Nicolás Vila - 240230

Guillem Escriba- 242123

Introduction to Parallel and Distributed Programming

Report Lab 2

Exercise 1 — MPI Communicators and Groups. 20%

```
[u188331@ip-10-49-0-109 1_hello_world]$ cat out_helloworld.out

PHASE 1

Hi, I am rank 0. My communicator is MPI_COMM_WORLD and has a size of 16 processes
Hi, I am rank 1. My communicator is MPI_COMM_WORLD and has a size of 16 processes
Hi, I am rank 2. My communicator is MPI_COMM_WORLD and has a size of 16 processes
Hi, I am rank 3. My communicator is MPI_COMM_WORLD and has a size of 16 processes
Hi, I am rank 4. My communicator is MPI_COMM_WORLD and has a size of 16 processes
Hi, I am rank 5. My communicator is MPI_COMM_WORLD and has a size of 16 processes
Hi, I am rank 6. My communicator is MPI_COMM_WORLD and has a size of 16 processes
Hi, I am rank 7. My communicator is MPI_COMM_WORLD and has a size of 16 processes
Hi, I am rank 8. My communicator is MPI_COMM_WORLD and has a size of 16 processes
Hi, I am rank 9. My communicator is MPI_COMM_WORLD and has a size of 16 processes
Hi, I am rank 10. My communicator is MPI_COMM_WORLD and has a size of 16 processes
Hi, I am rank 11. My communicator is MPI_COMM_WORLD and has a size of 16 processes
Hi, I am rank 12. My communicator is MPI_COMM_WORLD and has a size of 16 processes
Hi, I am rank 13. My communicator is MPI_COMM_WORLD and has a size of 16 processes
Hi, I am rank 14. My communicator is MPI_COMM_WORLD and has a size of 16 processes
Hi, I am rank 15. My communicator is MPI_COMM_WORLD and has a size of 16 processes

PHASE 2:

Hi, I was rank 0 in communicator MPI_COMM_WORLD which had 16 processes. Now I'm rank 0 in communicator SPLIT_COMM_0 which has 4 processes
Hi, I was rank 1 in communicator MPI_COMM_WORLD which had 16 processes. Now I'm rank 1 in communicator SPLIT_COMM_0 which has 4 processes
Hi, I was rank 2 in communicator MPI_COMM_WORLD which had 16 processes. Now I'm rank 2 in communicator SPLIT_COMM_0 which has 4 processes
Hi, I was rank 3 in communicator MPI_COMM_WORLD which had 16 processes. Now I'm rank 3 in communicator SPLIT_COMM_0 which has 4 processes
Hi, I was rank 4 in communicator MPI_COMM_WORLD which had 16 processes. Now I'm rank 0 in communicator SPLIT_COMM_1 which has 4 processes
Hi, I was rank 5 in communicator MPI_COMM_WORLD which had 16 processes. Now I'm rank 1 in communicator SPLIT_COMM_1 which has 4 processes
Hi, I was rank 6 in communicator MPI_COMM_WORLD which had 16 processes. Now I'm rank 2 in communicator SPLIT_COMM_1 which has 4 processes
Hi, I was rank 7 in communicator MPI_COMM_WORLD which had 16 processes. Now I'm rank 3 in communicator SPLIT_COMM_1 which has 4 processes
Hi, I was rank 8 in communicator MPI_COMM_WORLD which had 16 processes. Now I'm rank 0 in communicator SPLIT_COMM_2 which has 4 processes
Hi, I was rank 9 in communicator MPI_COMM_WORLD which had 16 processes. Now I'm rank 1 in communicator SPLIT_COMM_2 which has 4 processes
Hi, I was rank 10 in communicator MPI_COMM_WORLD which had 16 processes. Now I'm rank 2 in communicator SPLIT_COMM_2 which has 4 processes
Hi, I was rank 11 in communicator MPI_COMM_WORLD which had 16 processes. Now I'm rank 3 in communicator SPLIT_COMM_2 which has 4 processes
Hi, I was rank 12 in communicator MPI_COMM_WORLD which had 16 processes. Now I'm rank 0 in communicator SPLIT_COMM_3 which has 4 processes
Hi, I was rank 13 in communicator MPI_COMM_WORLD which had 16 processes. Now I'm rank 1 in communicator SPLIT_COMM_3 which has 4 processes
Hi, I was rank 14 in communicator MPI_COMM_WORLD which had 16 processes. Now I'm rank 2 in communicator SPLIT_COMM_3 which has 4 processes
Hi, I was rank 15 in communicator MPI_COMM_WORLD which had 16 processes. Now I'm rank 3 in communicator SPLIT_COMM_3 which has 4 processes

PHASE 3

Hi, I was rank 0 in communicator SPLIT_COMM_2 which had 4 processes. Now I'm rank 4 in communicator EVEN_COMM which has 8 processes.
Hi, I was rank 2 in communicator SPLIT_COMM_2 which had 4 processes. Now I'm rank 5 in communicator EVEN_COMM which has 8 processes.
Hi, I was rank 0 in communicator SPLIT_COMM_3 which had 4 processes. Now I'm rank 6 in communicator EVEN_COMM which has 8 processes.
Hi, I was rank 0 in communicator SPLIT_COMM_0 which had 4 processes. Now I'm rank 0 in communicator EVEN_COMM which has 8 processes.
Hi, I was rank 2 in communicator SPLIT_COMM_0 which had 4 processes. Now I'm rank 1 in communicator EVEN_COMM which has 8 processes.
Hi, I was rank 0 in communicator SPLIT_COMM_1 which had 4 processes. Now I'm rank 2 in communicator EVEN_COMM which has 8 processes.
Hi, I was rank 2 in communicator SPLIT_COMM_1 which had 4 processes. Now I'm rank 3 in communicator EVEN_COMM which has 8 processes.
Hi, I was rank 2 in communicator SPLIT_COMM_3 which had 4 processes. Now I'm rank 7 in communicator EVEN_COMM which has 8 processes.

PHASE 4

Hi, I was rank 1 in communicator MPI_COMM_WORLD which had 16 processes. Now I'm rank 0 in communicator ODD_COMM which has 8 processes.
Hi, I was rank 3 in communicator MPI_COMM_WORLD which had 16 processes. Now I'm rank 1 in communicator ODD_COMM which has 8 processes.
Hi, I was rank 5 in communicator MPI_COMM_WORLD which had 16 processes. Now I'm rank 2 in communicator ODD_COMM which has 8 processes.
Hi, I was rank 7 in communicator MPI_COMM_WORLD which had 16 processes. Now I'm rank 3 in communicator ODD_COMM which has 8 processes.
Hi, I was rank 9 in communicator MPI_COMM_WORLD which had 16 processes. Now I'm rank 4 in communicator ODD_COMM which has 8 processes.
Hi, I was rank 11 in communicator MPI_COMM_WORLD which had 16 processes. Now I'm rank 5 in communicator ODD_COMM which has 8 processes.
Hi, I was rank 13 in communicator MPI_COMM_WORLD which had 16 processes. Now I'm rank 6 in communicator ODD_COMM which has 8 processes.
Hi, I was rank 15 in communicator MPI_COMM_WORLD which had 16 processes. Now I'm rank 7 in communicator ODD_COMM which has 8 processes.
[u188331@ip-10-49-0-109 1_hello_world]$
```

