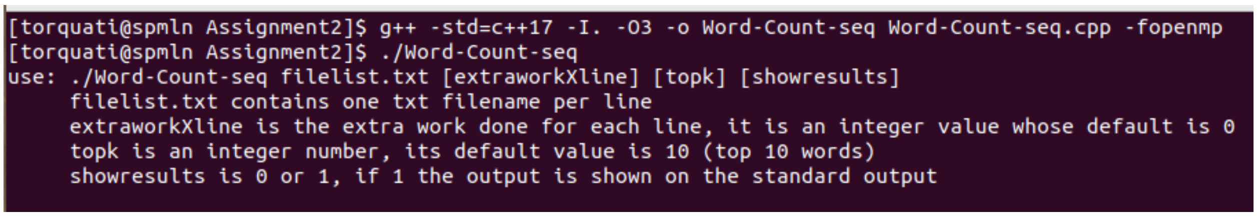
Assignment 2
SPM course a.a. 23/24

April 9, 2024

Counting words: a map-reduce computational pattern on a stream of text files

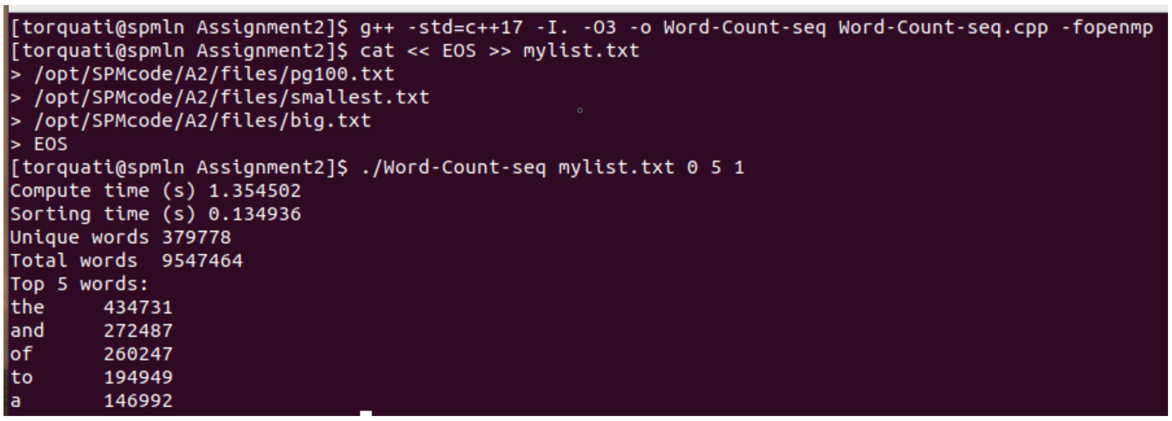
We want to find the top k words of a given set of text files provided. The top k words are the k most frequent words occurring in all input files.

The Word-Count-seq.cpp file gives one possible C++ sequential implementation of this problem.



```
[torquati@spmln Assignment2]$ g++ -std=c++17 -I. -O3 -o Word-Count-seq Word-Count-seq.cpp -fopenmp
[torquati@spmln Assignment2]$ ./Word-Count-seq
use: ./Word-Count-seq filelist.txt [extraworkXline] [topk] [showresults]
     filelist.txt contains one txt filename per line
     extraworkXline is the extra work done for each line, it is an integer value whose default is 0
     topk is an integer number, its default value is 10 (top 10 words)
     showresults is 0 or 1, if 1 the output is shown on the standard output
```

The first argument (filelist.txt) is a filename containing, one per line, the path to other text files whose words we want to count. Let's ignore the extraworkXline optional argument for the moment. The third optional argument topk, sets the value k, which, by default, is set to 10 (i.e., it computes the top 10 words). The fourth optional argument showresults, if set to 1, allows the user to print the outputs of the top k words to the standard output. By default, output results are not printed. For example, consider the following execution:



```
[torquati@spmln Assignment2]$ g++ -std=c++17 -I. -O3 -o Word-Count-seq Word-Count-seq.cpp -fopenmp
[torquati@spmln Assignment2]$ cat << EOS >> mylist.txt
> /opt/SPMcode/A2/files/pg100.txt
> /opt/SPMcode/A2/files/smallest.txt
> /opt/SPMcode/A2/files/big.txt
> EOS
[torquati@spmln Assignment2]$ ./Word-Count-seq mylist.txt 0 5 1
Compute time (s) 1.354502
Sorting time (s) 0.134936
Unique words 379778
Total words  9547464
Top 5 words:
the     434731
and     272487
of      260247
to      194949
a       146992
```

In this run, the program counted all the words in the pg100.txt, smallest.txt, and big.txt files contained in the /opt/SPMcode/A2/files directory (this directory includes 70 text files with a variable number of lines). Then, it printed the top 5 occurrences of the words in all files listed in the mylist.txt file.

The argument extraworkXline (set to 0 by default, and also set to 0 in the example), allows the user to increase the computation granularity associated with each line of each parsed text file.

Implement a parallel map-reduce version of the Word-Count-seq.cpp code using **OpenMP tasks** and considering the file /opt/SPMcode/A2/filelist.txt as the first input argument. Evaluate the parallel version's performance by varying the extraworkXline parameter to the following values: 0 (default), 1000, and 10000.

Once you have the parallel code, **email it to the teachers along with a brief document** (PDF format, max 4 pages) describing how to compile and execute the code, the performance figures obtained, comments, problems faced, solutions adopted, etc. You can use the spmcluster.unipi.it frontend node or the spmnuma.unipi.it machine (or both) for the tests.

Deadline: Two weeks. However, no later than the end of the lectures (end of May 2024)