WordFrequencies

Michael Edelman

December 2021

1 Sequential Design

All processing happens in the doIt().The Sequential Implementation uses a BufferedReader to read in a text file into a String. That String is then split into words. **Those words are stored in an array**. That array is then iterated through, updating a HashMap that maintains a word to frequency mapping. While iterating the algorithm keeps note of the current max Frequency. After the HashMap is created the algorithm iterates through its entry set two times. The first time to find all the entries with the max frequency value we found during the HashMap creation and put them in a set. The second time to find all the entries with a frequency value of one and put them in a set.

2 Parallel Design

The parallel solution closely models the sequential one in its main concept. The main difference is the parallel solution partitions **the array of words** amongst a number of threads and is concurrently processed by those threads to produce the HashMap with ”more hands than one”. The design utilizes Javas synchronized keyword,the Thread class, and join(). The synchronized key word is used to ensure the max frequency value variable is updated by a thread atomically and appropriately (the condition is actually true). Join() is used to ensure that all the threads have done their part in making the HashMap which can then be used to find the maxOccurences and OneOccurences.

3 Performance Results

There were two ways Performance results were determined. I wrote a test wish displayed a concrete ratio of the Sequential performance for a large text file to a parallel performance for a large text file. A speed up was not observed, it even seemed to slow down. Another way that performance was determined was by examining (provided by the IDE) the times in ms that each test took. Especially the tests that dealt with large text files. These numbers for a sample run are displayed below. The Lesson is Threads can be expensive and require deep thought into how to partition work with them.

Seq	Par
180 ms	366 ms
132 ms	264 ms
33 ms	49 ms