Doruk Taneli 60066

# Assignment 3 Report

I uploaded a single main.cpp for both part 1 and part 2. Part 1 can be run by commenting out the two #pragma omp.

## Part 1

First I initialized the MPIs.

```
// Initializations
MPI_Init(&argc, &argv);
MPI_Comm_size(MPI_COMM_WORLD, &num_procs);
MPI_Comm_rank(MPI_COMM_WORLD, &myrank);
```

Then only the master thread is reading the .mtx file.
Master thread then calculates the size and displacements of csr array portions to send to each processor.
Next, the csr arrays are scattered to all threads using MPI_Scatter and MPI_Scatterv.
After scattering, I start the clock.

For each time step, all processes calculate their parts of the array.
Then the results are gathered and broadcasted in an efficient way using MPI_Allgatherv.
Rhs is set to the new result, then the next time step starts.

After All time steps are calculated, I call MPI_Finalize.

I am getting an error that I couldn't solve so I couldn’t do the tests.

## Part 2

I calculated the number OpenMP threads relative to the mpi processes so the total number of threads would be 16 at all tests.

```
tot_omp_threads = 16 / num_procs;
omp_threads_per_mpi = tot_omp_threads / num_procs;
```

I parallelized using OpenMP as such:

```
for (int k = 0; k < time_steps; k++)
{
  #pragma omp parallel for shared(result) num_threads(omp_threads_per_mpi) schedule(dynamic)
  for (int i = 0; i < matrix.n; i++)
  {
    myResult[i] = 0.0;
    for (int j = myRowptr[i]; j < myRowptr[i + 1]; j++)
    {
      #pragma omp atomic update
      myResult[i] += myMatVal[j] * rhs[myColInd[j]];
    }
  }
```

I made the scheduling dynamic so it has better load balancing.

I couldn't finish Part 1 so I couldn't do the tests for Part 2 either.

# References

I looked at examples from the following codes on the internet.

1. https://mpitutorial.com/tutorials/ → scatter/gather parts
2. https://github.com/taoito/matvec-mpi/blob/master/matvec_mpi.c