N-BODY SIMULATION USING MPI

Abhyudaya Mourya

CSE 633 – Parallel Algorithms

Mentor: Dr. Russ Miller

University at Buffalo
Department of Computer Science and Engineering
School of Engineering and Applied Sciences

Problem Statement

"N-body problem is a scientific problem where given *n* bodies/particles in a system, with mass, initial position and initial velocity for each, we need to determine how they evolve over time under the mutual set of forces acting between them"

Quick Review: Basic Numerical (Serial) Solution

- **Input:** n bodies, masses $\{m_1, m_2, \ldots, m_n\}$, initial positions $\{ {}^0_1x, {}^0_2x, \ldots, {}^0_nx \}$ and initial velocities $\{ {}^0_1v, {}^0_2v, \ldots, {}^0_nv \}$
- Assuming gravitational forces to be considered, acceleration is given by:

$$\boldsymbol{a_i} = {}^{\boldsymbol{F_i}}\!/_{m_i} = \frac{\sum_k G m_i m_k \frac{(x_k - x_i)}{|x_k - x_i|^3}}{m_i} = \sum_k G m_k \frac{(x_k - x_i)}{|x_k - x_i|^3}$$

- For time step t, we now get:

$$ {}^{t+1}_i x - {}^t_i x = \Delta x = {}^t_i v \Delta t + \tfrac{1}{2} {}^t_i a \Delta t^2$$
$$ {}^{t+1}_i v - {}^t_i v = \Delta v = {}^t_i a \Delta t$$

Thus, implying an $O(n^2)$ run-time for each iteration of the problem

Parallelized Solution – The Theory

1. Master core reads input data and broadcasts to all PUs
2. Each PU is then responsible for position and velocity update of n/p particles
3. Each PU then collects data of other particles after time step to act as input for the next step (MPI_Allgather)
4. Repeat 1-to-3

- Runtime α $(\frac{n^2.i}{p})$; where:

  n = No. of particles

  i = No. of iterations

  p = No. of processing elements

Implementation Details

- All n particles divided across p processors, so that each processes n/p particles for their dynamics update
- Once updated, the new data is then sent out via MPI_Allgather() to all processors
- When assigning per processor data, we take ceiling of n/p and pad with empty values to simplify the MPI_Allgather() operation
- Input structure as below:

```
Mass, X_coordinate, Y_coordinate, Velocity_x, Velocity_y
```

- These are randomly generated using a separate simple python script

Challenges

- Designing & Coding
  - For p processors, there will be approximately p^2 inter-processor communication calls
  - Accuracy: in large computations with large quantity of floats, precision errors need to be tracked to avoid compounding over time
- Benchmarking
  - There are 4 variables for the input data viz. no. of particles (n), no. of iterations (i), no. of nodes (N) and no. of cores per nodes (c)
  - These need to be converted to standard n vs p formats for better comprehension
  - Further more, these presented a huge number of combinations in comparison to simpler n vs p problems

Assumptions & Solutions

- 2-D Domain Only

  For simplification in operations and visualization, use only 2-D coordinates for position and velocity

- Cyclic Boundaries

  Since over time the particles may float off to very long distances (and out-of-bound distance values) causing issues with visualization

- Benchmarking against 1-processor case for accuracy

  Consider base case runs and compare the final states of all runs to base case (states of all particles) for accuracy verification

