Motion of N particles using MPI

HPC Project WS 18/19

Dude Durga Prasanth
Matr. No: 62752

Examiner
Prof. Dr. Oliver Rheinbach
Institut für Numerische Mathematik und Optimierung
TU Bergakadieme Freiberg

January,2020

Abstract: The aim of this Project is to simulate the motion of n particles, which interact through a gravitational force using appropriate masses and initial conditions and parallelize the code using MPI and to perform scalability tests on the TUBAF cluster.

Introduction: The objective of this project is Simulate motion of n particles in C, calculating the coordinates of the particles at each timestep and then parallelizing the program with MPI to get a speedup on the performance of the program. The algorithm employed here is $O(n^2)$ complexity. It means n particles will have n(n-1)/2 interactions. To perform scalability tests, different number of particle sizes and different time steps are chosen.
Furthermore, visualization of the movement of the particles is done in python scripts. This script reads the coordinates of every particle at each time step and creates image at each timestep and combined GIF file, which shows the movement of the n particles.

Algorithm: The algorithm implemented is $O(n^2)$
for each timestep:
for particles 1 to n:
update position of each particle:
x(t+1) = x(t) + v(t) *t + 0.5*a(t) *t*t
update acceleration of each particle:
a(t+1) += G*m[j]/ x(t+1) * xt+1)
update velocity of each particle:
v(t+1) = v(t) + 0.5*(a(t) + a(t+1))

Serial Code:
Serial code involves the implementation of the Algorithm by creating initial positions and initial velocities of the particles by using random numbers in c. A time analysis has been performed for different number of particles for a duration of 10s by varying the step size at which the coordinates of the each particle is evaluated. The updated coordinated are saved in a text file and are visualized using python code.

Time step	Computation time n=128	Computation time n=256	Computation time n=1024
0.5	0.0087	0.0271	0.408
0.1	0.039	0.117	2.073
0.05	0.079	0.242	4.081
0.01	0.362	1.086	20.349
0.005	0.702	2.116	43.125