As we can see from the screenshot, the output of the program printed in the terminal is similar to the objective of this exercise with the only exception of the print order that may vary but it is because we have avoided the use of sleeps giving to the code a better performance and a faster execution time.

Nicolás Vila - 240230
Guillem Escriba- 242123
Introduction to Parallel and Distributed Programming

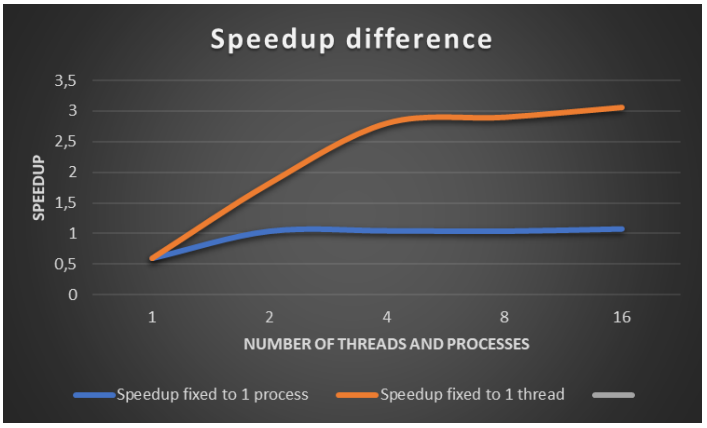
Universitat Pompeu Fabra Barcelona

Exercise 2 — MPI I/O and global communications. 20%

Number of Threads	Average execution time (fixed to 1 Process)
1	0,5902424
2	0,5677902
4	0,564273
8	0,5675362
16	0,5477618

Number of processes	Average execution time (fixed to 1 thread)
1	0,5902424
2	0,3251788
4	0,2111284
8	0,2040924
16	0,1933696

We can clearly observe that when we fixed the number of threads to 1 and increased the number of processes, the strong speedup was considerably higher than in the other case. In fact, when we parallelized the code just by increasing the number of processes we observe that the average execution time was around 3 times faster. In the case of changing the number of threads, however we noticed just a slight improvement.