Assumptions & Solutions

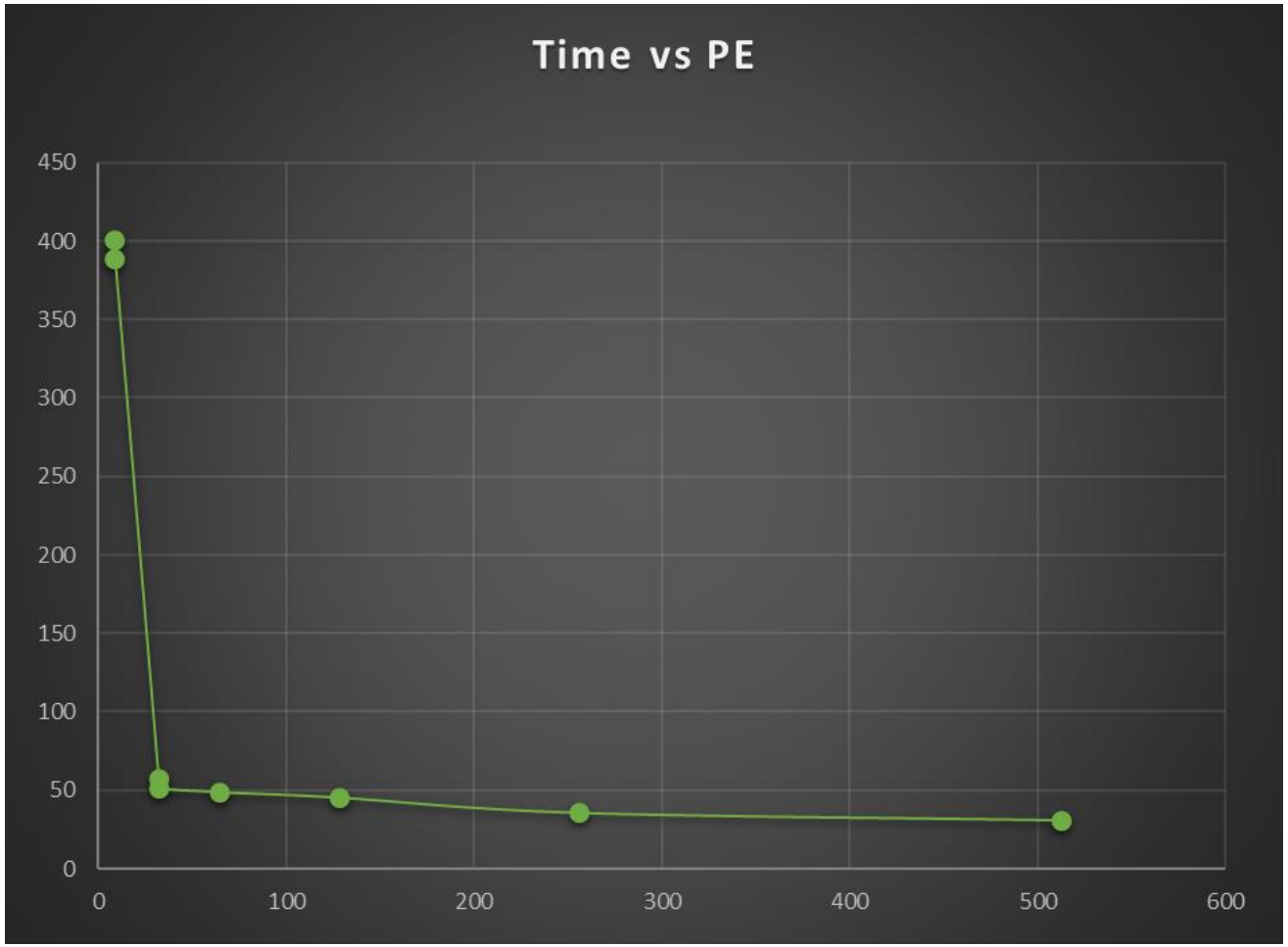
- Selecting correct set of data points to benchmark (and optimize the number of runs)

  Used Design-Expert tool for selection of optimum data points so as to optimize the combinations to be run

