PARALLEL N-BODY SIMULATION

Pratik Chavan

CSE- 633 - Parallel Algorithms

Mentor: Dr. Russ Miller

University at Buffalo The State University of New York

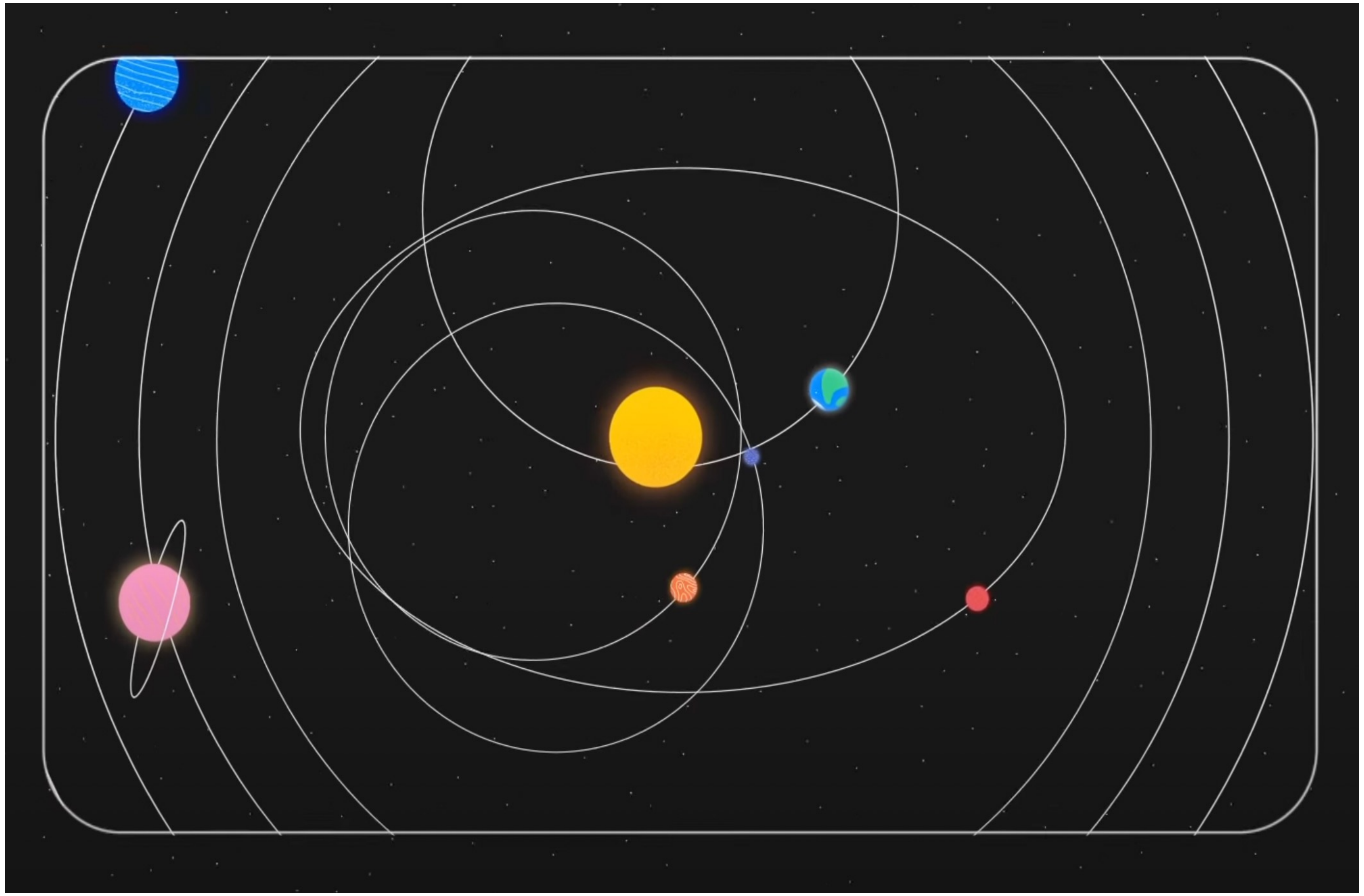
What is it?

N-body problem is a scientific problem in which we try to determine the motions of a group of celestial bodies or objects interacting with each other gravitationally.

Problem Statement

Given n number of bodies with mass, initial position and velocity for each, we predict their interactive forces, and consequently, predict their true orbital motions for all future times.

Why?



