# SOLVING N-BODY PROBLEM USING PARALLEL APPROACH

Sakshi Singhal

CSE 633 – Parallel Algorithms

Guided By: Dr. Russ Miller

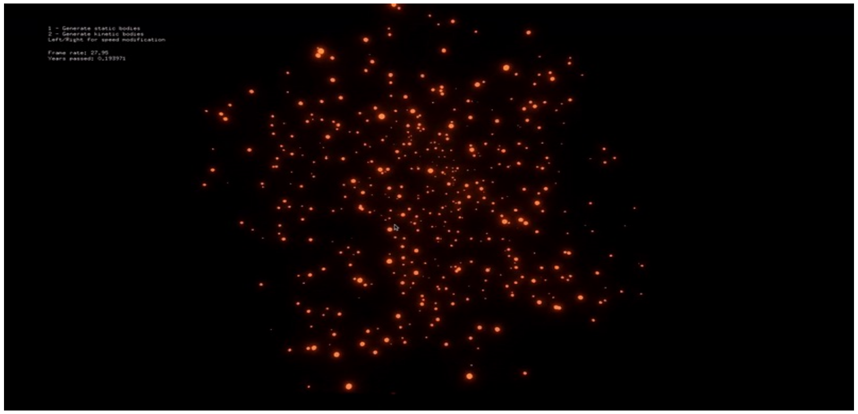
University at Buffalo The State University of New York

# Content

- Problem Statement
- How we can solve the problem sequentially
- What is the Parallel Approach to solve the problem
- Result/ Output of some experiments
- Conclusion & Future Scope

# Problem Statement

Initializing the random masses, velocities, positions of N particles we try to calculate forces present between them, as a result, their actual orbital movements for all possible periods after certain iterations.

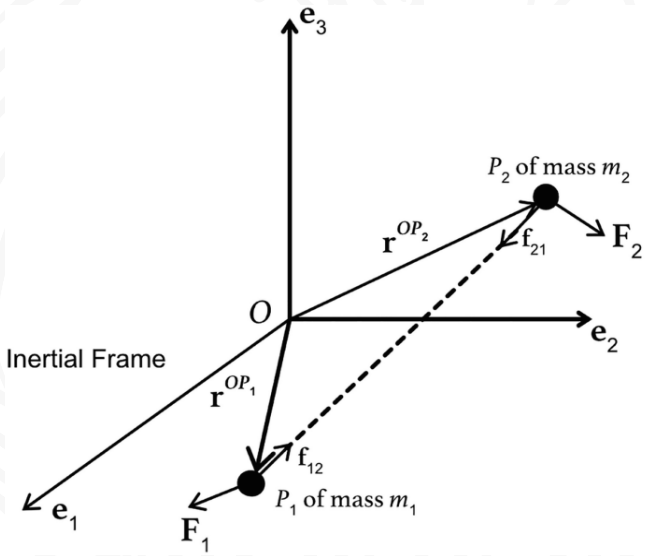


# Sequential Algorithm

1. Consider taking input as number of particles having masses $(m_1, m_2, m_3, \ldots\ldots, m_n)$, initial velocities $(v_1, v_2, v_3, \ldots\ldots, v_n)$ and their positions vector (that means in x and y coordinate $(p_1, p_2, p_3, \ldots\ldots, p_n)$
2. Newton's second law of motion states that mass times acceleration $m_i\, d^2\mathbf{q}_i/dt^2$ is equal to the sum of the forces on the mass.

But Newton's law of Gravity says that the gravitational force felt on mass

$m_i$ by a single mass $m_j$ is given by $F_{ij} = \frac{G m_i m_j}{||p_j - p_i||^2} \frac{(p_j - p_i)}{||p_j - p_i||} = \frac{G m_i m_j (p_j - p_i)}{||p_j - p_i||^3}$