Results Classifications

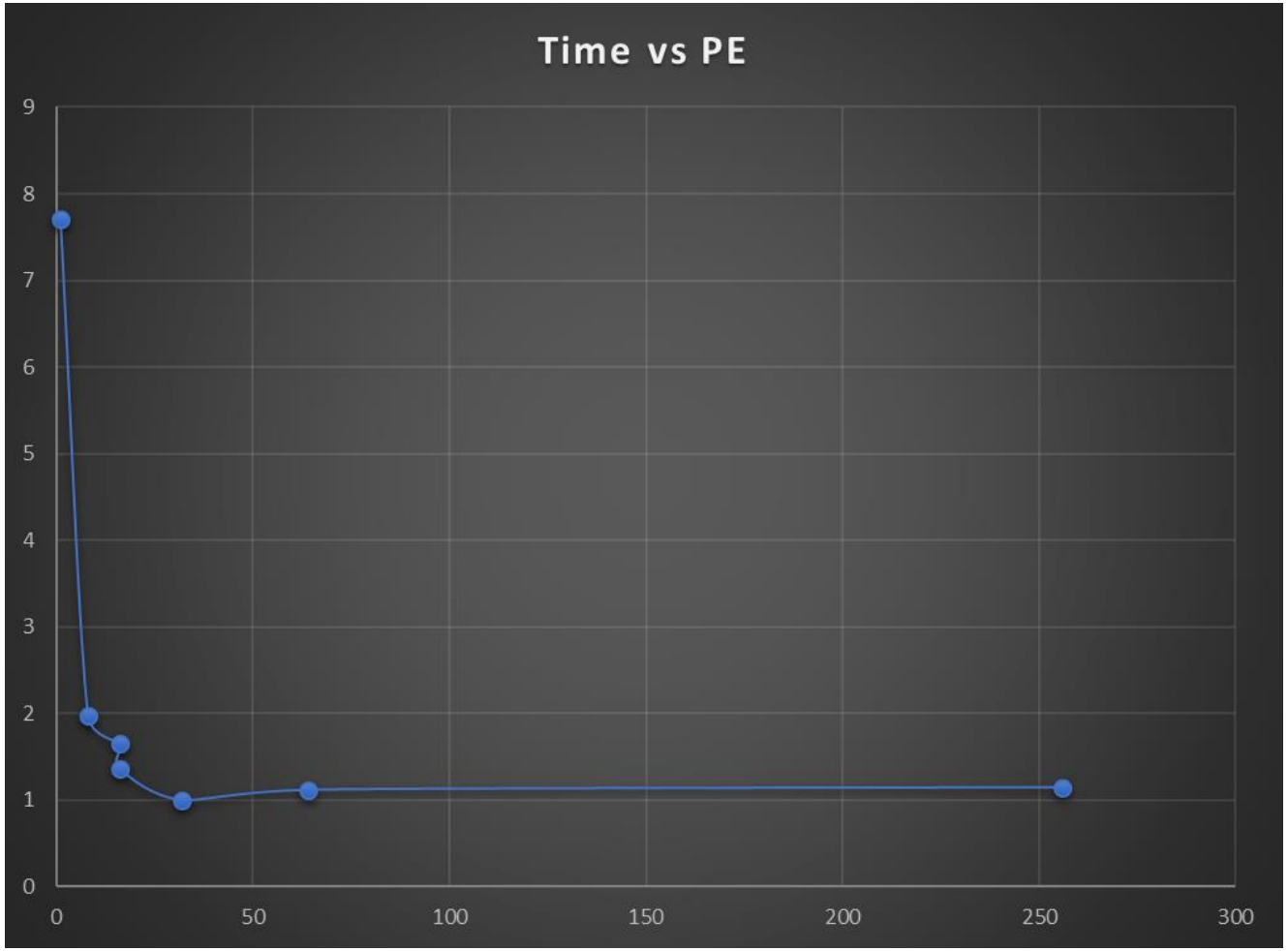
- Fixed data size results
- Fixed problem vs processor size results
- Cumulative results

Results for Fixed n-values (2000P 3000I)