1. Solving this problem is interesting and challenging if …

2. Calculating for n-bodies can get chaotic, why?

3. Simulation has always been the way!

4. Parallelizing can help, how?

Let's start from the basics

- **Given:** n bodies, masses $[m_1, m_2, \dots m_n]$, initial positions $[p_1, p_2, \dots p_n]$ and velocity $[v_1, v_2, \dots v_n]$
- **Calculate** the acceleration by summing up the Gravitational forces on a particular body/particle by using;

$$\mathbf{F}_{ij} = \frac{G m_i m_j}{\left\|\mathbf{q}_j - \mathbf{q}_i\right\|^2} \cdot \frac{\left(\mathbf{q}_j - \mathbf{q}_i\right)}{\left\|\mathbf{q}_j - \mathbf{q}_i\right\|} = \frac{G m_i m_j \left(\mathbf{q}_j - \mathbf{q}_i\right)}{\left\|\mathbf{q}_j - \mathbf{q}_i\right\|^3},$$

- Now, after each time step t, we calculate the displacement and the final velocity

$$t + {}_i^1 p - {}_i^t p = \triangle x = {}_i^t v \triangle t + \frac{1}{2} {}_i^t a \triangle t^2$$

$$t + {}_i^1 v - {}_i^t v = \triangle v = {}_i^t a \triangle t$$

Parallel Solution

- For a given time t
- The master core reads input data and broadcasts using MPI_Bcast to all the child processors.
- Each child processor receives the data { mass, position and velocity } and works on the updating of the position and velocity.
- Each child processor will also collect data from other bodies to calculate the force exerted after each time step t
- We repeat this process for time t

Runtime: O$(n^2.\, i/p)$ $where\ n = no.of\ bodies\ i = no.of\ iterations, p = no.of\ processing\ elements$

If it's still O(n^2) then why parallelize? Remember Gustafson's Law

Programming Layout

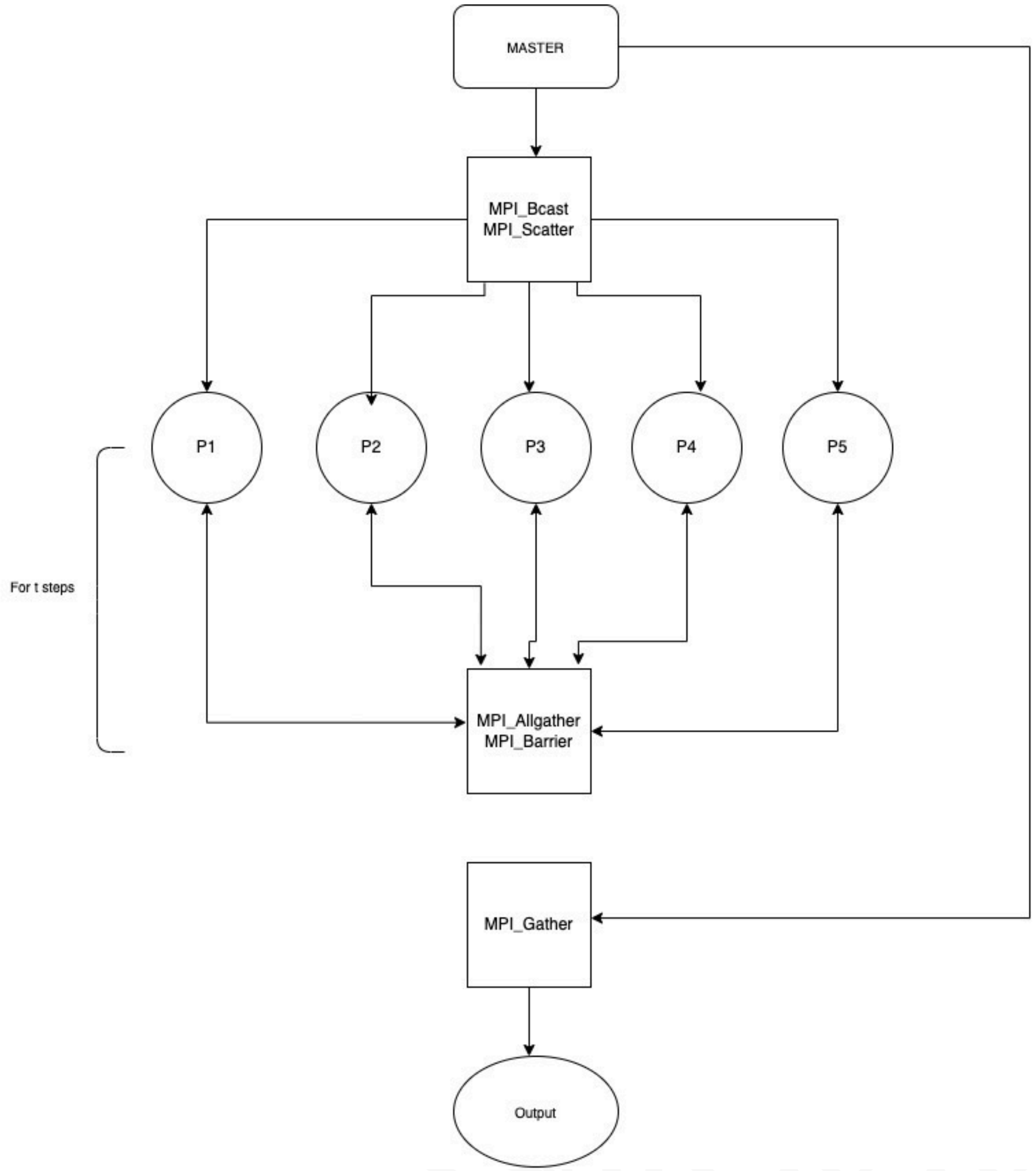
- Generate random mass, velocity and position vectors and initialize the first input structure as:

  [Mass, (x_coordinate, y_coordinate), (velocity_x, velocity_y)]

- Divide n bodies/particles across p processors, making each processor work on each body independently using MPI_Scatter.
- After calculating the displacement at time t, this output as given as input to other processors via MPI_Allgather.
- The same process of calculating the acceleration and displacement is repeated for time t.

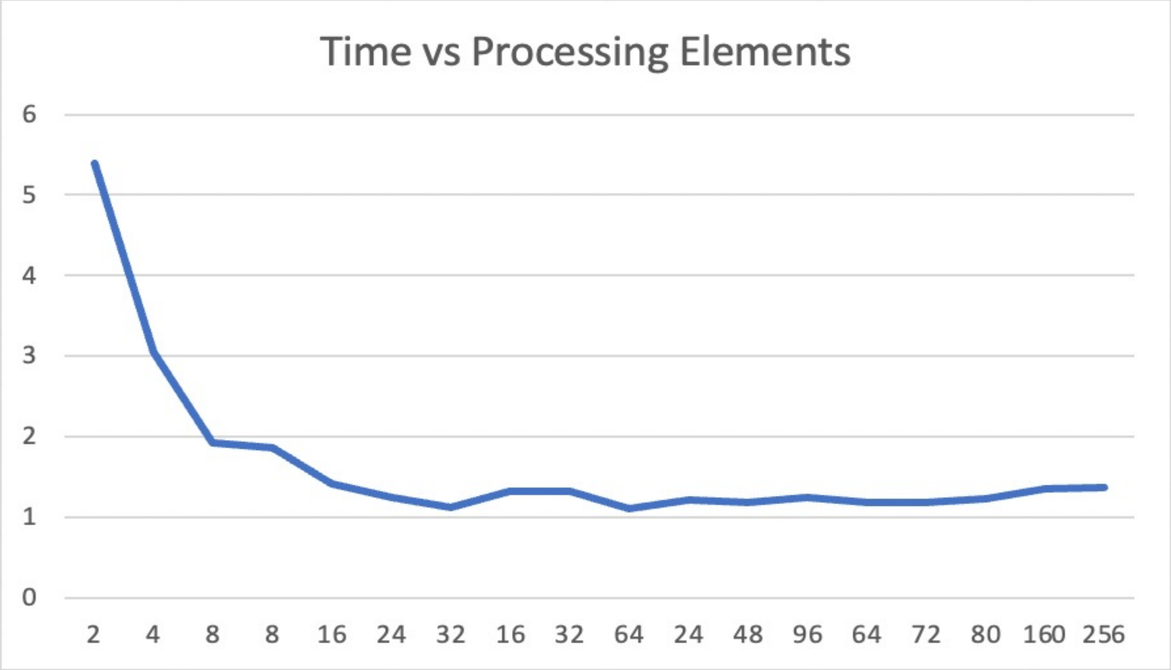
Process Layout

- The master reads the initial input data and broadcasts to all the the child processors
- Using MPI_Scatter the data is divided amongst the child processors.
- Then, for t-steps i.e., iteration the child processors do the computation and send the data to other processors using MPI_Allgather
- The process is made to wait until all other processors call MPI_Allgather using MPI_Barrier.
- Finally, the master collects the data by calling MPI_Gather and outputs it into the output file.