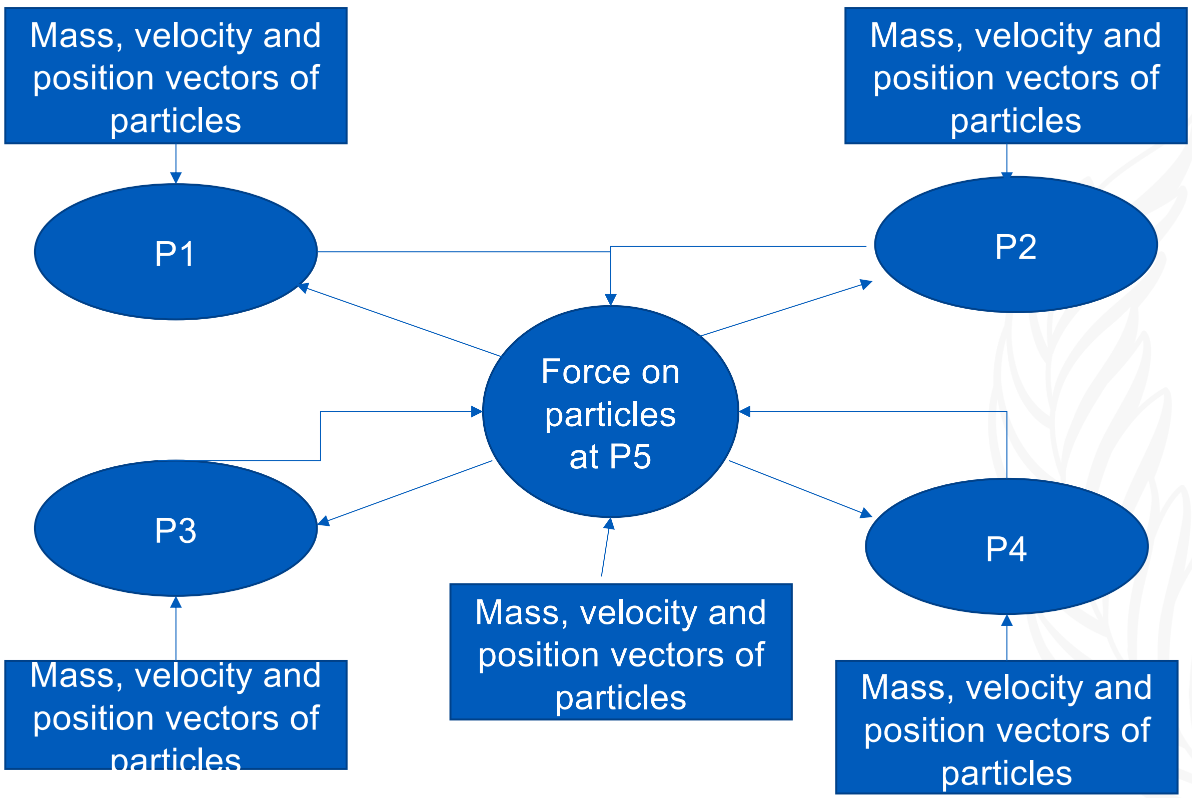

But to calculate force on n particle we need to calculate summation of forces of (n-1) particles on n and this will lead to time complexity of O($n^2$)

# Parallel Algorithm

- Defining the number of particles and evenly distributing it across processors.
- These particles have masses, velocity and position vector.
- Calculate force on each particle due to all other particles from both the directions.
- Once we have calculated force of each particle, we will send the data back to all the processor and update the position and velocity and then again calculate the force.
- Repeat this till time 't' iterations.
- Below are the main MPI Functions used

