Report Assignment 2 - MPI

Emre Akdag s1087894 - Danny Swart s1083428 - Alwin van den Eyssel s1083142
Parallel Computing
May 2023

Discussion

We were unable to create a working implementation. We do, however, have code as a proof of concept for our idea. To explain the idea, let's start with a bit of pseudo-code:

```
MASTER
for i in range(iterations)
        for each vector element v in V with index i

                // add logic for boundary checking
                send V[i-1] to node c+1
                send V[i] to node c+1
                send V[i+1] to node c+1

                receive V[i] from node c+1

WORKER
receive V[i-1] from node 0
receive V[i] from node 0
receive V[i+1] from node 0

result = computeIteration(V[i-1],V[i],V[i+1])
send result to node 0
```

This pseudo-code shows how the master delegates the work of computing the next iteration to other ranks. In short, the master node (rank 0) sends a vector element along with its neighbors to a worker node (rank n, where n > 0) to compute the next generation. This happens for each of the vector elements in the vector. This means that in an iteration, the vector elements are computed in pairs of the amount of ranks.

As is clear from this explanation, this could potentially be implemented using MPI_Scatter, as we are dividing the vector in small chunks to be processed by our worker nodes. Do note that we are not sub-dividing individual elements in our vector, but in groups of three instead (neighbors). This would look like:
V = { A, B, C, D, E}
r_0 = (master)
r_1 = {0, A, B}
r_2 = {A, B, C}
r_3 = {B, C, D}
r_4 = {C, D, E}
r_5 = {D, E, 0}

The divided results would then need to be gathered by MPI_Gather.

The attached code shows this idea (without the use of MPI_Scatter), but likely with too many errors in semantics.

Anticipated results

Given what we know about MPI, it is likely that our MPI implementation idea is just about as fast as an OpenMP implementation. This is because the algorithm as stated does not take into account the algorithm improvements by making use of what we know about the cache and how it influences the speed of the program which were stated in the assignment. If we do take into account those optimizations, then it is likely to be much faster.

Team

Our overall contribution to the project was evenly distributed, since we did every task of the project together.

Code:	Emre 33.3%, Danny 33.3%, Alwin 33.3%
Data collection:	Emre 33.3%, Danny 33.3%, Alwin 33.3%
Report:	Emre 33.3%, Danny 33.3%, Alwin 33.3%