Table1: Computation times for serial code

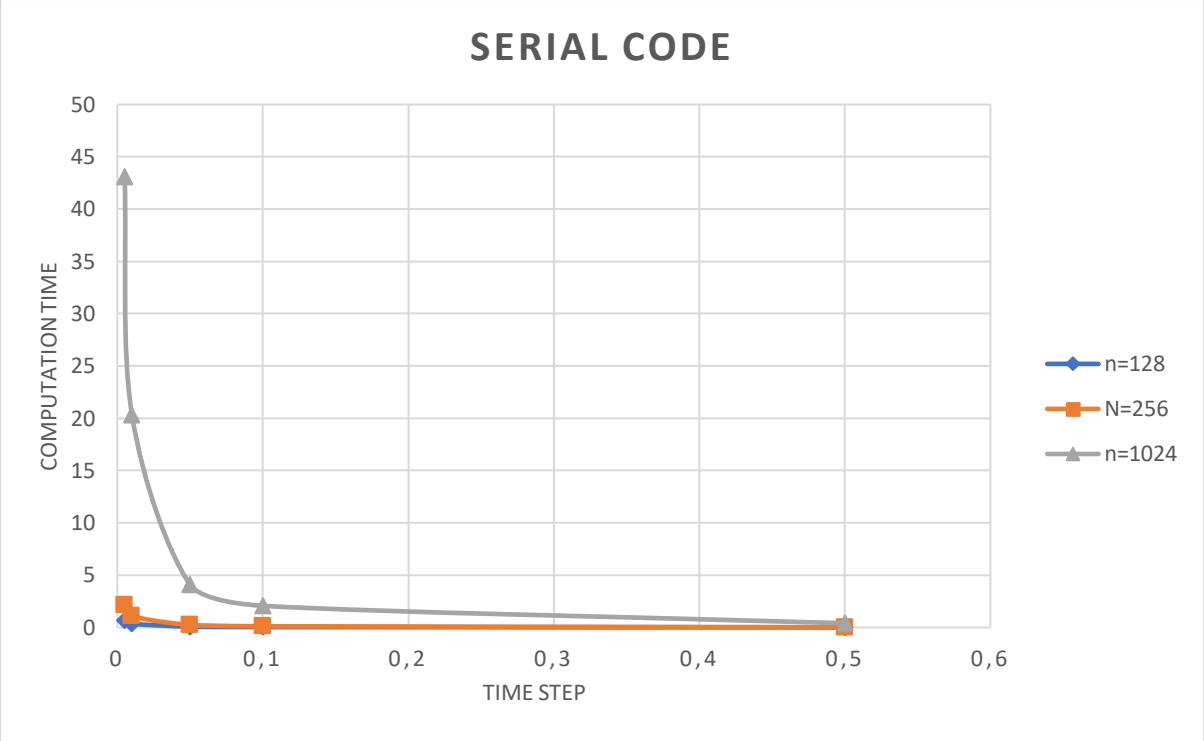

Fig1: Computation times of Serial code

Parallelizing with MPI:

In serial code it has been observed that the maximum amount of time is spent in acceleration update, because this function is called large number of times. So, to improve the running time of the code, parallel program must reduce the time spent in acceleration update. Serial code is parallelized in a way the task of computing accelerations for the different particles are divided among the different processors. There is no data transfer between the processors and all the processors are completely independent. Code performance is analyzed for 2,4,6,8processors for a simulation of duration 10 seconds and at various time steps. Only until 8 processors are used because convergence is achieved with 8 processors.

Time step	Computation time n=128	Computation time n=256	Computation time n=1024
0.5	0.0084	0.0262	0.148
0.1	0.0147	0.0441	0.731
0.05	0.0223	0.0719	1.444
0.01	0.126	0.378	7.337
0.005	0.243	0.739	14.807

Table2: Computation times for Parallel code, processors =2

Time step	Computation time n=128	Computation time n=256	Computation time n=1024
0.5	0.00176	0.00628	0.081
0.1	0.00721	0.02163	0.399
0.05	0.0122	0.0416	0.795
0.01	0.064	0.192	3.867
0.005	0.115	0.355	7.811

Table3: Computation times for Parallel code, processors =4

Time step	Computation time n=128	Computation time n=256	Computation time n=1024
0.5	0.00165	0.00595	0.075
0.1	0.00714	0.02142	0.363
0.05	0.0119	0.0407	0.682
0.01	0.065	0.195	3.471
0.005	0.129	0.397	6.734

Table4: Computation times for Parallel code, processors =6

Time step	Computation time n=128	Computation time n=256	Computation time n=1024
0.5	0.001323	0.004969	0.072
0.1	0.00752	0.02256	0.307
0.05	0.0102	0.0356	0.606
0.01	0.055	0.165	3.261
0.005	0.106	0.328	6.343

Table5: Computation times for Parallel code, processors =8

SCALABILITY TESTS:

Strong Scalability:

Strong scaling is defined as how the solution time varies with the number of processors for a fixed total problem size. In order to perform strong scalability, number of particles are kept constant (at n=128, n= 256, n=1024) and number of processors will be increased gradually until convergence is achieved. This scalability is done at different time steps.