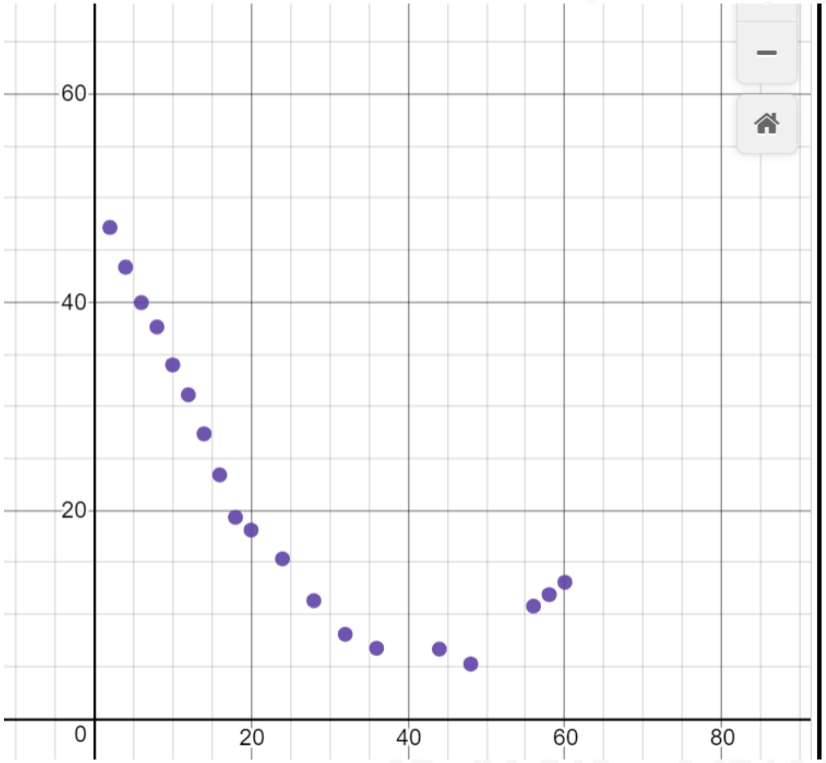
MPI_Bcast, MPI_Scatter, MPI_Barrier, MPI_Allgather, MPI_Gather

# How does it take place:



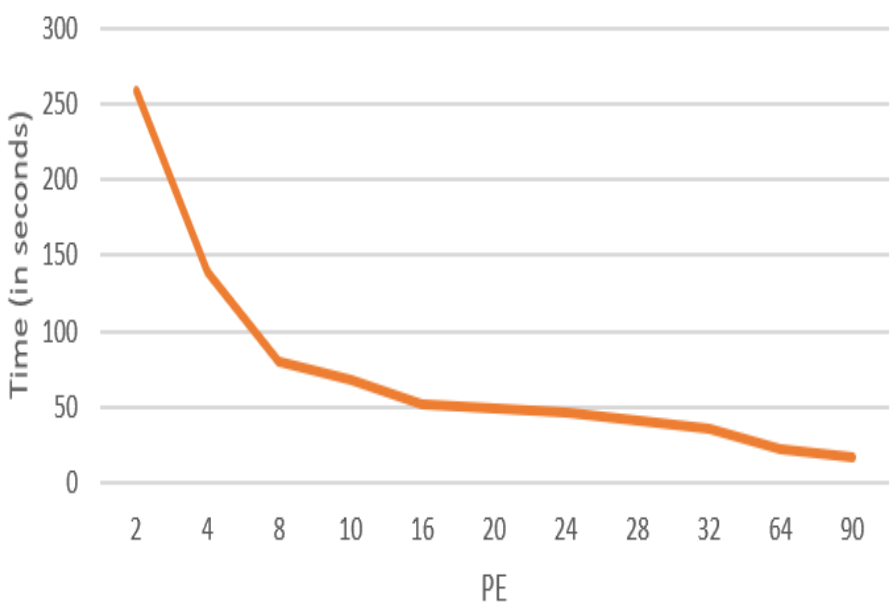
# Parallel execution during the midterm results

| particles=100, iterations=200 | |
|---|---|
| Processors | Time(seconds) |
| 2 | 47.17105 |
| 4 | 43.360829 |
| 6 | 39.96194 |
| 8 | 37.619051 |
| 10 | 33.976331 |
| 12 | 31.098934 |
| 14 | 27.348912 |
| 16 | 23.398761 |
| 18 | 19.329021 |
| 20 | 18.099923 |
| 24 | 15.340203 |
| 28 | 11.323444 |
| 32 | 8.0912884 |
| 36 | 6.756678 |
| 44 | 6.6655433 |
| 48 | 5.2356789 |
| 56 | 10.786663 |
| 58 | 11.8998877 |
| 60 | 13.09 |



