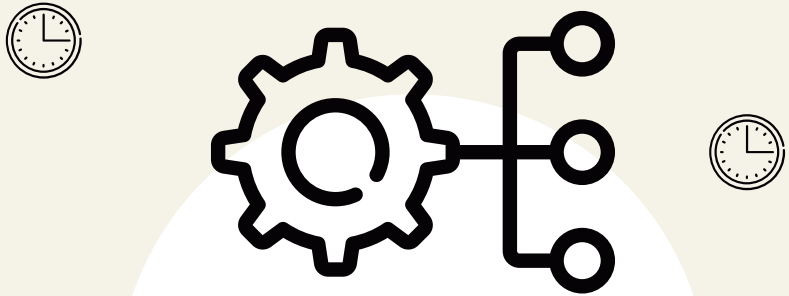
universidade de aveiro

Message Passing Interface

Computação em Larga Escala - Assignment II



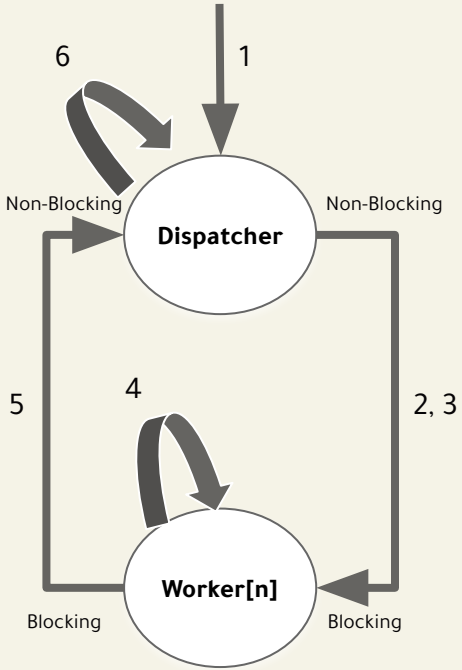
Diogo Magalhães 102470
Rafael Gil 118377

Problem 1 - Implementation

Objective: Count the total number of words and the number of words having at least two equal consonants in one or several text files.

1. Dispatcher processes the command line arguments and stores the files.
2. Dispatcher broadcasts the bufferSize.
3. Dispatcher gets a chunk of data and sends to each worker without blocking.
4. Worker processes the chunks.
5. Worker answers the partial results.
6. Dispatcher joins the partial results and outputs the final results.

For the program to work, we must have at least a dispatcher and one worker, meaning that we need at least two MPI processes.



Problem 1 - Results

OS: Ubuntu **CPU:** 6 core 3.3GHz

Command: mpiexec -n <number-of-processes> ./prog1/main -f dataSet1/text*.txt -b <buffer-size>

	1 worker	*2 workers*	*4 workers*	*8 workers*
1024 bytes	0.003344s	0.002263s	0.001389s	0.001253s
8192 bytes	0.003032s	0.001875s	0.001275s	0.001103s

Command: mpiexec -n <number-of-processes> ./prog1/main -f <file> -b 8192

	1 worker	*2 workers*	*4 workers*	*8 workers*
text0.txt	0.000048s	0.000067s	0.000119s	0.000208s
text2.txt	0.001399s	0.001082s	0.000676s	0.000536s

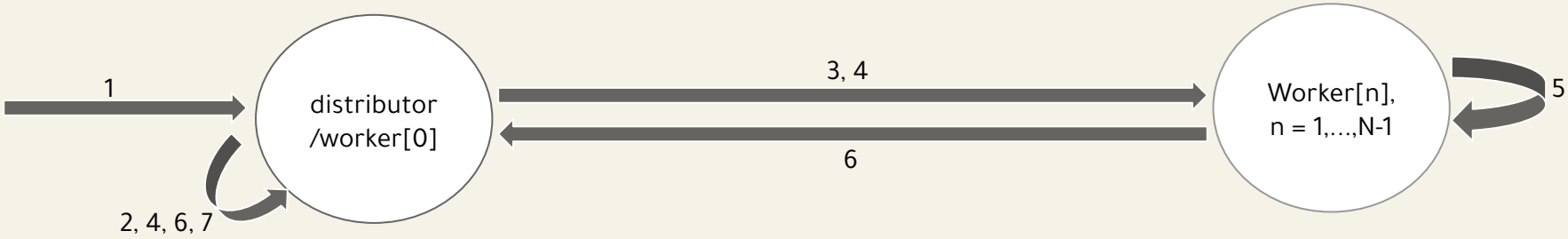
Problem 1 - Conclusions

When we analyse the previous tables, we can conclude that:

- In the first table, it is possible to see the impact of using smaller and bigger buffer sizes. Bigger buffer sizes, usually means less buffer requests from the worker to the dispatcher, so the worker will be less times stopped waiting for the dispatcher, meaning, usually, faster times.
  In our tests, where we changed the buffer size from 1024 to 8192, we got improvements of 9.33% when using 1 worker, 17.15% when using 2 workers, 8.20% when using 4 workers and 8.25% when using 8 workers.
- In the second table, it is observed that depending on the size of the file, the usage of more processes might be prejudicial to the performance of the program since for smaller files processes might be created but not used since all the file would have already been processed by another process.
- Overall, the results obtained and shown in both tables were expected and show that the performance of the text processing improved a lot with the usage of MPI.
  The improvements between analyzing all the files and using a fixed 8192 buffer size where of 38.16% from 1 to 2 workers, 32% from 2 to 4 workers and 13.5% from using 4 to 8 workers, which means an improvement of 63.62% from 1 to 8 workers.

Problem 2 - Implementation

Objective: Sort, in descending order, a sequence of numbers using the bitonic sort algorithm.



1. Distributor validates the command line arguments.
2. Distributor reads the file and stores the data.
3. Distributor sends the length of the number array to the workers (MPI_Bcast).
4. Distributor sends the subsequences to the workers (MPI_Scatter).
5. Workers sort the subsequence.
6. Workers send the sorted subsequence to the distributor (MPI_Gather).
7. Distributor validates the sequence.

Problem 2 - Results

OS: Ubuntu **CPU:** 6 core 3.3GHz

	1 proc	*2 proc*	*4 proc*	*8 proc*
32	0.000027s	0.000152s	0.000198s	0.000739s
256K	0.039346s	0.022066s	0.015245s	0.015199s
1M	0.182225s	0.100645s	0.065384s	0.061635s
16M	4.035222s	2.293201s	1.760282s	1.757250s

Problem 2 - Conclusions

When we analyse the time table, we can conclude that:

- Even though more processes lead to a reduction in execution time, the rate of improvement decrease as the number of processes increase, seen by the disparity between the execution times of 1 to 2 workers and between 4 to 8 workers.
- When processing files of smaller dimension, it is not efficient to use several processes, as it can even lead to losing performance as shown in the row of the datSeq32.bin file.
- After 4 processes, the time reduction by using more processes becomes almost unnoticeable.
- Overall, the results were expected. The observed improvements, for the 16M file, where of 43.17% from 1 to 2 workers, 23.24% from 2 to 4 workers and 0.17% from using 4 to 8 workers, which means an improvement of 56.45% from 1 to 8 workers.