Word Count Problem & Cross Correlation Calculation

Based on MPI

Filipe Pires | 85122 | filipesnetopires@ua.pt
João Alegria | 85048 | joao.p@ua.pt

University of Aveiro, DETI

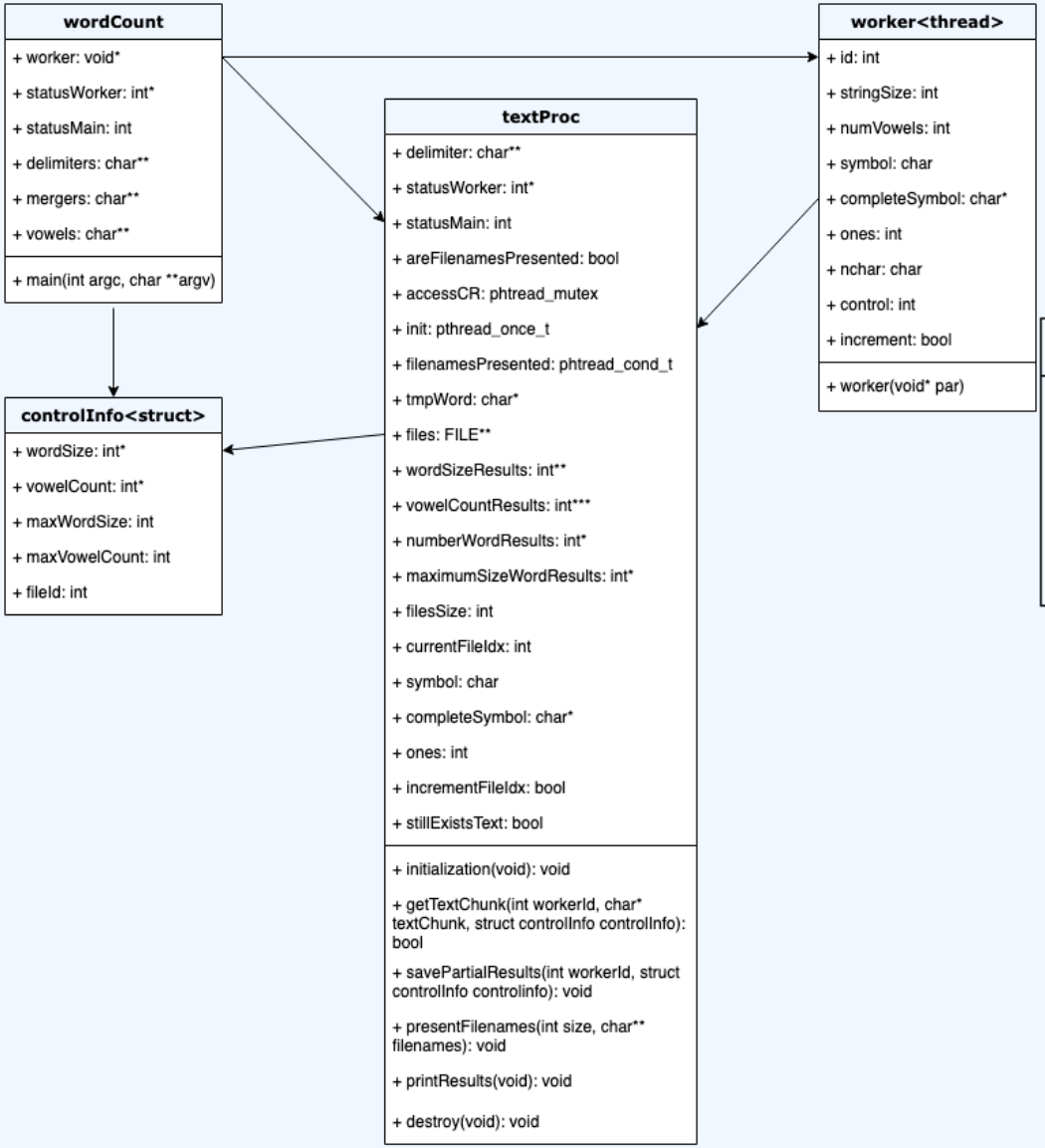
May 11, 2020

Word Count Problem: Multi-Thread Mapping

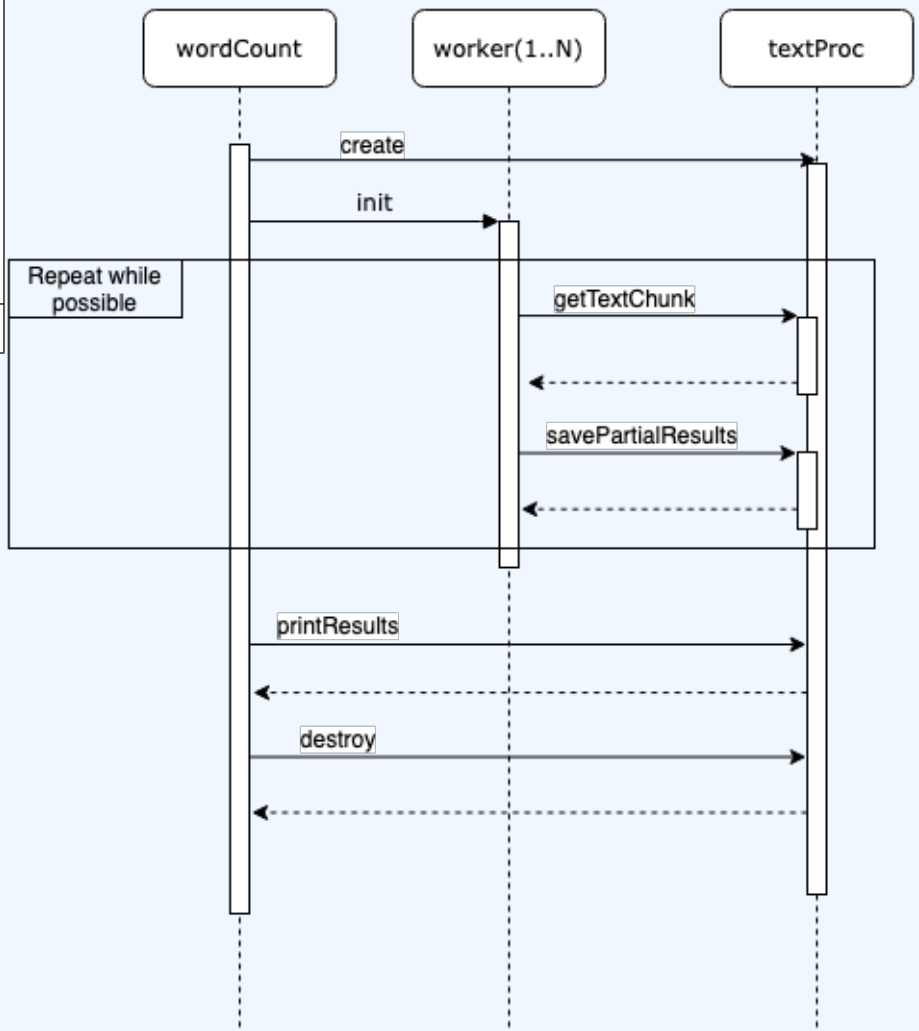
The team efforts were focused on mapping the initial single-threaded implementation of the program to a multi-threaded environment. The required mapping was:

- A shared memory space would keep track of the files to be processed.
- Each worker thread would ask the shared memory for a chunk of text, process it and return the results to the shared memory.
- The shared memory would manage the files' content internally, enabling the distribution of chunks of text.
- The shared memory would keep track of all received results, enabling a print in the end of the global results processing of each file given as input.

