Assignment 2:
Multiprocessing

universidade de aveiro

Pedro Silva (93011) & João Soares (93078)
Master in Informatics Engineering
2021/2022

Problem 1 - Text processing in Portuguese: Main Ideas

- **Structures** to facilitate operations:
  - *ChunkText* - Structure of a chunk of text from a file.
  - *FileText* - Structure of File containing number of words, number of words starting with a vowel and number of words ending with a consonant.
  - *ChunkResults* - Structure of Partial Results for a file, contains metrics obtained by processing a single text chunk.
- **Processes**:
  - **Dispatcher**:
    - Command line arguments processing (files to process and number processes).
    - Allocate memory and initialize file results structures(FileText).
    - Initializes and creates threads:
      - Thread that reads successively the file and creates Chunks(ChunkText) to save them in the Shared Region (FIFO).
      - Thread that obtains Chunks from the Shared Region, distributes them to the workers and receives their processing result (*ChunkResults*) and compiles results into file structure(FileText). When all matrices are sent informs the workers no more work to do else informs them that there is still work to do.
    - Print the Results (*FileText*) of processing each file.
  - **Workers**:
    - While there is work (chunk to process) indicated by the Dispatcher, receives a ChunkText, processes the number of words, the number of words starting with a vowel and number of words ending with a consoant and delivers the results for a chunk back to the Dispatcher.

Problem 1 - Text processing in Portuguese: Timing Results

File/	1		2		4		8	
nº Workers	Avg Time	Variance	Avg Time	Variance	Avg Time	Variance	Avg Time	Variance
text0.txt	0.0003962	0.00001	0.00048409090	0.0001	0.0005767272	0.0001	0.02567754545	0.001
text1.txt	0.00066135	0.00001	0.00156336	0.001	0.001300727	0.001	0.03501454	0.001
text2.txt	0.0082443333	0.0001	0.00604409	0.001	0.0051566363	0.001	0.1710736363	0.03
text3.txt	0.00421172	0.001	0.00203063	0.001	0.0022793636	0.001	0.0714108181	0.001
text4.txt	0.0064596363	0.001	0.00517145	0.001	0.0045451818	0.001	0.1466777	0.03
All	0.0106304545	0.001	0.007875272	0.001	0.0089424545	0.001	0.324643636	0.03

- Average Time of 10 consecutive executions with FIFO size: 10000 and ChunkSize 100.
- As the text size increases, the time to calculate metrics increases as well, taking this into account the use of more workers would naturally lead to great benefit in performance. Instead we can observe more dubious results, while using 2 or 4 workers seems lead to some increase in performance in most cases, the use of 6 workers leads to huge decrease in performance across the board.
- We can explain this when we take into account the overhead of using various workers in what is (in comparison) a less resource intensive operation, that is reading a file and counting words.

Problem 2 - Determinant of a Square Matrix : Main Ideas

- **Structures** to facilitate operations:
  - *MatrixResult* - Structure of the Matrix processed Result.
  - *FileMatrices* - File.
  - *Matrix* - Matrix to Process.
- **Processes**:
  - **Dispatcher**:
    - Process Command Line Arguments (file and number processes).
    - Save File Properties (Number of Matrices and Order of the Matrices).
    - Initializes and creates threads:
      - Thread that reads successively the file and creates Matrices (Matrix) to save them in the Shared Region (FIFO).
      - Thread that obtains Matrices from the Shared Region, distributes them to the workers(Isend) and receives (Irecv) their processing result (*MatrixResult*) and saves them after verifying the completion of the operation. When all matrices are sent informs the workers else informs them that there is still work to do.
    - Print the Results (*MatrixResult*) of the processment of the matrices.
  - **Workers**:
    - While there is work (matrix to process) indicated by the Dispatcher, receives a Matrix and after verifying the completion of the sending operation, processes the determinant of the matrix and finally, delivers the result back to the Dispatcher.

Problem 2 - Determinant of a Square Matrix : Timing Results

File/	1		2		4		8	
nº Workers	Avg Time (s)	Variance	Avg Time (s)	Variance	Avg Time (s)	Variance	Avg Time (s)	Variance
mat128_32.bin	0.041535	8,00E-06	0.042265	4,00E-06	0.050958	1.4e-05	0.43929	0.003889
mat128_64.bin	0.045605	2.3e-05	0.042891	4,00E-06	0.055966	1.3e-05	0.367191	0.021085
mat128_128.bin	0.062842	8,00E-06	0.050261	1,00E-05	0.057152	2,00E-06	0.303502	0.012435
mat128_256.bin	0.336044	7,00E-05	0.203318	0.000174	0.191741	1.4e-05	0.45262	0.010653
mat512_32.bin	0.146281	1,00E-06	0.160576	1.8e-05	0.195098	0.000256	1.312314	0.029321
mat512_64.bin	0.154533	0.000135	0.162669	0.000235	0.224619	0.001653	1.276129	0.021468
mat512_128.bin	0.222875	3,00E-06	0.181258	0.000533	0.210663	5.3e-05	1.321057	0.055036
mat512_256.bin	1.391468	0.00397	0.776707	0.000329	0.766861	0.000717	1.743077	0.069582

- Average Time of 10 consecutive executions with FIFO size: 512
- As the matrix order increases, the time to calculate the determinant also increases, so files with higher matrices order benefit to a great extent from using more workers. Still, even with high order matrices, using 8 workers gets significantly worst performance results.
- So in the files with lower order matrices, the processing time is low, which makes workers available quickly and because the communication costs and overhead increase with the use of more workers the execution times will become higher with more workers instead of decreasing which makes less beneficial the use of more workers.
- In the files with higher order matrices, the processing time is higher, as such, the increase in workers benefits to a certain level the performance but the effects of increasing the number of workers still take place, so for more than a certain number of workers the performance also gets significantly worst.

Comparison between Prob 1 and Prob 2

- Comparing the results of the two problems both are similar in results behavior. Up to 2 and 4 workers the performance improves significantly but with 8 workers the performance gets worse, as already mentioned due to the created overhead.
- Although problem 2 was implemented with non-blocking operations, it did not present a significant improvement in the use of a large number of workers similar to problem 1 implemented with synchronous operations.
- Finally, it should be taken into account that problem 1 handles much less data while problem 2 handles much more data, when considering that problem 2 is able to take much more advantage of parallelism in its implementation.