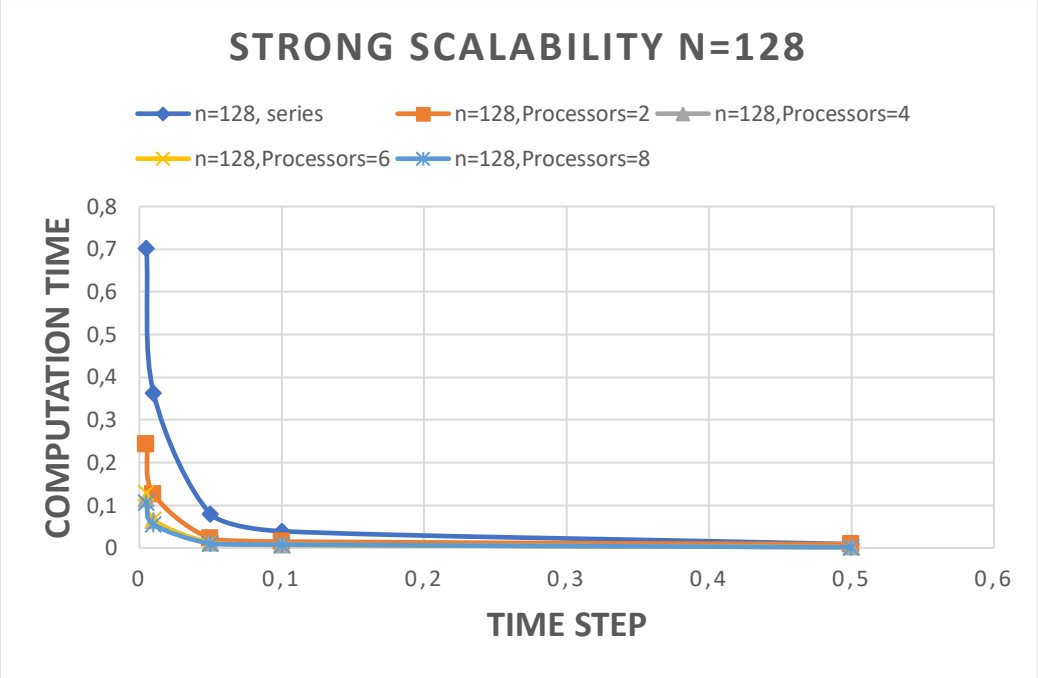

Fig2: Computation times for n=128 at different number of processors

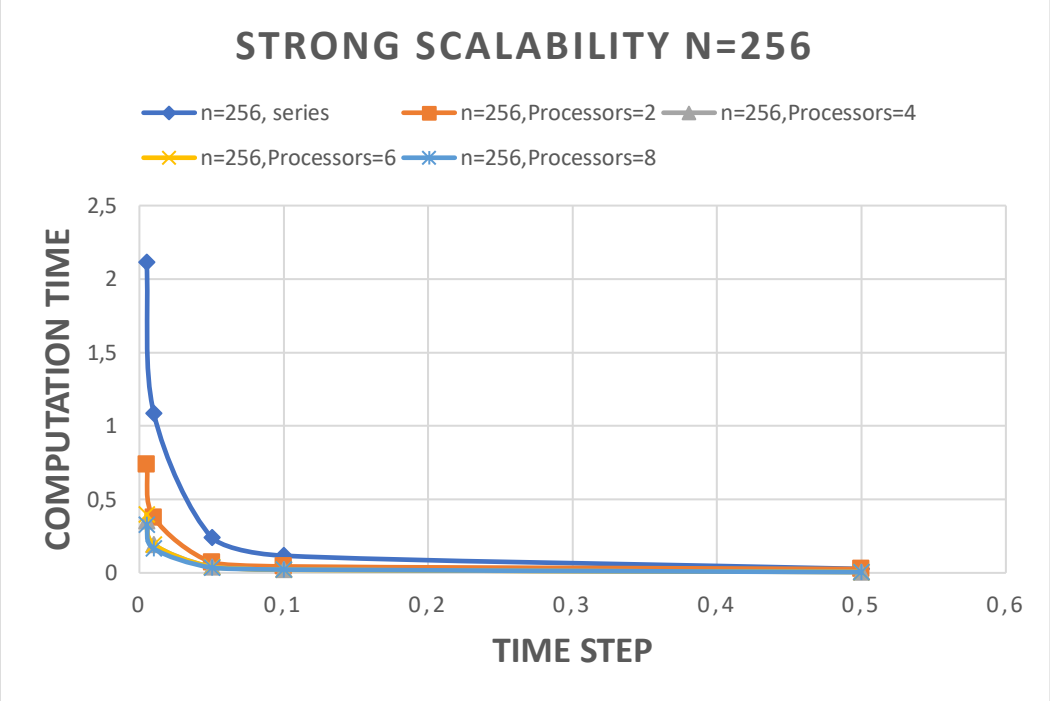

Fig3: Computation times for n=256 at different number of processors

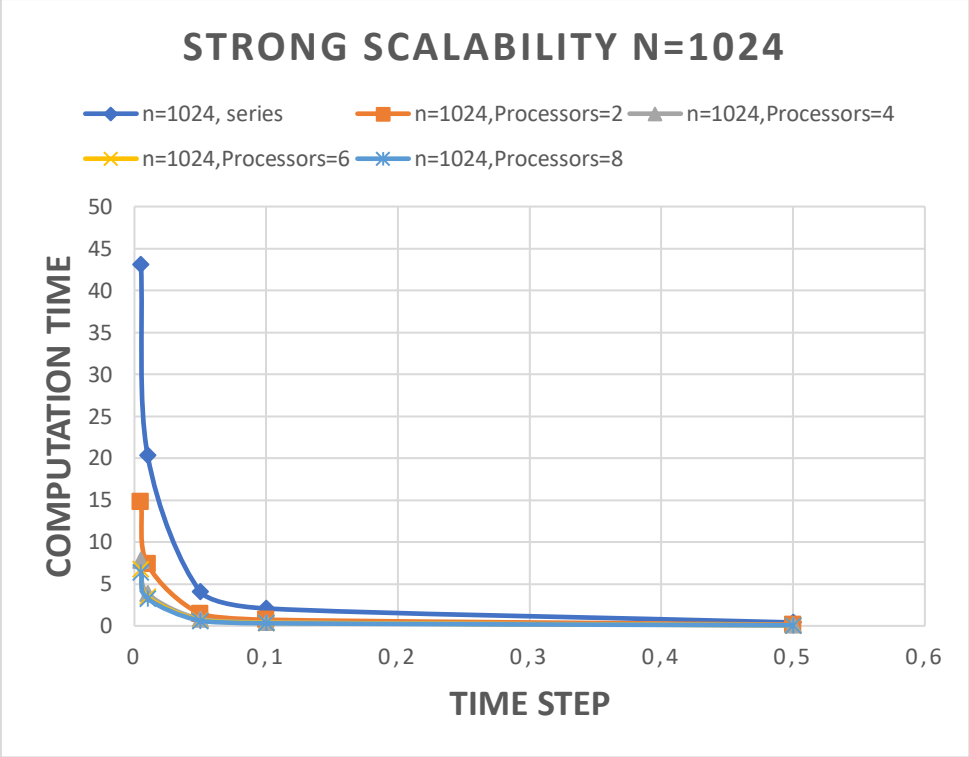

Fig4: Computation times for n=1024 at different number of processors

Observations:

From above plots it can be observed that at larger timestep serial code and the MPI parallel code give almost the same computational time, which means no change in the performance of the code. The performance of the MPI code drastically improves with the smaller and smaller time steps. It can be observed that MPI code with 8 processors provides a speedup of about 7 times than that of the serial code. The increase in the number of processors shows the effect on the computational time, which indicates that the code is successfully parallelized.

Weak scalability:

Weak scaling is defined as how the solution time varies with the number of processors for a fixed problem size per processor. In order to perform the weak scalability test, the number of particles per each processor is kept constant. Here, the number of particles per each processor is taken as 128. MPI code is run for n=128 at 1 processor, n=256 at 2 processors and n=1024 at 8 processors. This means that every time the number of particles per processor is constant. And the code is run at different time steps.