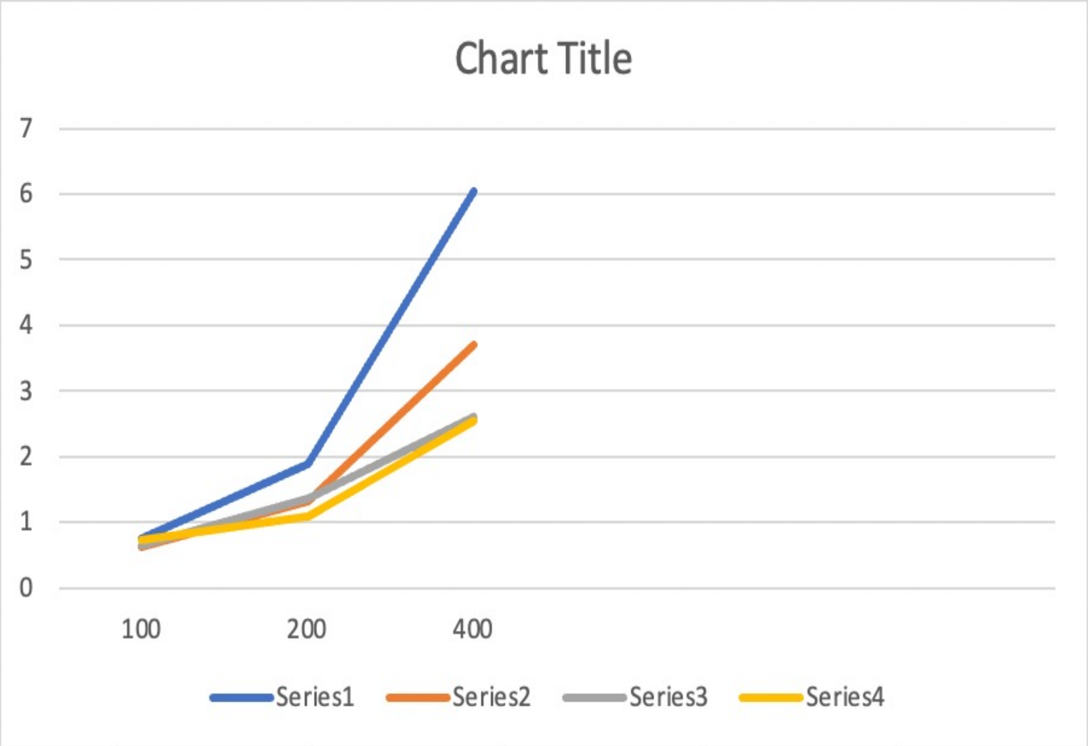
Results for fixed n values

Iteration = 5000	Particles = 200		
Node	Cores	Processing Elem	Time
1	2	2	5.39
2	2	4	3.05
2	4	8	1.92
4	2	8	1.87
4	4	16	1.41
4	6	24	1.24
4	8	32	1.12
8	2	16	1.33
8	4	32	1.33
8	8	64	1.11
12	2	24	1.22
12	4	48	1.19
12	8	96	1.24
16	4	64	1.18
18	4	72	1.19
20	4	80	1.23
20	8	160	1.36
32	8	256	1.37



Results for varying problem size

Iterations = 5000				
Particles	Nodes	Cores	Time	Processing Elem
100	2	4	0.75	8
200	2	4	1.88	8
400	2	4	6.05	8
100	4	4	0.62	16
200	4	4	1.32	16
400	4	4	3.7	16
100	8	4	0.64	32
200	8	4	1.37	32
400	8	4	2.6	32
100	16	4	0.73	64
200	16	4	1.08	64
400	16	4	2.54	64



Conclusion

- We see a U curve formation for a fixed n and large number of processors because of the higher number of communication calls between processes i.e. Amdahl's law.
- For increase in number of particles and constant number of processing elements we see a increase in time.
- For higher problem sizes with a higher number of processing elements the solution scales very well i.e., the solution works efficiently for large number of particles.

References

- n-body problem - Wikipedia
- Newton’s three-body problem explained - Fabio Pacucci | TED-Ed

THANK YOU