES

- The best result of parallelization is achieved with 8 – 12 cores.
- The performance improvement with parallelization becomes more conspicuous with higher number of processing elements.
- For the computation with more than 12 cores, the overhead of message passing takes over the computability offered by the processing elements.

OBSERVATIONS

- The Barnes Hut Tree algorithm is definitely an improvement over the All-Pairs Calculation method, but due to the creation of very deep tree, it requires significantly high amount of memory.
- For any system with bodies more than 2000, it gave following error:
  - terminate called after throwing an instance of 'std::bad_alloc'
- The computation offered by processing elements gets tough competition with the overhead of message passing between each other.

REFERENCES

- Experiences with Parallel N-Body Simulation By Pangfeng Liu and Sandeep Bhatt.
- Comparison of Parallel Solutions to N-Body Problems by Seong-Gi SEO
- Scalable Parallel Formulations of the Barnes-Hut Method for n-Body Simulations by Ananth Y. Grama, Vipin Kumar, and Ahmed Sameh