Word Count Problem & Cross Correlation Calculation

MPI Implementation

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

University of Aveiro, DETI

May 19, 2020

Multi-Thread to MPI Mapping

Given that a multi-thread solution was already implemented, the team's efforts were focused on mapping the it to the MPI environment. The required mapping was:

- The previous central entity(monitor) was translated to a dispatcher that is responsible for the same functionalities and is implemented in a predefined process created by the MPI platform.
- The dispatcher is responsible for keeping track of files to process, provide data chunk when requested, stored partial results and print final results.
- The previous thread workers are now implemented as process supported by the MPI platform.
- There logic keeps the same, request a data chunk, processing it and sending the results to the collecting entity.

Word Count Entity Interaction

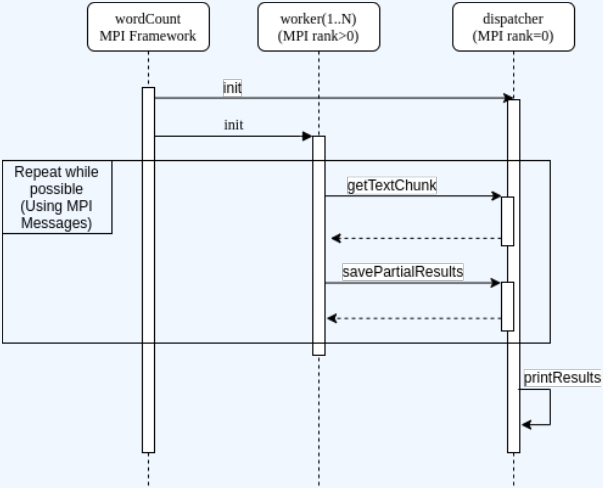

Figure: Entity Interactions

Cross Correlation Entity Interaction

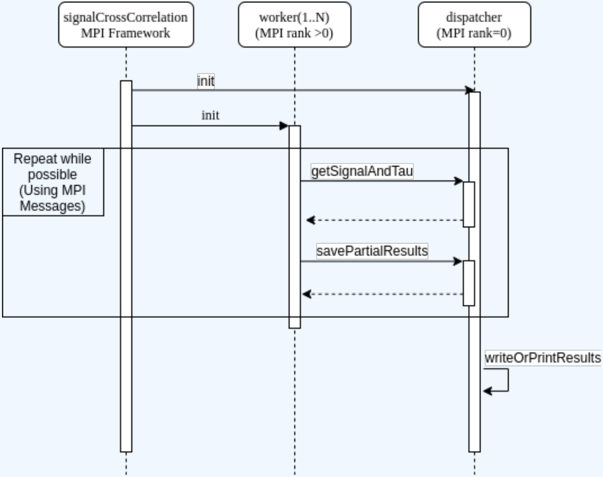
signalCrossCorrelation MPI Framework
worker(1..N) (MPI rank >0)
dispatcher (MPI rank=0)
init
init
Repeat while possible (Using MPI Messages)
getSignalAndTau
savePartialResults
writeOrPrintResults

Figure: Entity Interactions

Results

As expected, the more worker processes the user 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-process versions of the programs using the MPI library for calculating occurring frequencies of word lengths and computing the circular cross-correlation of pairs of signals clearly showed us the astonishing results and proved to be a good option to the multi-thread approach.

By carefully orchestrating the worker processes, 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