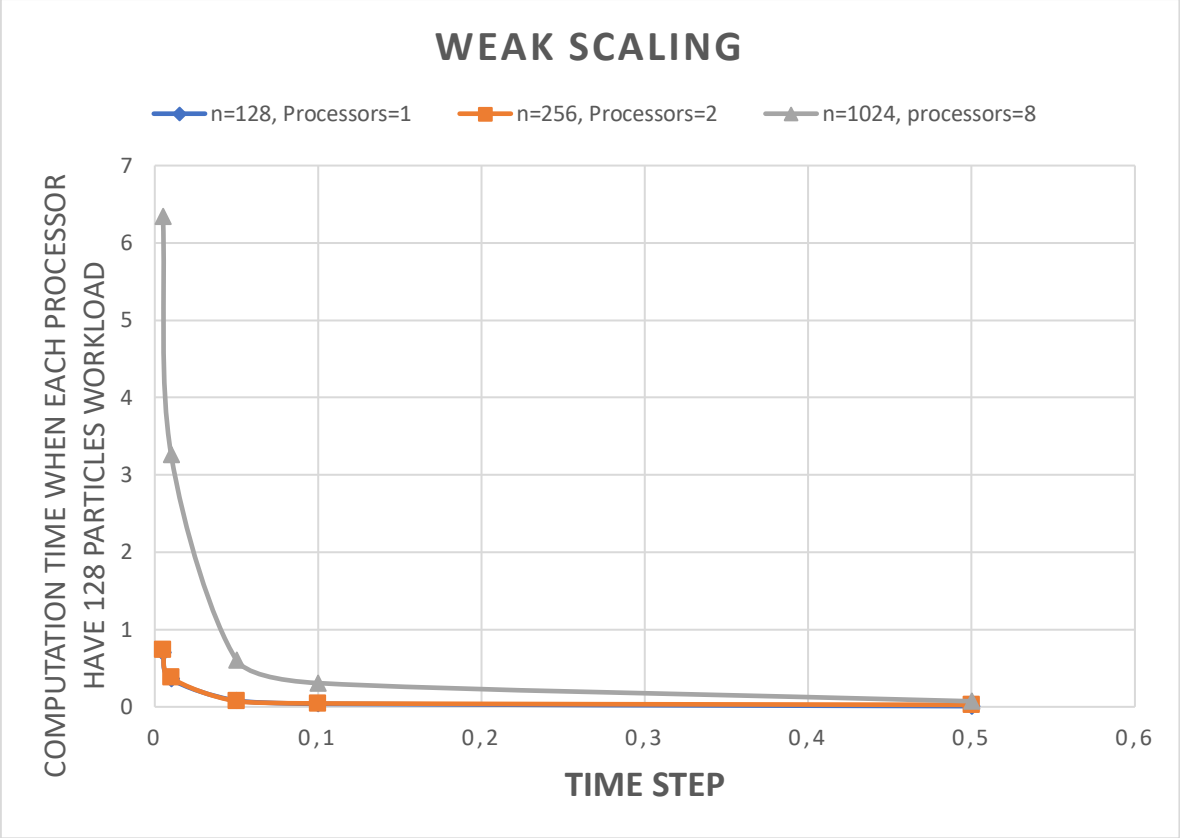

Fig5: Computation time vs Timesteps when each processor has equal number of particles n=128

