Map Reduce Phase 3

CIS-687 OBJECT ORIENTED DESIGN

PROF. SCOTT ROUECHE

5/31/2022

Anthony Redamonti
SYRACUSE UNIVERSITY

Map Reduce Phase 3:

1. Link to repository: https://github.com/ColtonWilson/MapReduce_Project_Phase-3

2. "How-To" document and code reviews are in the repository.

3. I was responsible for designing and implementing the Map Process and Reduce Process projects. Both projects build executables which are called by the Workflow class in the MapReduce project. The executables use methods contained in the MapLibrary and ReduceLibrary DLL's. The Map Reduce project defaults to 5 processes (5 map and 5 reduce). I also implemented 5 threads per process in the reduce process. To configure the number of processes or the number of threads per process, change the constant variables RMAX or TMAX in Workflow.cpp.

   My partner, Colton Wilson, implemented multithreading in the map process. He also performed unit testing and found/fixed bugs in the Workflow class and the Map Process project. He also designed and implemented an algorithm for combining all output files from the reduce threads into one file.

   The Workflow class creates all map processes and waits for them to finish. It then creates all reduce processes and waits for them to finish.

   In the beginning, all lines in all input files are inserted into a vector. Next, the vector is divided evenly into smaller vectors, one for each map process. The smaller vector is split (again) evenly for each thread in the process. Each thread produces an intermediate file. The map process waits for all its threads to complete, then combines all the intermediate files created by each thread into one intermediate file. When all map processes finish, the number of intermediate files should be equal to the number of map processes created.

   For example, if using 2 processes with 2 threads per process with file name "IntermediateFile.txt":

   IntermediateFile11.txt is created by process 1, thread 1.
   IntermediateFile12.txt is created by process 1, thread 2.
   Process one then combines them into IntermediateFile1.txt and deletes the superfluous files.

   The reduce process behaves similarly. Each reduce process takes in a unique intermediate file. It then splits that file into smaller files for each thread in the process. After all threads and all processes have finished, an algorithm is used to combine the output file from each thread into one file.

4. The project was a success and a learning experience. It took some time to research how to create a process in a Windows OS. The CreateProcess method provided by the Windows API turned out to be the best method. Inserting each process ID into a vector and waiting on each process to finish was a more reliable method of waiting for multiple processes to finish than the

WaitForMultipleObject method provided by the Windows API. We were able to produce the same output as previous project phases, and the overall runtime was reduced.

My partner, Colton Wilson, was great. He is always responsive and helpful.