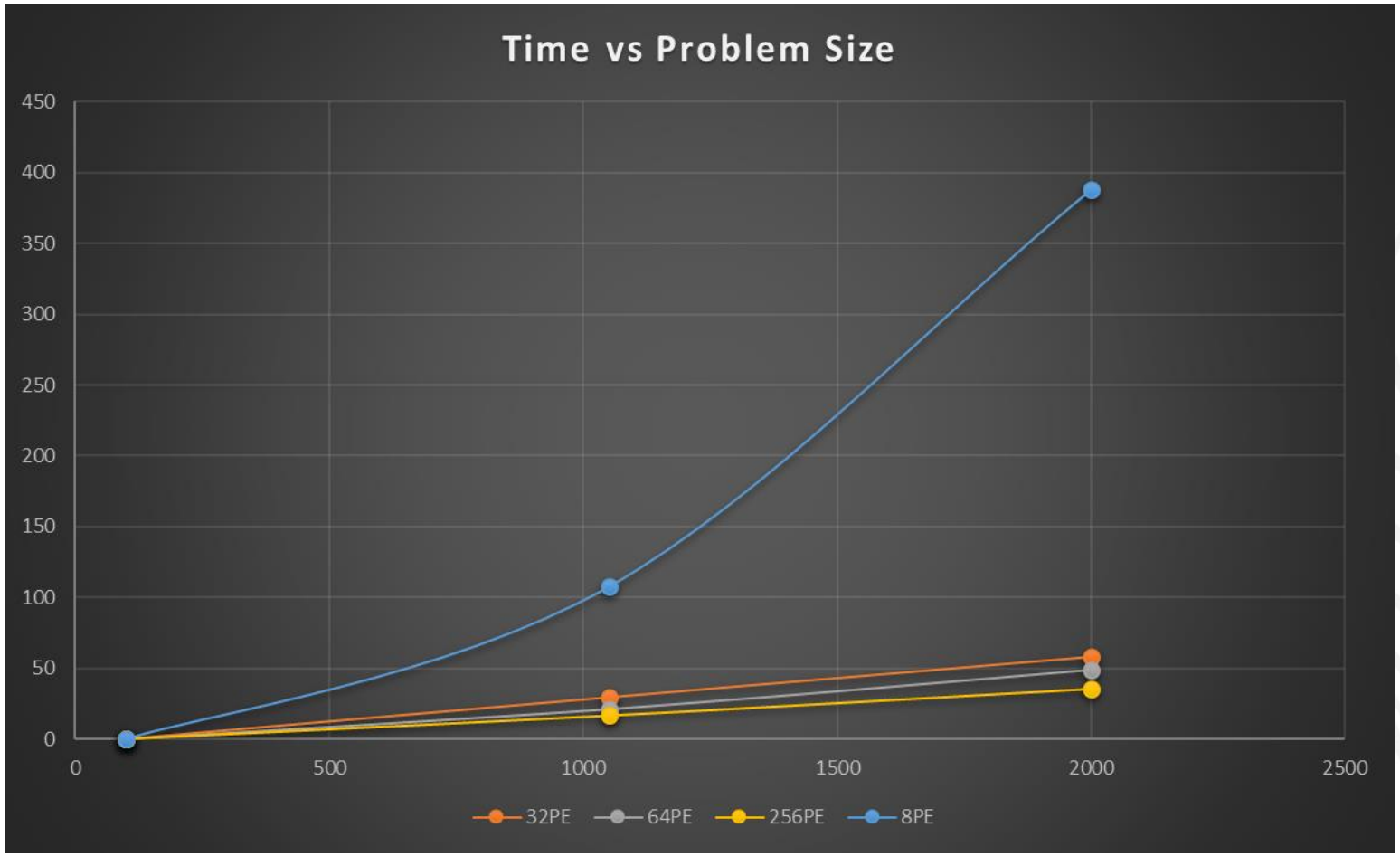
Nodes	Cores per Node	PE	Time
2	4	8	388.53
8	1	8	400.7
8	4	32	57.75
4	8	32	51.67
16	4	64	48.84
32	4	128	45.33
32	8	256	35.66
64	8	512	30.89

Results for Fixed n-values (100P 3000I)