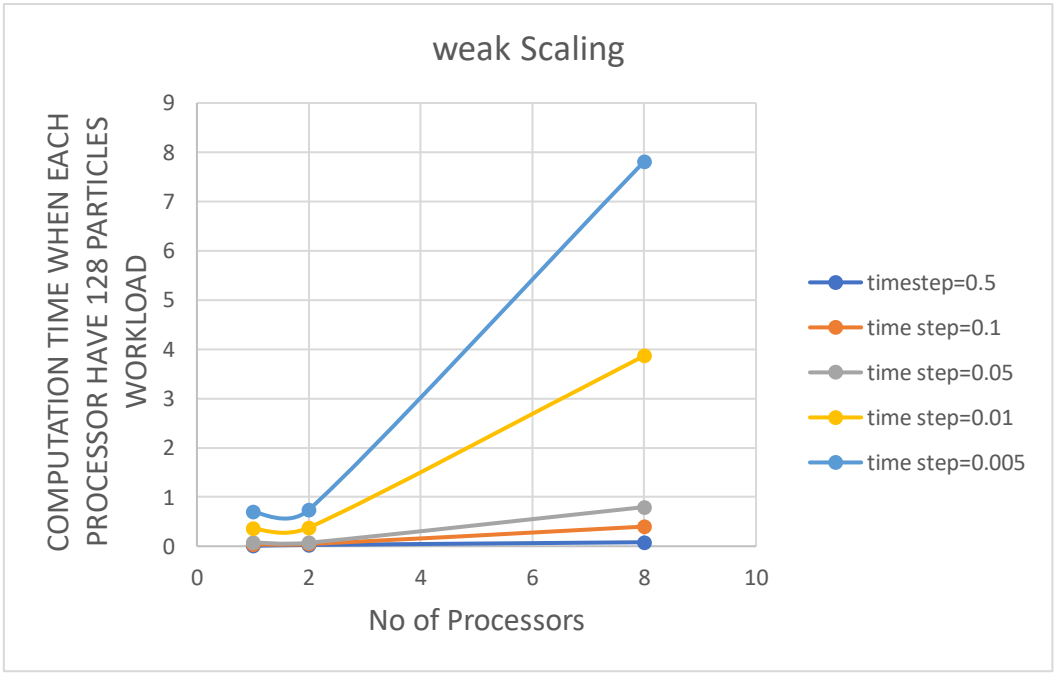

Fig5: Computation time vs No of processor when each processor has equal number of particles n=128 at different time steps

Observations:

Here each processor has a workload of 128 particles. From the above plots it can be observed that the computation time with number of processors increases

linearly although each processor has same workload of 128 particles, this is due to the complexity of problem is $o(n^2)$. Hence it can be concluded that the complexity of $O(n^2)$ is achieved and program is successfully parallelized by using MPI.

Visualization:

Running of serial code or Parallel code creates a text file with name output_positions.txt. This text file contain the x, y, z coordinates of each particles at each time steps. Using these positions, a python code is developed to plot all the particle positions in 3d axes at each time step. This code will generate many plots and will be saved as PNG files. Another python code is written to join all the PNG files to GIF file. This GIF image shows the movement of all the particles.

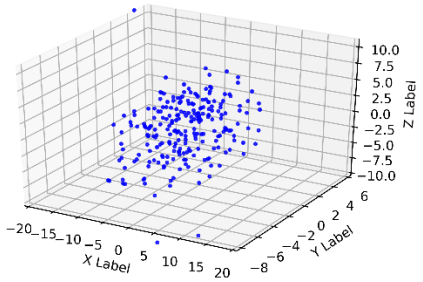

Fig5: at t= 1 sec

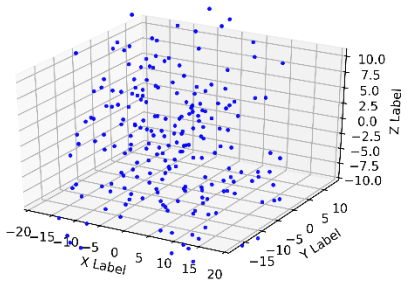

Fig6: at t= 5 sec

USAGE OF THE PROGRAM:

Complilation:

mpicc -o simulation parallel_N_particle_simulationnparticle.c -lm

Run:

Mpirun -np < number of processors > ./simulation

Conclusion:

Strong and weak scalability tests show that the code for the n-Particle simulation is successfully parallelized using MPI. There is no transfer of data between the processors and all the processors are completely independent. Strong scaling shows the change in the computational time with increase of number of processors and weak scaling shows that for complexity of $O(n^2)$ the computational time increases linearly with the number of processors, which are the good indications that the code is parallelized. The positions of the particles are successfully visualized by using python programming.