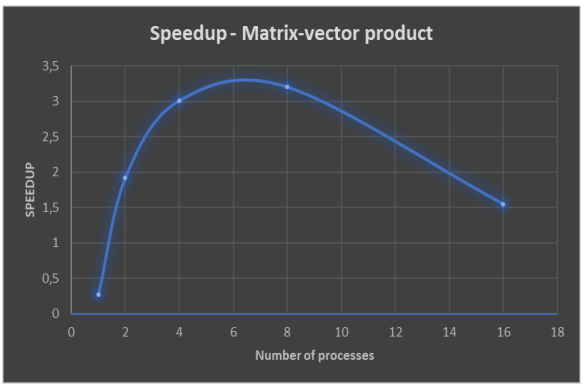
If we run the code with 2 processes and 12 threads, we obtain an average execution time of 0,324170 seconds. On the other hand, when we try to run the code with 4 processes and 6 processes we obtain an average execution time of 0,197551 seconds. Therefore, the last case is substantially better than the first one which is a result that we already expected.

Analyzing the speedup in the previous question we observed that increasing the number of processes made our code perform better than when we only increased the number of threads.
That's why when we double the number of processes (from 2 to 4), even having decreased by a half the number of threads, we checked that the performance was better in the latter case.

Nicolás Vila - 240230
Guillem Escriba- 242123
Introduction to Parallel and Distributed Programming

Exercise 3 — MPI I/O and global communications. 20%

Number of Processes	Average running time
1	0,266378
2	0,13870825
4	0,08842925
8	0,083153
16	0,17183075



As we can see in the strong speedup plot above, the execution time of the matrix-vector product code decreased significantly when we used 2 and 4 threads. Moreover, when we tried using 8 processes, the execution time also decreased in comparison to the 4 process case but not as much. Finally, when we executed the code with 16 processes we observed a large increase in the execution time which seemed odd to us. We believe that this happened because the overhead of managing all the different processes and sharing the vector between every process and gathering the information to obtain the final result must be higher than the time we gain by parallelizing the code with that many number processes.

Exercise 4 — Data types and global communications. 20%

- 4 processes

```
Initial Matrix (rank 0)
0 0 0 0
0 0 0 0
0 0 0 0
0 0 0 0


Final Matrix (rank 0)
0 0 0 0
1 1 1 1
2 2 2 2
3 3 3 3
```

- 8 processes

```
Initial Matrix (rank 0)
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0


Final Matrix (rank 0)
0 0 0 0 0 0 0 0
1 1 1 1 1 1 1 1
2 2 2 2 2 2 2 2
3 3 3 3 3 3 3 3
4 4 4 4 4 4 4 4
5 5 5 5 5 5 5 5
6 6 6 6 6 6 6 6
7 7 7 7 7 7 7 7
```

Nicolás Vila - 240230
Guillem Escriba- 242123
Introduction to Parallel and Distributed Programming

upf. Universitat Pompeu Fabra Barcelona

Exercise 5 — Point to point communications. 20%

After the execution of both files with 1 and 8 processes we have got the following results:

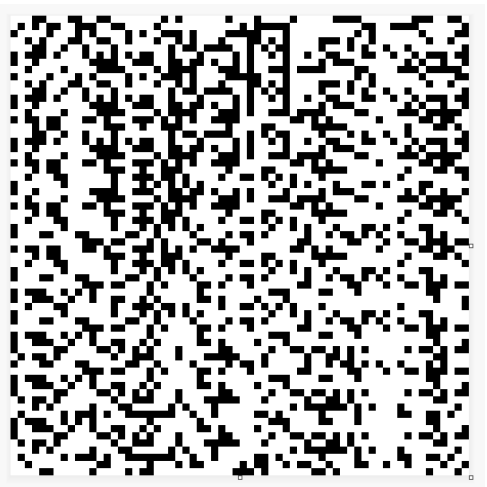
Number of Processes / File	life1.bin	life2.bin
1	0.023788s	0.073321s
8	1.286837s	0.341709s

