CS 4175 – Parallel Computing
Professor: David Pruitt
Adrian Avendano
80615869

Project 2 – Map Reduce Assignment

Problems encountered when completing the assignment and how I overcame them.

The main problem encountered in this lab was the global dictionary. I was getting inconsistent numbers at all iterations of the program; this was solved through the use of a local dictionary instead of assigning everything to the shared dictionary. Also, the computer I am using for the VMs behaves erratically with technical issues which consumes most of the time rather than the implementation.

Any problems I was not able to overcome.

In this case, I was able to overcome most of the problems since the logic/implementation for this assignment was not as complicated. It was mostly looking at the examples provided from the professor.

About how long it took me to complete the assignment.

Implementation took me around ~2-3 hours since most of the logic was done on the previous assignment. It took me around another our to access the VM on the computer/

Performance Measurements for 1, 2, 4, and 8 threads.
Output from cmd (got rid of some print statements to avoid clutter in report. Those statements are present in actual cmd and code):

student@linux-4oae:~/Desktop/map-reduce-adrDav> python3 mapReduce.py

Starting count_words function with 1 thread(s)
Elapsed time for thread #1: 2.970501793000949
{'hate': 332, 'love': 3070, 'death': 1016, 'night': 1402, 'sleep': 470, 'time': 1806, 'henry': 661, 'hamlet': 475, 'you': 23306, 'my': 14203, 'blood': 1009, 'poison': 139, 'macbeth': 288, 'king': 4545, 'heart': 1458, 'honest': 434}
Total word count: 54614

Starting count_words function with 2 thread(s)
Elapsed time for thread #2: 1.8461219469991192
{'hate': 332, 'love': 3070, 'death': 1016, 'night': 1402, 'sleep': 470, 'time': 1806, 'henry': 661, 'hamlet': 475, 'you': 23306, 'my': 14203, 'blood': 1009, 'poison': 139, 'macbeth': 288, 'king': 4545, 'heart': 1458, 'honest': 434}
Total word count: 54614

Starting count_words function with 4 thread(s)
Elapsed time for thread #4: 1.1633956570003647

{'hate': 332, 'love': 3070, 'death': 1016, 'night': 1402, 'sleep': 470, 'time': 1806, 'henry': 661, 'hamlet': 475, 'you': 23306, 'my': 14203, 'blood': 1009, 'poison': 139, 'macbeth': 288, 'king': 4545, 'heart': 1458, 'honest': 434}
Total word count: 54614

Starting count_words function with 8 thread(s)
Elapsed time for thread #8: 1.0797202269986883
{'hate': 332, 'love': 3070, 'death': 1016, 'night': 1402, 'sleep': 470, 'time': 1806, 'henry': 661, 'hamlet': 475, 'you': 23306, 'my': 14203, 'blood': 1009, 'poison': 139, 'macbeth': 288, 'king': 4545, 'heart': 1458, 'honest': 434}
Total word count: 54614

Why the program behaves as it does with the number of threads?

Multiple threads mean that the program can handle multiple actions in this case it can “kind of” divide the transcripts and work concurrently with other threads, thus taking less time for the program to finish. In this case, more threads means that the program is able to finish faster, however creating an insane amount of threads can also mess with the time and create chaos in the computer.

Output from the cpuInfoDump.sh

model name : Intel(R) Core(TM) i5-8300H CPU @ 2.30GHz
8 72 432