Assignment 1
Multithreading

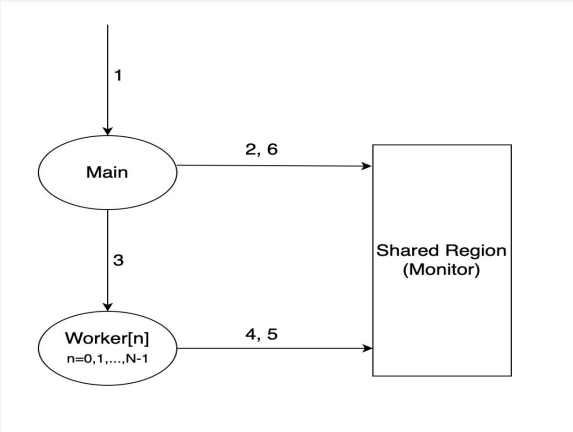
CLE - Computação em Larga Escala
Março 2022

Eduardo Fernandes - 98512
Alexandre Pinto - 98401

Text Processing - Multithreading Implementation

Objective: Given a text file count the total number of words, and also the number of words containing an a, e, i, o, u, y. The multithreading implementation splits the work among the worker threads.

1. Main processes the command line arguments;
2. Main saves the file names and initialize the counters for each file in the shared region;
3. Main creates the worker threads and splits the text files into chunks of text. Stores the chunks in a FIFO in the shared region;
4. Workers get a text chunk at time, process it, and save partial results in the shared region;
5. Repeat until there is no more text chunks;
6. Main finally prints the results for each file.



Text Processing - Results

The results below were obtained with a chunk size of 4096 and processing all text files.

- **1 Worker** (one thread) : Elapsed Time = 0.004699 s;
- **2 Worker** (two threads) : Elapsed Time = 0.002364 s;
- **4 Worker** (four threads) : Elapsed Time = 0.001857 s;
- **8 Worker** (eight threads) : Elapsed Time = 0.001765 s;

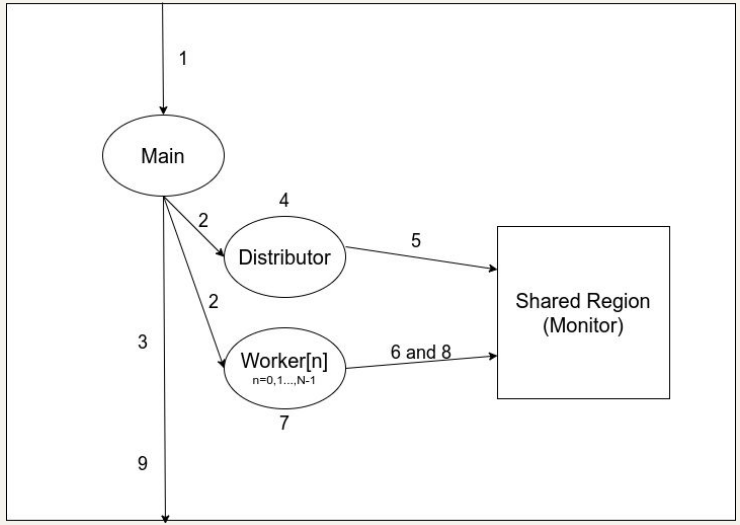
Command:

./prog1 -t (*num_threads*)-f (*files*)

Integers Sorting - Multithreading Implementation

Objective: Given a binary file sort the integers there presented. The multithreading implementation uses a distributor thread to split the work among the worker threads.

1. Main processes the command line arguments;
2. Main creates the distributor and worker threads;
3. Main waits for threads to terminate their work;
4. Distributor read integers from file;
5. Distributor defines sub arrays and distributes work;
6. Workers get work;
7. Workers merge sort work;
8. Workers save work;
9. Main verifies final results;



Integers Sorting - Results

Objective: Given a binary file sort the integers there presented. The multithreading implementation uses a distributor thread to split the work among the worker threads.

Number Threads \ File Name	1	2	4	8	16
dataSeq32.bin	0.000913s	0.001369s	0.002059s	0.0003040s	0.005997s
dataSeq256K.bin	0.056335s	0.049808s	0.065092s	0.099427s	0.128381s

Command:

./prog2 (*num_threads*) (*filename*)

Conclusion

After looking at our results, we came to the following conclusions:

- With the increased number of worker threads, a clear decrease in execution times was registered for program 1;
- With the increased number of worker threads, the execution times were higher;
- This can be due to a small number of integers to sort, because with the bigger file the time was getting closer to the singlet thread implementation.
- In general, multithreaded programs should offer an improvement in the performance when compared to single threaded ones.