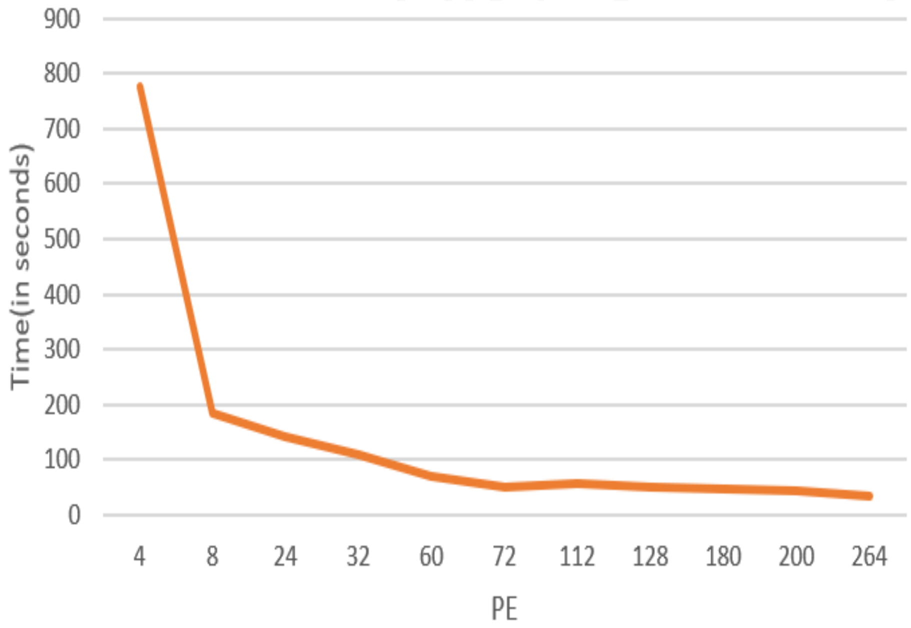
# Experiment on fixed number of Particles and Iteration on 1 core per node

| Particles=5000 | Iteration=7000 | | |
|---|---|---|---|
| Node | Core Per Node | Processing Element=(Node * Code Per Node) | Time (in seconds) |
| 2 | 1 | 2 | 257.9915 |
| 4 | 1 | 4 | 139.2165 |
| 8 | 1 | 8 | 79.84352 |
| 10 | 1 | 10 | 67.82787 |
| 16 | 1 | 16 | 51.89456 |
| 20 | 1 | 20 | 48.36529 |
| 24 | 1 | 24 | 46.45003 |
| 28 | 1 | 28 | 41.20975 |
| 32 | 1 | 32 | 36.11346 |
| 64 | 1 | 64 | 21.43282 |
| 90 | 1 | 90 | 16.23896 |



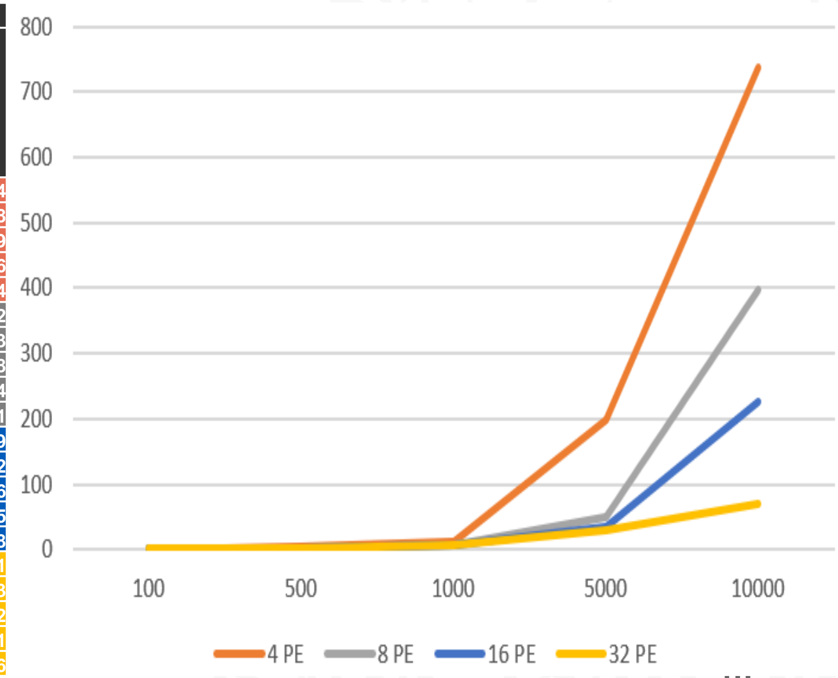
# Experiment on fixed number of Particles and Iteration on multiple core per node