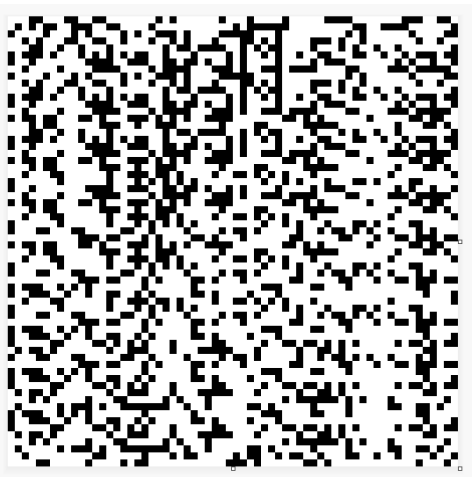
We can see that the first file is faster to execute with only one process. This is because the input file is not big enough to make necessary the communication between processes and the overhead that this generates is bigger than what is gained after parallelizing the code. For this reason 1 process is faster because it avoids that overhead. But the situation changes for the second file because of the size of the input. In this case, the overhead is minimal compared to the total amount of work, and as we can see in the table, the 8 processes version is about 4 times faster than the 1 process version.

In the following images we can see the different inputs and the corresponding outputs, both outputs (1 and 8 processes) as well as the inputs are the same.

Bitmap life1.bin (1 Process vs 8 Processes):

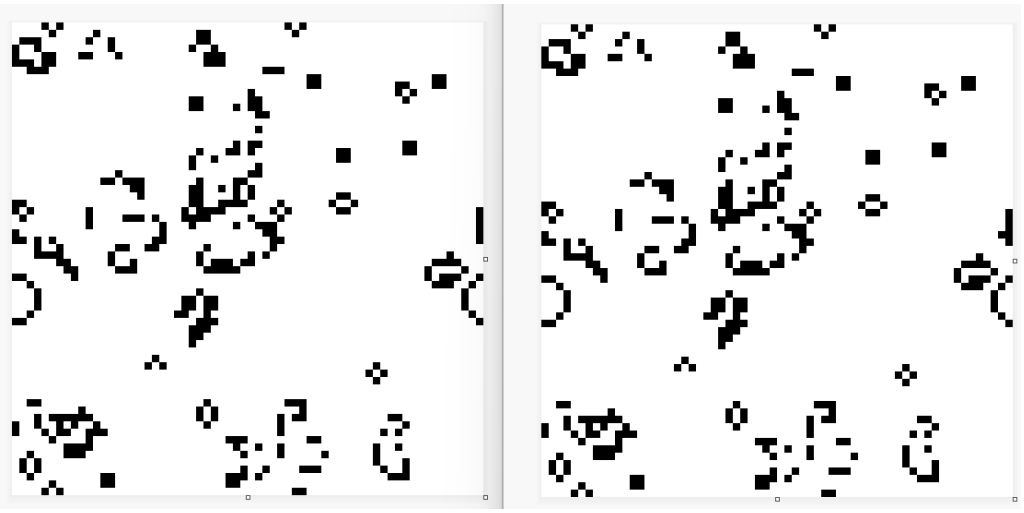
START:





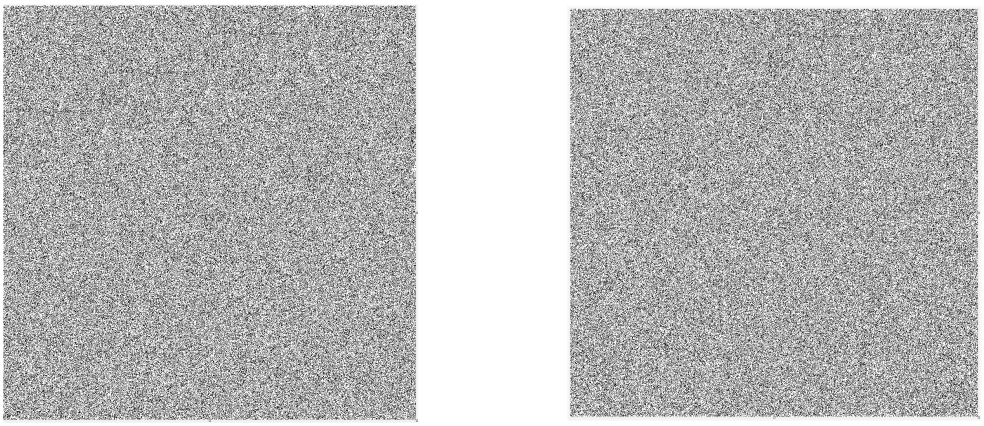
Nicolás Vila - 240230
Guillem Escriba- 242123
Introduction to Parallel and Distributed Programming

END:



Bitmap life2.bin (1 Process vs 8 Processes):

START:



END: