I figured out that my map-reduce version of the past assignment using OpenMP was giving the wrong output. Therefore, I modified my version this time.

Some problems I encountered were when I was trying to update a dictionary while processes where doing their tasks, they somehow kept overwriting the dictionary. For example, if thread 1 found 6 instances of the word hate and thread 2 found 10 instances of the word hate the final dictionary was 10 occurrences of hate instead of 16.

Then I remembered that MPI is quite different than openMP in regards of memory in processes. Each thread has independent memory therefore I had to have a shared object type of approach.

That’s when I using the method gather(), where the global communicator will gather all the dictionaries that the processes updated and it would return a list of dictionaries. Then, thread of rank 0 would sum all occurrences and output a final dictionary.

Now let’s see their output and runtime:

1 Thread:

Text

Description automatically generated



2 Threads:

Text

Description automatically generated



4 Threads:

Text

Description automatically generated



8 Threads:

Text

Description automatically generated



Overall Count:

{'hate': 188, 'love': 2404, 'death': 964, 'night': 825, 'sleep': 272, 'time': 1180, 'henry': 615, 'hamlet': 475, 'you': 14591, 'my': 13198, 'blood': 719, 'poison': 98, 'macbeth': 288, 'king': 3039, 'heart': 1125, 'honest': 309}

As you can see, my program distributes files evenly, and outputs everything in parallel when running.

CPU INFO: model name : Intel(R) Core(TM) i7-1065G7 CPU @ 1.30GHz

4 36 220