| Particles=8000 | Iterations=7000 | | |
|---|---|---|---|
| Node | Core Per Node | Processing Element=(Node * Code Per Node) | Time (in seconds) |
| 2 | 2 | 4 | 775.8501 |
| 2 | 4 | 8 | 185.0405 |
| 4 | 6 | 24 | 140.3759 |
| 4 | 8 | 32 | 108.9103 |
| 6 | 10 | 60 | 70.82313 |
| 6 | 12 | 72 | 52.00095 |
| 8 | 14 | 112 | 58.2785 |
| 8 | 16 | 128 | 51.83277 |
| 10 | 18 | 180 | 48.39373 |
| 10 | 20 | 200 | 42.55528 |
| 12 | 22 | 264 | 33.81453 |



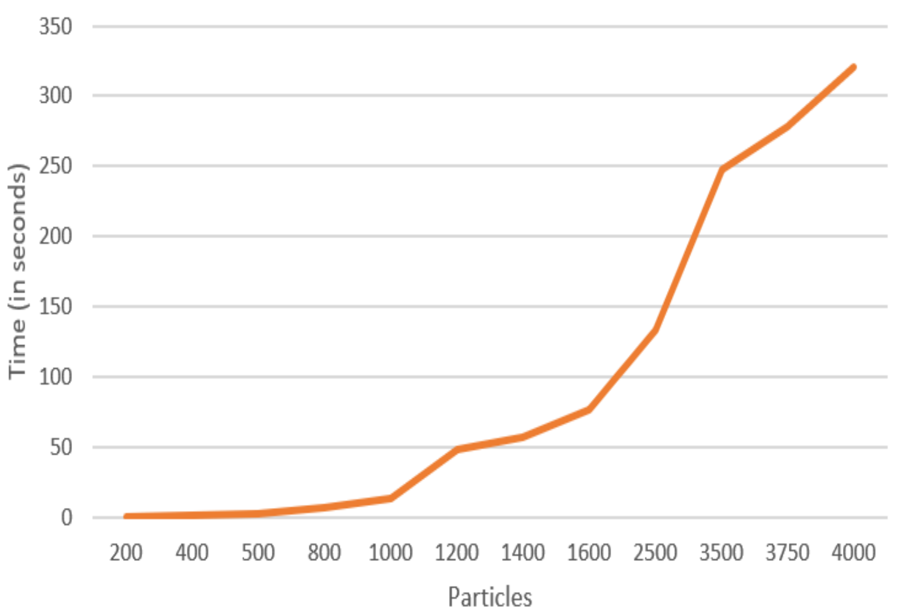
# Experiment to see increase in number of particles keeping PE constant

| Iteration=10000 | | | | |
|---|---|---|---|---|
| Particles | Node | Core Per Node | Processing Element=(Node * Code Per Node) | Time (in seconds) |
| 100 | 4 | 1 | 4 | 0.7074 |
| 500 | 4 | 1 | 4 | 4.67098 |
| 1000 | 4 | 1 | 4 | 12.8379 |
| 5000 | 4 | 1 | 4 | 198.786 |
| 10000 | 4 | 1 | 4 | 736.724 |
| 100 | 8 | 1 | 8 | 0.70442 |
| 500 | 8 | 1 | 8 | 2.91603 |
| 1000 | 8 | 1 | 8 | 6.04038 |
| 5000 | 8 | 1 | 8 | 50.6734 |
| 10000 | 8 | 1 | 8 | 397.281 |
| 100 | 16 | 1 | 16 | 0.8959 |
| 500 | 16 | 1 | 16 | 3.11082 |
| 1000 | 16 | 1 | 16 | 5.98086 |
| 5000 | 16 | 1 | 16 | 35.0035 |
| 10000 | 16 | 1 | 16 | 227.368 |
| 100 | 32 | 1 | 32 | 1.06811 |
| 500 | 32 | 1 | 32 | 3.2483 |
| 1000 | 32 | 1 | 32 | 6.082 |
| 5000 | 32 | 1 | 32 | 28.7501 |
| 10000 | 32 | 1 | 32 | 70.6746 |



