Evan Teschko
Assignment 3
Parallel and Concurrent Programming
04/26/2024

Program Design:

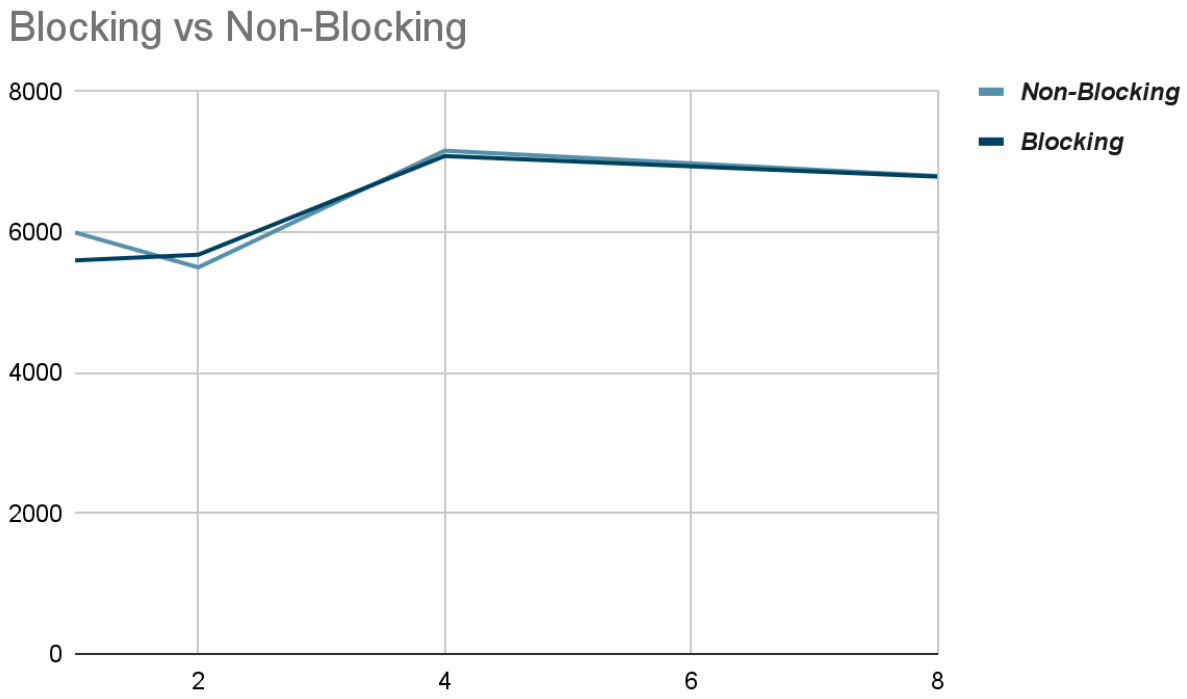
This MPI-based program simulates Conway's Game of Life using parallel processing to enhance performance across multiple processors. Initially, the root process sets up a large grid with random alive or dead cells and distributes this configuration to all processes using `MPI_Bcast` for consistent starting conditions. The grid is divided into horizontal sections (rows) assigned to each processor, which independently updates its segment. Boundary data between these sections are synchronously exchanged using `MPI_Sendrecv`, ensuring each processor can accurately apply the game's rules to its cells.

The simulation employs `MPI_Allreduce` to check for changes across the grid at the end of each generation, allowing for an early termination if no changes are detected. The reliance on blocking communications aids in simplicity but could impact scalability under certain conditions.

Test Plan and Test Cases:

Machine Name	**MacPro**
Os Name(version)	**Sonoms 14.1**
Procesor	**M1 pro**
Clock Speed	**3.2 GHz**
Compiler	**Mpicc**

I plan to evaluate the performance of my Game of Life simulation by running a series of tests across various processor counts using a script. The script is designed to execute the program with 1, 2, 4, 8, 16, and 32 processes, repeating each test three times to ensure consistency and reliability in the results. It utilizes `mpirun` to initiate the program under different conditions and captures the execution time for each run, logging these timings into an output file. This approach allows me to gather performance data, which will be crucial for analyzing the efficiency of the parallelization.



I think these tests help show that the parallelized version that ran using openmpi was significantly faster than the base version of Conway's Game of Life. The base version ran the test in around 3 hours but using openmpi it was about 1 hour and 30 minutes on average.

Analysis and Conclusions:

In testing Conway's Game of Life on a laptop with up to 8 cores, I've observed both the computational limits and scalability of parallel processing. The performance indicates potential for scalability but also highlights diminishing returns due to increased overhead from inter-process communication. This suggests that inefficiencies in my code's messaging system may be slowing the application over time rather than optimizing it. When comparing the run times with my other programs that I made during the class this was definitely the fastest the code has run. Also something went wrong with my script when I was testing. The processes seemed to start and end at different times in my output so I tried to arrange them in a way that made sense.

Limited by my hardware, I couldn't test beyond 8 cores. This limitation emphasizes the need for more advanced computing infrastructure to fully evaluate and optimize the scalability of MPI applications. To address the observed inefficiencies in the message-passing implementation, more comprehensive testing with the programs needed.

Works Cited

1

Kendall, Wes. "MPI Broadcast and Collective Communication." *MPI Tutorial*, mpitutorial.com/tutorials/mpi-broadcast-and-collective-communication/.

2

Nurkkala, Tom. "MPI Basics" *YouTube*, uploaded by Nurkkala, 3 years ago, www.youtube.com/watch?v=c0C9mQaxsD4.