Exercise Set 8

Darina Öö

18th of March 2024

1 Code Description

The core of the simulation involves calculating the acceleration, potential energy, and kinetic energy of each particle based on its interactions with neighboring particles. The simulation leverages MPI to distribute the computation across multiple processors, reducing the overall wall clock time required for execution.

The modified code could be found in the md1d2.cpp file. Here's a brief overview of the code's key components:

- Initialization of MPI and determination of the rank and size of the MPI world to facilitate parallel computation.
- Division of the total number of atoms (`nat`) among the available processors, ensuring each processor works on a subset of the entire system.
- Synchronization of computations across processors using `MPI_Reduce` to aggregate the total potential and kinetic energies, which are then output by the master process.
- Use of a simple leapfrog integration scheme for updating the velocities and positions of the particles.
- Calculation of elapsed wall clock time using high-resolution clocks, to measure the performance improvement with varying numbers of processors.

2 Running Instructions

First, compile the source code using the `mpicxx` compiler by executing the command `mpicxx -o md_simulation md1d2.cpp` in the terminal, within the directory containing the source code. This produces an executable named `md_simulation`.

The program can be executed with the `mpirun` or `mpiexec` command, depending on the MPI implementation. The general syntax for running the program is:

```
mpirun -np <number_of_processors>
./md_simulation <nat> <dt> <maxt> <vsc> <eout> [coout]
```

where <number_of_processors> specifies the number of processors, <nat> the total number of atoms, <dt> the time step, <maxt> the maximum number of time steps, <vsc> the velocity scale factor for initializing particle velocities, <eout> the frequency of energy information output, and [coout] (optional) the frequency of coordinate output.

For example, to run the simulation on 4 processors with 100 atoms, a time step of 0.01, for 1000 time steps, a velocity scale factor of 1.0, and to output energy information every 100 time steps, use the command:

```
mpirun -np 4 ./md_simulation 100 0.01 1000 1.0 100 > output_4p.txt 2>&1
```

The number of atoms (<nat>) should be divisible by the number of processors for an even workload distribution. The simulation's performance can vary based on the number of processors, specific hardware, and MPI implementation used.

3 Performance Analysis

The performance of the program was tested on an Ubuntu system running through Parallels on an M1 Mac. The simulation's wall clock time was observed for 1, 2, and 4 processors:

Number of Processors	Wall Clock Time (seconds)
1	46.1668
2	33.7062
4	29.7683

As evident from the table and Figure 1, the wall clock time decreases as the number of processors increases. This trend demonstrates the benefit of parallel processing in reducing the execution time for computationally intensive simulations. However, the rate of decrease in wall clock time diminishes with more processors, indicating a point beyond which adding more processors would yield diminishing returns due to the overhead associated with inter-processor communication.

4 Conclusion

The parallel molecular dynamics simulation showcases the effectiveness of using MPI for distributing computational tasks across multiple processors. The observed reduction in wall clock time with increased processors highlights the potential for significant performance improvements in scientific computations through parallel processing. Future work could explore optimizing communication between processors and expanding the simulation to three dimensions.

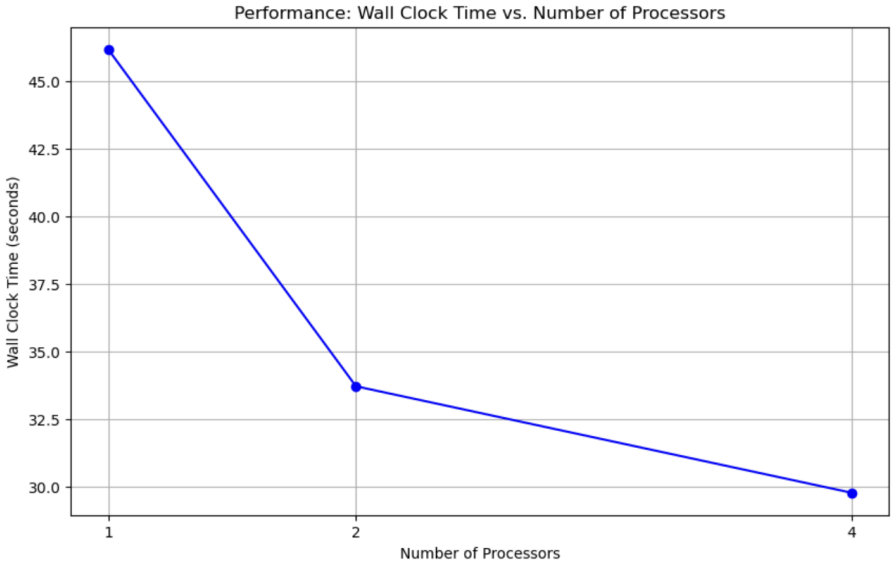

Figure 1: Performance