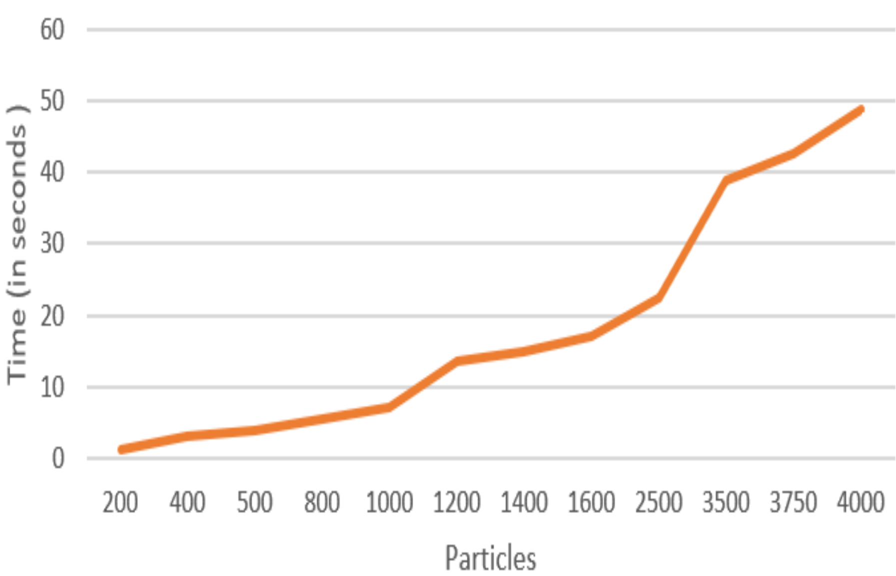
# Sequential Execution keeping Iteration constant

| Iteration=7000 | | | | |
|---|---|---|---|---|
| Particles | Nodes | Core Per Node | PE | Time |
| 200 | 1 | 1 | 1 | 0.677564 |
| 400 | 1 | 1 | 1 | 1.465859 |
| 500 | 1 | 1 | 1 | 2.218085 |
| 800 | 1 | 1 | 1 | 7.223454 |
| 1000 | 1 | 1 | 1 | 12.95424 |
| 1200 | 1 | 1 | 1 | 48.3492 |
| 1400 | 1 | 1 | 1 | 56.39432 |
| 1600 | 1 | 1 | 1 | 76.95455 |
| 2500 | 1 | 1 | 1 | 133.2046 |
| 3500 | 1 | 1 | 1 | 247.2937 |
| 3750 | 1 | 1 | 1 | 278.1287 |
| 4000 | 1 | 1 | 1 | 320.6927 |



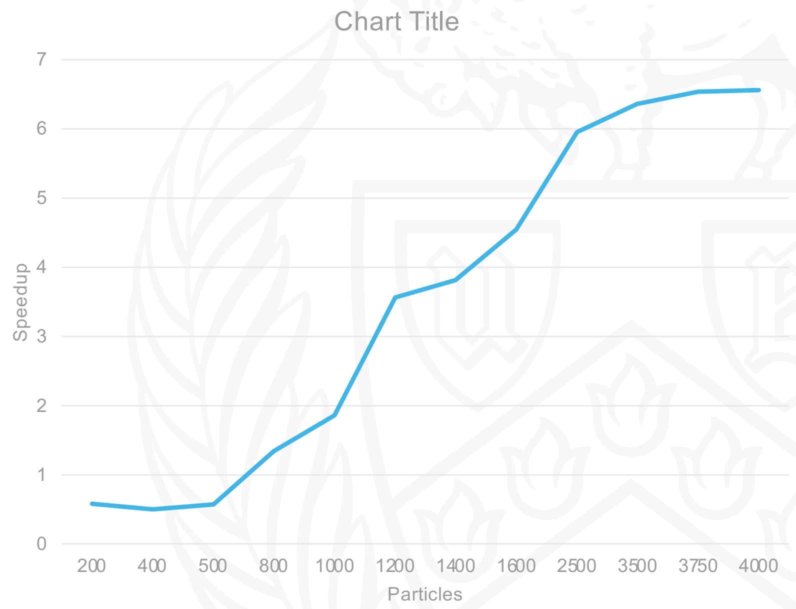
# Parallel Execution keeping Iteration constant and Data per PE constant