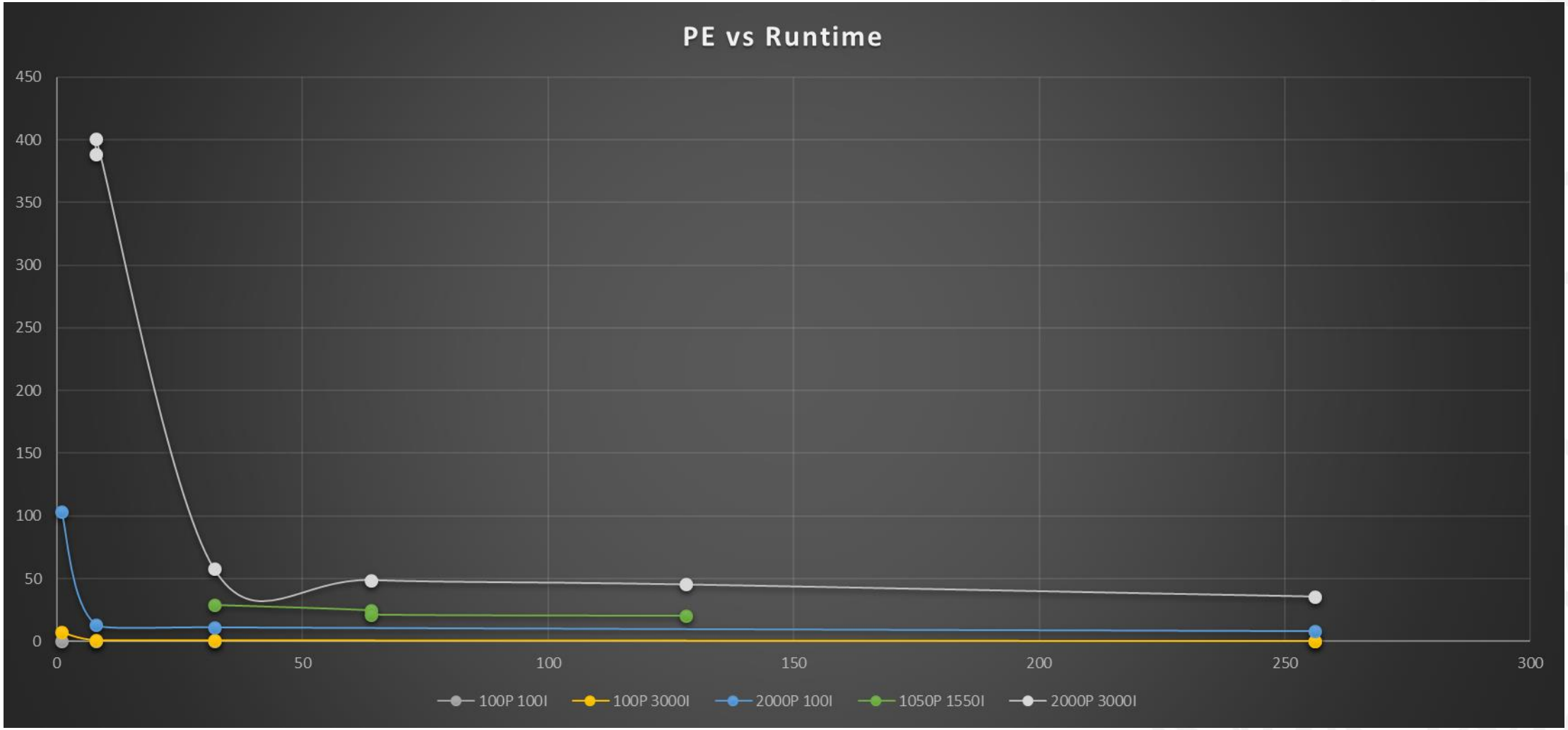
Nodes	Cores per Node	PE	Time
1	1	1	7.71
2	4	8	1.97
8	2	16	1.66
4	4	16	1.37
8	4	32	1
8	8	64	1.12
32	8	256	1.15

Results for Fixed p values



n	Nodes	Cores per Node	PE	Time
100	4	2	8	0.03
1050	4	2	8	107.63
2000	4	2	8	388.53
100	8	4	32	0.03
1050	8	4	32	29.34
2000	8	4	32	57.75
100	8	8	64	0.12
1050	8	8	64	21.15
2000	8	8	64	48.84
100	32	8	256	0.11
1050	32	8	256	16.73
2000	32	8	256	35.66

