ENPH 479 High Performance Computational Physics Parallelized Computing

Jane Cohen[1, *]

[1] *Department of Physics, Engineering Physics and Astronomy, Queen's University, Kingston, ON K7L 3N6, Canada*
In association with the Kingston Centre for Advanced Computing

(Dated: March 8, 2024)

In this paper, we investigate the efficiency of parallel computing for solving complex physical problems, such as the 1D heat equation. Through the use of the Message Passing Interface (MPI) and the Frontenac High Performance Computing cluster, we demonstrate a substantial reduction in computation time compared to serial computing methods. Our approach involves dividing the computational task across multiple processors, allowing for simultaneous calculations and data management through the SLURM scheduler. The results demonstrate the practical advantages of parallel computing in terms of speed and efficiency and the accuracy of the computational model, as confirmed by the expected physical behavior of heat dispersion in an aluminum rod.

I. BACKGROUND

A. Parallel Programming

Parallel programming utilizes multiple cpus to carry out simultaneous calculations or tasks. In our case MPI, the Message Passing Interface, is used to accomplish parallel programming. MPI provides a standardized way of passing data or "messages" between computers that are working in parallel. To be able to run MPI scripts, the Frontenac cluster is used. The Frontenac Cluster is a High Performance Computing cluster that allows computational tasks to be carried out using multiple cores or cpus.

B. SLURM

Each computer using the cluster can submit jobs to be carried out. To control which jobs are run and when, a scheduler is used. For the Frontenac cluser, the Simple Linux Utility Resource Manager (SLURM) scheduler is used. To run an MPI script, a SLURM job script must be written in addition to the MPI program. This job script specifies the resources that are needed and allows the script to be submitted as a job. Once the job has been submitted, SLURM takes over and runs the job.

C. SLURM Job Script

The SLURM job script used specifies the number of nodes and processes to be used for the job. In this case, one node and eight processors were used. In addition, the job script was used to specify the simulation time and output of the program. Because the job produces a plot that is saved the the local directory, the output file has no contents. However, it was included in the job script to allow for de-bugging as the script was written. For the simulation time, the program needs to be allocated enough time to complete all parts of the program. To ensure the end time was not interrupting any processes, the time taken to run the serial program was recorded, and used (including an additional time buffer) for the parallel program.

D. Solving the 1D Heat Equation

To explore the advantages of parallel computing, the 1D heat equation was solved using parallel programming. The MPI script written simulates the temperature distribution along an aluminum rod over time. At the start, the script initializes MPI communication, identifying the total number of processes and assigning each a unique rank. The spatial domain of the rod is then divided into segments, each allocated to a different MPI process, with the root process distributing these segments to ensure each process operates on a distinct portion of the rod. This setup enables parallel computation, where each process calculates the initial temperature distribution for its segment and sets appropriate boundary conditions. During the time-stepping phase, processes update temperature values based on a numerical approximation of the heat equation, exchanging boundary temperature values with adjacent processes to maintain continuity across segment boundaries. After completing the time steps, the index of iteration is checked. If it is an index of a time step specified by the user (in the array called snap_index), then the current local temperature arrays from each process are gathered. The root process then plots the data for the entire rod for this specific time stamp.

II. RESULTS

To demonstrate the benefits of parallel programming, the results of the parallel program described above were compared to a serial script written to solve same equation. Both programs used a rod length of 1m split over 1024 points, a diffusion constant D of $2.3 \times 10^4 \frac{m^2}{s}$ and fixed boundary conditions of 20°C for the ends of the rod. The serial script took 8.24s to complete all calculations and produce a plot. The parallel program was able to accomplish

* 18jpcw@queensu.ca

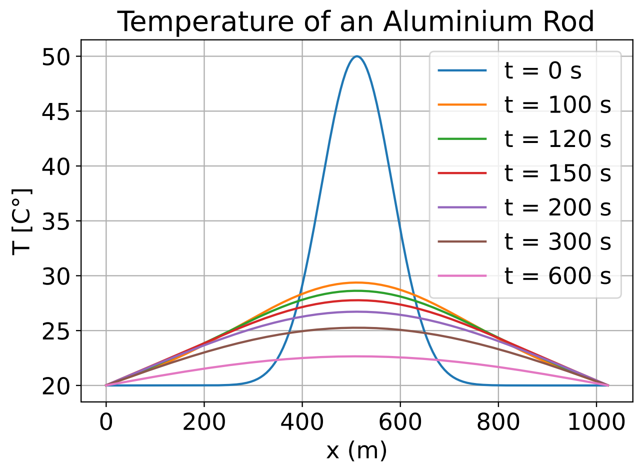

Figure 1. The temperature of an aluminium rod over time. As time increases, the temperature disperses across the rod. The ends of the rod are held at a boundary condition of 20 °C.

the same task in 3.28s. Figure 1 shows the results of this parallelized program. Over time, the heat in the rod disperses over its length. This is the expected behaviour of the system.

III. CONCLUSION

In this study, we explored the advantages of parallel computing through the implementation of MPI (Message Passing Interface) to solve the 1D heat equation. The use of the Frontenac High Performance Computing cluster and the SLURM scheduler facilitated the distribution of computational tasks across multiple processors. By partitioning the spatial domain of an aluminum rod into segments processed in parallel, there was a notable reduction in computation time. This outcome highlights the efficiency of parallel computing in handling computationally intensive tasks and demonstrates the critical role of MPI in enabling communication and coordination among the processors involved. The results, as shown by the temperature distribution over time, agree with the expected physical behavior of heat dispersion in a rod, thereby validating the computational model.