| Iteration = 7000 | | | | | |
|---|---|---|---|---|---|
| Particles | Nodes | Core Per Node | PE | Time | Data Per PE |
| 200 | 4 | 1 | 4 | 1.16823 | 50 |
| 400 | 8 | 1 | 8 | 2.942712 | 50 |
| 500 | 10 | 1 | 10 | 3.897655 | 50 |
| 800 | 16 | 1 | 16 | 5.400002 | 50 |
| 1000 | 20 | 1 | 20 | 6.988643 | 50 |
| 1200 | 24 | 1 | 24 | 13.58964 | 50 |
| 1400 | 28 | 1 | 28 | 14.78836 | 50 |
| 1600 | 32 | 1 | 32 | 16.92689 | 50 |
| 2500 | 50 | 1 | 50 | 22.37975 | 50 |
| 3500 | 70 | 1 | 70 | 38.8636 | 50 |
| 3750 | 75 | 1 | 75 | 42.56784 | 50 |
| 4000 | 80 | 1 | 80 | 48.87955 | 50 |



# Speedup

| Speedup=(tseq/tp) | | | | |
|---|---|---|---|---|
| Particles | Time(tseq) | Particles | Time(tp) | Speedup |
| 200 | 0.677564 | 200 | 1.16823 | 0.579992 |
| 400 | 1.465859 | 400 | 2.942712 | 0.498132 |
| 500 | 2.218085 | 500 | 3.897655 | 0.569082 |
| 800 | 7.223454 | 800 | 5.400002 | 1.337676 |
| 1000 | 12.95424 | 1000 | 6.988643 | 1.853614 |
| 1200 | 48.3492 | 1200 | 13.58964 | 3.557798 |
| 1400 | 56.39432 | 1400 | 14.78836 | 3.813427 |
| 1600 | 76.95455 | 1600 | 16.92689 | 4.546291 |
| 2500 | 133.2046 | 2500 | 22.37975 | 5.952015 |
| 3500 | 247.2937 | 3500 | 38.8636 | 6.363119 |
| 3750 | 278.1287 | 3750 | 42.56784 | 6.533775 |
| 4000 | 320.6927 | 4000 | 48.87955 | 6.560877 |



# Conclusion

- As per Amdahl's law as the number of communication between particles increase the time decreases and that is what took place in the first graph for fixed number of particles and increase in processors we get U curve.
- In sequential execution and parallel execution keeping the iterations constant and increasing the number of particles we see increase in time.
- As per the Gustafson's Law, When the number of particles increases but the number of processing elements stays constant, we observe an increase in computation time.
- Speedup reaches a saturation at around 2500 Particles.

# Future Work:

- Access nodes greater than 90 nodes with 1 core per node.
- Try implementing parallel approach using OpenMPI.

University at Buffalo The State University of New York

# References


- https://www.youtube.com/watch?v=vjUaNJqIWTs
- https://en.wikipedia.org/wiki/N-body_simulation
- https://curc.readthedocs.io/en/latest/programming/MPI-C.html
- http://www.cs.toronto.edu/~wayne/research/thesis/msc/node24.html
- http://www.dartmouth.edu/~rc/classes/intro_mpi/overview_parallel_prog.html#top
- https://gereshes.com/2018/05/07/what-is-the-n-body-problem/

University at Buffalo The State University of New York

# THANK YOU!
# ANY QUESTIONS?