Solution Diagram & Entity Interaction



(a) Solution Diagram



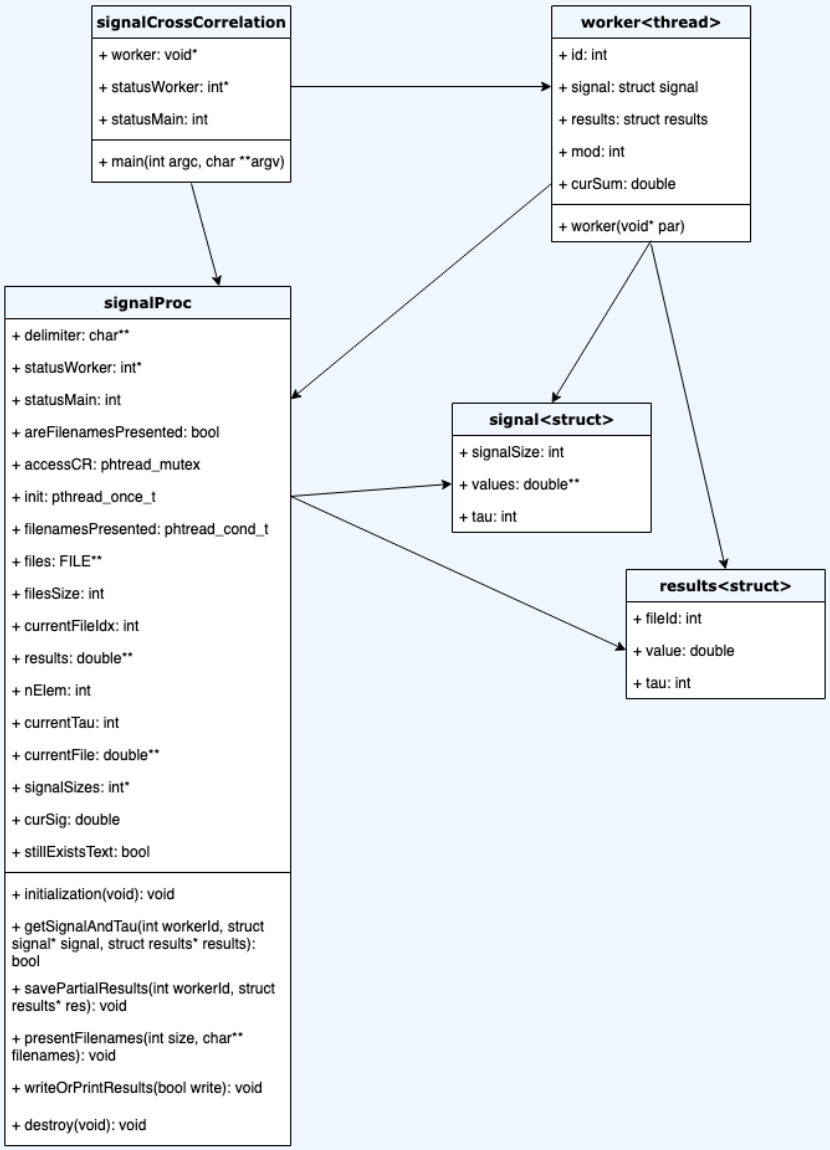
(b) Entity Interactions

Cross Correlation Problem: Multi-Thread Mapping

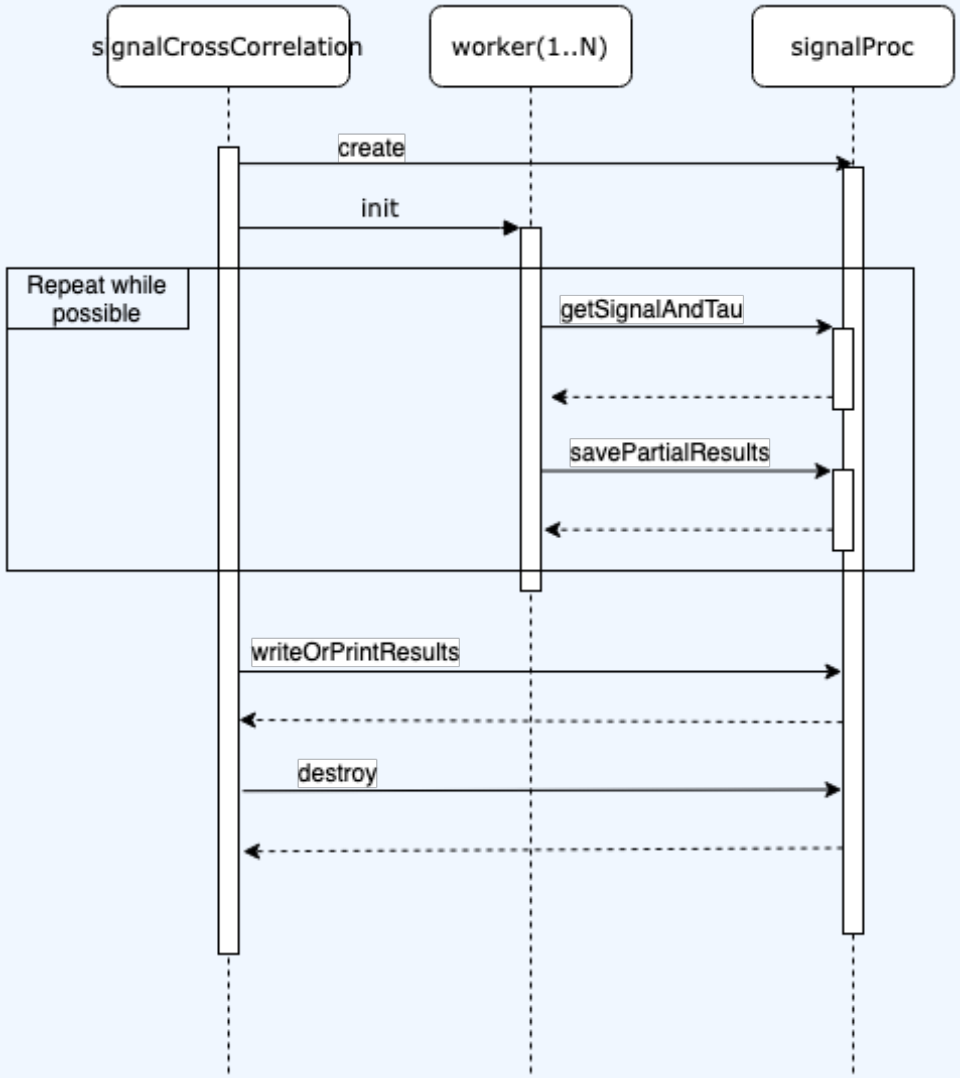
Once again, our task was to map a single-threaded implementation of the solution for this second problem, previously developed by us, to a multi-threaded version of such implementation. The required mapping was:

- A shared memory space would keep track of the files to be processed.
- Each worker thread would ask the shared memory for the values of a signal and a specific τ, calculate the cross correlation and return the results to the shared memory.
- The shared memory would manage the files' content internally, enabling the distribution of the same signals but with different τ values.
- The shared memory would keep track of all received results, enabling the program to write the results in the end of each file or to print them to the console.

Solution Diagram & Entity Interaction



(a) Solution Diagram



(b) Entity Interactions

Results

As expected, the more worker threads the program deploys, the faster it delivers the results.
By analyzing the table below, it is not difficult to see an inverse relation between the number of threads and the execution times.

		Average Execution Time (s)							
		Problem 1				Problem 2			
		text1	text2	text3	text4	sigVal1	sigVal2	sigVal3	sigVal4
# of Threads	1	0.007	0.032	0.014	0.033	0.010	0.089	1.282	20.155
	2	0.005	0.021	0.011	0.018	0.008	0.051	0.675	10.543
	4	0.004	0.018	0.009	0.015	0.004	0.033	0.394	5.981

Table: average execution time of running the programs 10 times for each input file and for each different number of worker threads.

Conclusions

The results of implementing multi-threaded versions of the programs for calculating occurring frequencies of word lengths and computing the circular cross-correlation of pairs of signals clearly showed us the astonishing advantages of such architecture.

By carefully orchestrating the worker threads, we were able to speed up our solutions up to 4 times. But in fact this achievement can be much greater if the hardware contains more CPU cores and therefore supports a higher maximum number of threads.

For future work, one possible improvement would be do adopt different computation approaches. For example, in the case of the cross-correlation computation, multidimensional techniques for decomposition could be studied to find out if they would accelerate the program. Unfortunately, however, this was not possible to be conducted due to external factors beyond our grasp.