Title Page

# 1. Hardware Resource Description

The program was executed on UG lab machine with the following hardware specifications.

## 1.1 Number of Cores

## The machine is equipped with 8 physical cores and a total of 16 logical CPUs (threads), allowing for robust parallel computation.

## 1.2 Other Relevant Configurations or Constraints

Architecture: x86\_64

CPU Model: 11th Gen Intel(R) Core(TM) i7-11700 @ 2.50GHz

Socket(s): 1

# 2. Program Architecture

A screenshot of a computer

Description automatically generated

## 2.1 Mode 1 Architecture

## A diagram of a computer program Description automatically generated

## In Mode 1, the program performs all computations sequentially. No multi-threading is used, and the program iterates through each particle one by one to compute the force with the closest neighboring particle. To maintain consistency in computation, an "INF" particle is padded at both the beginning and the end of the particle list. This allows for the use of three points: , and for force computations, even for the first and last particles.

## 2.2 Mode 2 Architecture

A diagram of a computer program

Description automatically generated

Mode 2 employs a multithreading model. The program creates multiple threads at the start, and each thread is assigned a portion of the dataset to work on. The dataset is divided as evenly as possible among the threads. Once a thread completes its calculations, it returns. To ensure accurate computation, sub-chunks that include neighboring particles are used. For example, to compute the forces for particles 4, 5, and 6, the sub-chunk assigned to the thread would be particles 3, 4, 5, 6 and 7.

## 2.3 Mode 3 Architecture

A diagram of a diagram

Description automatically generated with medium confidence

# Mode 3 uses MPI to create leader processes, each responsible for an almost equal partition of the dataset. These leaders further subdivide their data into smaller chunks and place them in a queue. Worker threads then pick up these chunks from the queue to perform the force computations. The threads continue to take more work from the queue until it is empty, ensuring efficient load-balancing and utilization of resources.

# 3. Results

Please see attached verification\_results.csv.

# 4. Parallelism

## 4.1 Use of Software Threads/Processes

* Mode 1: No use of software threads or processes, entirely sequential computation.
* Mode 2: Uses multiple software threads created at the start to divide the dataset as evenly as possible among the threads.
* Mode 3: Employs MPI to create leader processes that manage almost equal partitions of the dataset. Each leader further divides the data into smaller chunks placed in a queue. Worker threads pick these up for computation.

## 4.2 Use of Hardware Threads/Cores

The program was executed on a machine with 8 physical cores and 16 logical CPUs. Modes 2 and 3 are designed to make efficient use of multiple cores and threads for parallel computation.

# 5. Parallel vs. Serial

## 5.1 Mode 1: Serial Parts

In Mode 1, all computations are performed serially. The program iterates through each particle one by one to calculate the Coulomb force with the closest neighboring particle.

## 5.2 Mode 2: Parallel and Serial Parts

* Parallel: The force computations are parallelized, with each thread responsible for a portion of the dataset.
* Serial: The initial data parsing and final result aggregation are performed serially.

## 5.3 Mode 3: Parallel and Serial Parts

* Parallel: The force computations are parallelized. Each MPI leader process handles a nearly equal partition of the dataset and manages a queue of smaller data chunks for worker threads.
* Serial: Similar to Mode 2, the initial data parsing and final result aggregation are serial operations.

# 6. Mode 2: Time vs. Number of Threads

## 6.1 Chart/Table/Diagram

## 6.2 Explanation of Results

# 7. Mode 3: Execution Time vs. Data Size

## 7.1 Chart/Table/Diagram

## 7.2 Explanation of Results

# 8. Execution Time vs. Mode

## 8.1 Chart/Table/Diagram

## 8.2 Explanation of Results

# 9. Speedup

## 9.1 Explanation of Superlinear Speedup

## 9.2 Justification

# 10. Re-usability

## 10.1 Optimized Code Parts