Algorithm Description: The goal of this program is to come up with an OpenMP and an MPI program for the 1D Jacobi problem. Then compare the execution times of these programs along with comparing execution times using multiple processors, cores, and buffer sizes for the MPI program.

Description of parallel approach: Our OpenMP program is going to be using a parallel "for" loop when computing the values for the "cur" array. This one task will greatly decrease execution times for the program. We are going to be using a blocked (halo) approach in our MPI program. This means that we are going to be executing this program with multiple threads computing the first and last indices for our computed array and then make their way inwards to compute the rest. And lastly joining them together for our final array.

Machine description:

- **Name**: cod, anchovy (for multi-processing)
- **Number of cores**: 8, up to 6 used per machine
- **Cache sizes**:
  - L1d cache: 32K
  - L1i cache: 32K
  - L2 cache: 256K
  - L3 cache: 15360K

Experimental results:

Figure 1:

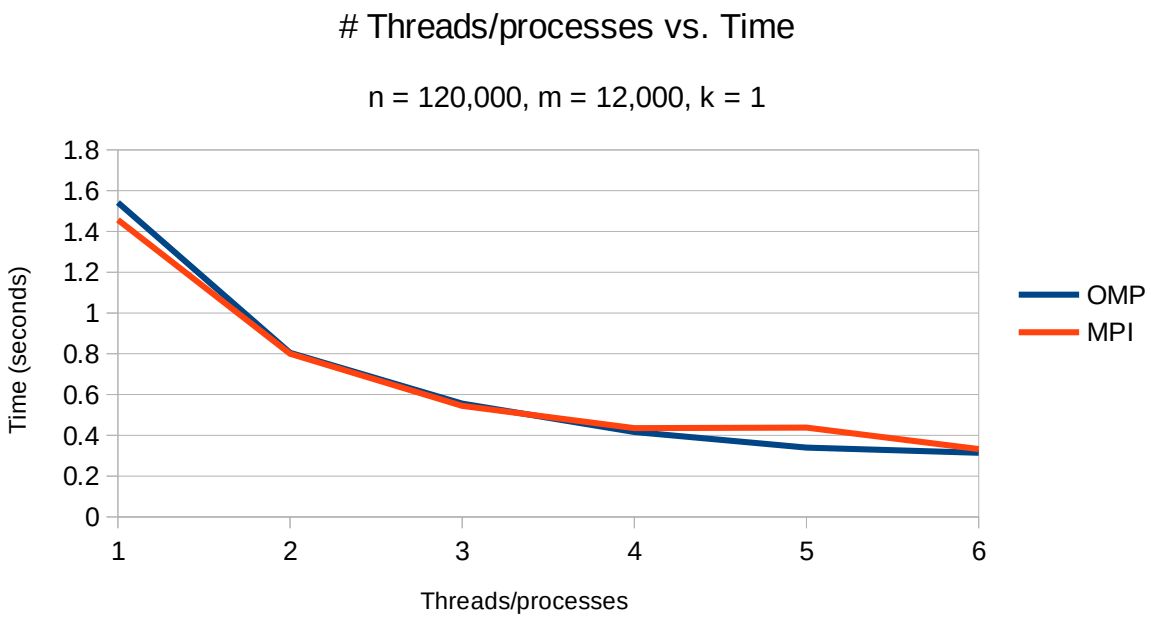


Figure 2:

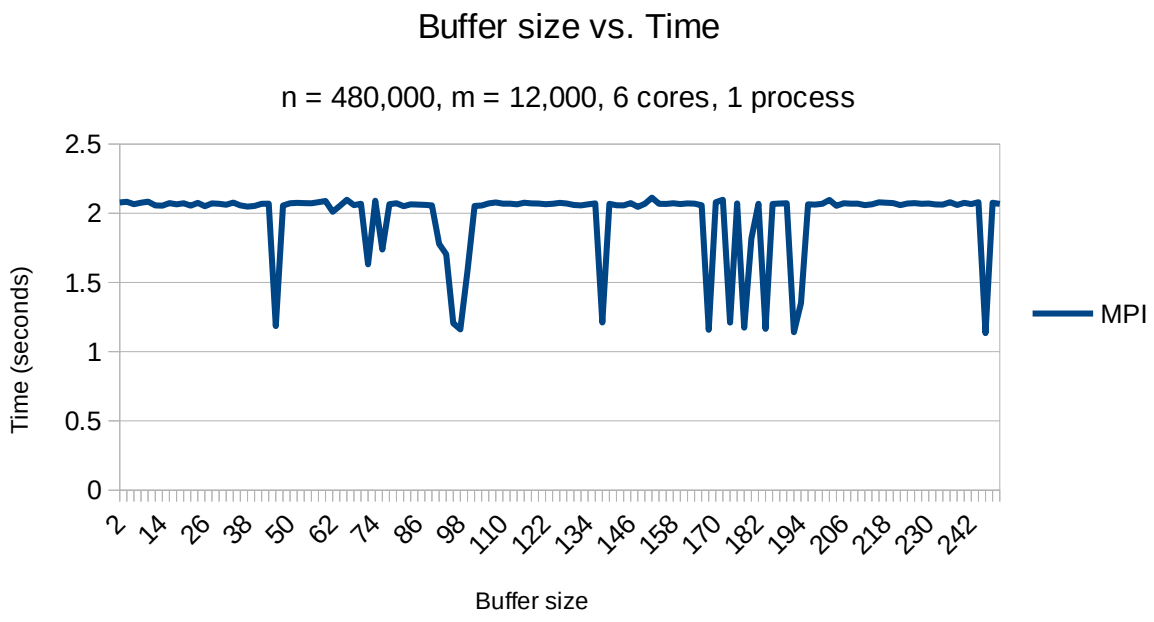
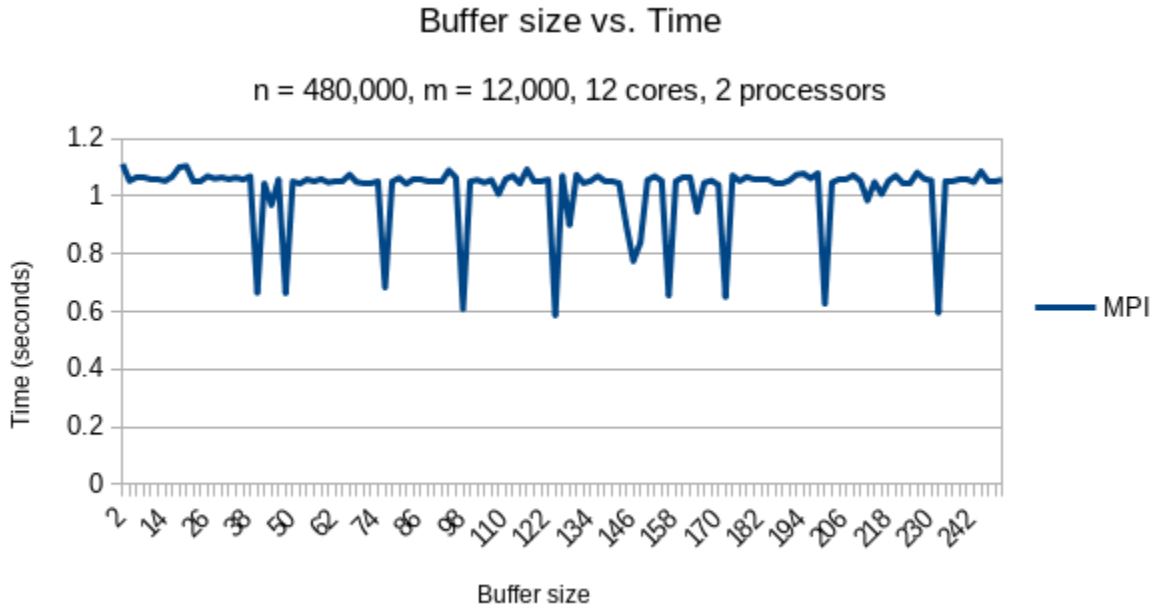


Figure 3:



Conclusion:

The execution time between OpenMP and MPI didn’t differ much when charted side by side, so I don’t think there is much benefit to using MPI over OpenMP in that case. I’ve learned that buffer size doesn’t have much to do with performance, except that I get random drops in execution time. I’m not sure what causes the sudden drop in execution times in figures 2 & 3 because it is very inconsistent when I test it, and I don’t get the same results or drops every time.