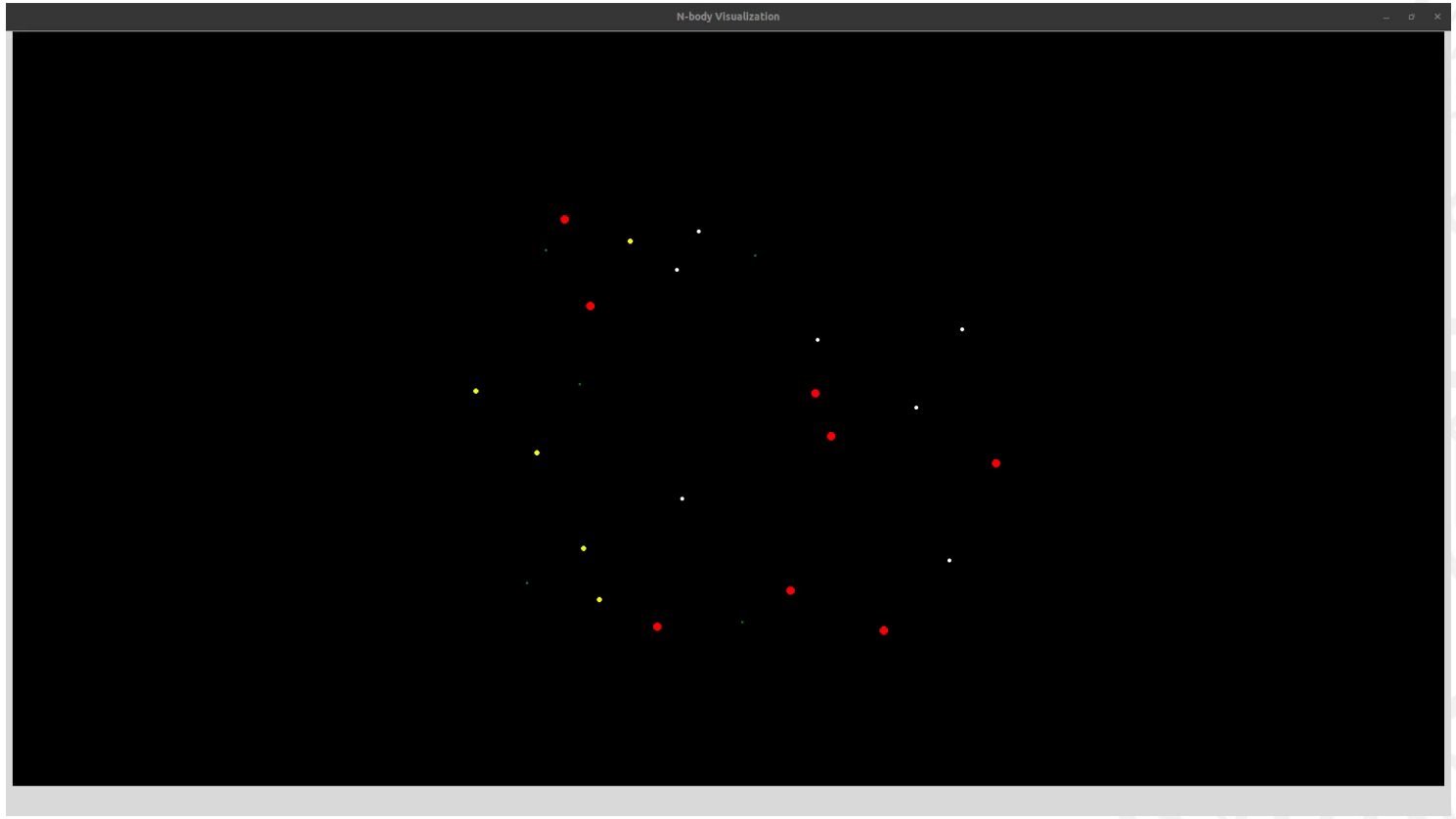
Cumulative Results



Inferences

- Based on Gustafson's law, true application of parallel processing where we solve "bigger" problems rather than solving problems "faster"
- For lower number of particles, we are able to see Amdahl's law being a blocker to performance because of the higher ratio of communication to processing
- For higher problem sizes with corresponding increase in processor size, the problem does indeed scale very well

(Not-so-Good!) Visualization



Note: PDF Version may have issues playing the video

References

- https://software.intel.com/en-us/download/intel-mpi-library-for-linux-os-developer-guide
- http://www.scholarpedia.org/article/N-body_simulations_(gravitational)
- https://en.wikipedia.org/wiki/N-body_problem
- Code will be hosted at: https://github.com/WhizK1D/cse633-mpi-nbody-simulation

QUESTIONS?

Thank you!