PARALLEL SORT INVESTIGATION

CS4402 2023/24

NAME: Ines Roman Gracia

STUDENT ID: 123123969

INDEX

INTRODUCTION

In this paper we are going to develop an investigation work on the direct, bucket, odd-even and bitonic sorting algorithms that have been implemented in C using the MPI library for their parallelization. The execution is carried out on my personal computer with 8 processors. We will present the execution times of the algorithms when sorting 10,000,000 of data when executing the different algorithms with 1, 2, 4 and 8 processors, we will analyze the communication and computation times of the processors as well as the overall execution time and finally we will compare the performance of the 4 algorithms.

DIRECT SORTING

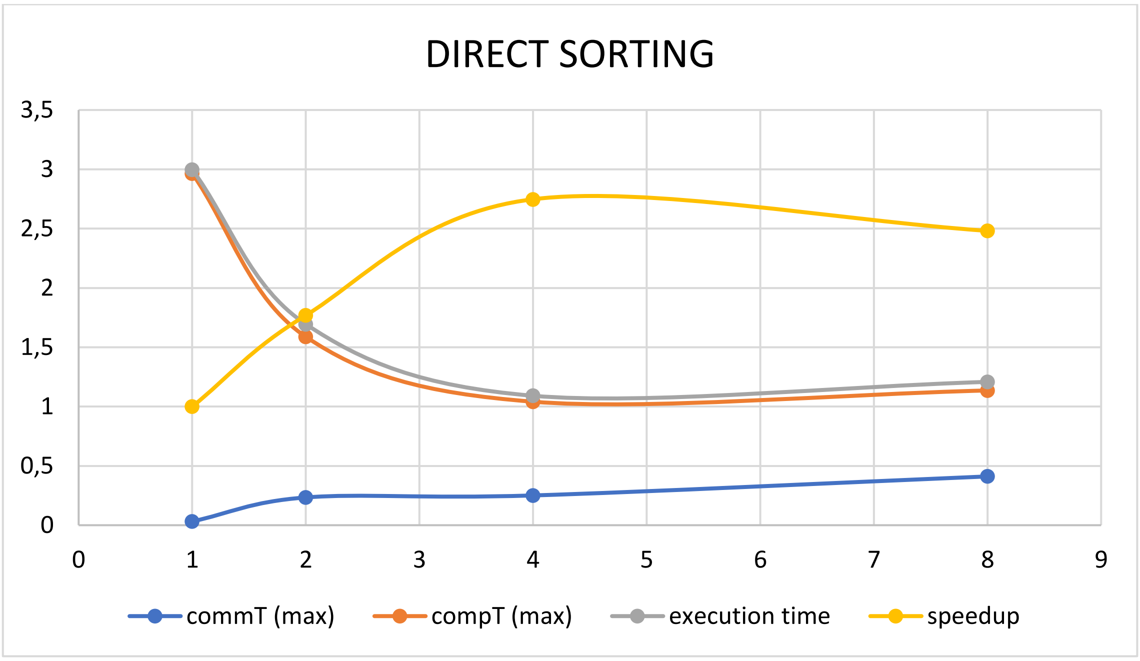
```
mpirun -np 1 ./MPI_Sort_direct
Processor 0: commT = 0.032216, compT = 2.965444
Overall Exec time = 2.997684 with 1 procs

mpirun -np 2 ./MPI_Sort_direct
Processor 1: commT = 0.234069, compT = 1.460391
Processor 0: commT = 0.041757, compT = 1.588057
Overall Exec time = 1.694489 with 2 procs

mpirun -np 4 ./MPI_Sort_direct
Processor 1: commT = 0.240635, compT = 0.775963
Processor 3: commT = 0.245295, compT = 0.777036
Processor 2: commT = 0.250846, compT = 0.778753
Processor 0: commT = 0.050195, compT = 1.040853
Overall Exec time = 1.091077 with 4 procs

mpirun -np 8 ./MPI_Sort_direct
Processor 1: commT = 0.385507, compT = 0.515597
Processor 7: commT = 0.392360, compT = 0.514264
Processor 3: commT = 0.391397, compT = 0.515579
Processor 5: commT = 0.395371, compT = 0.513068
Processor 2: commT = 0.396711, compT = 0.516858
Processor 6: commT = 0.407043, compT = 0.512976
Processor 4: commT = 0.411218, compT = 0.517053
Processor 0: commT = 0.072202, compT = 1.135827
Overall Exec time = 1.208202 with 8 procs
```

A few observations of the output obtained by running the code with 1, 2, 4 and 8 processors are:

- When it’s executed in only one processor the communication time is not zero because the processor is still calling the MPI functions on the code.
- Communication time is much lower for processor 0 than the rest of the processors. This is due to Processor 0 being responsible for scattering the array.

- Computation time becomes higher for Processor 0 while it decreases for the rest of the processors since it’s the responsible for merging the local arrays gathered at the end of the program.

processors	commT (max)	compT (max)	execution time
1	0.032216	2.965444	2.997684
2	0.234069	1.588057	1.694489
4	0.250846	1.040853	1.091077
8	0.411218	1.135827	1.208202



As mentioned earlier, we can observe from the graphs that computation and execution times decrease significantly as we transition from 1 processor to 2 processors and further to 4 processors. However, there is a slight increase in these times when using 8 processors. Also, the increase in communication time is more pronounced when moving from 4 processors to 8, compared to the increase from 2 to 4 processors. Consequently, this leads to the speedup reaching its peak when executed with 4 processors, after which it decreases when employing more processors.

BUCKET SORTING

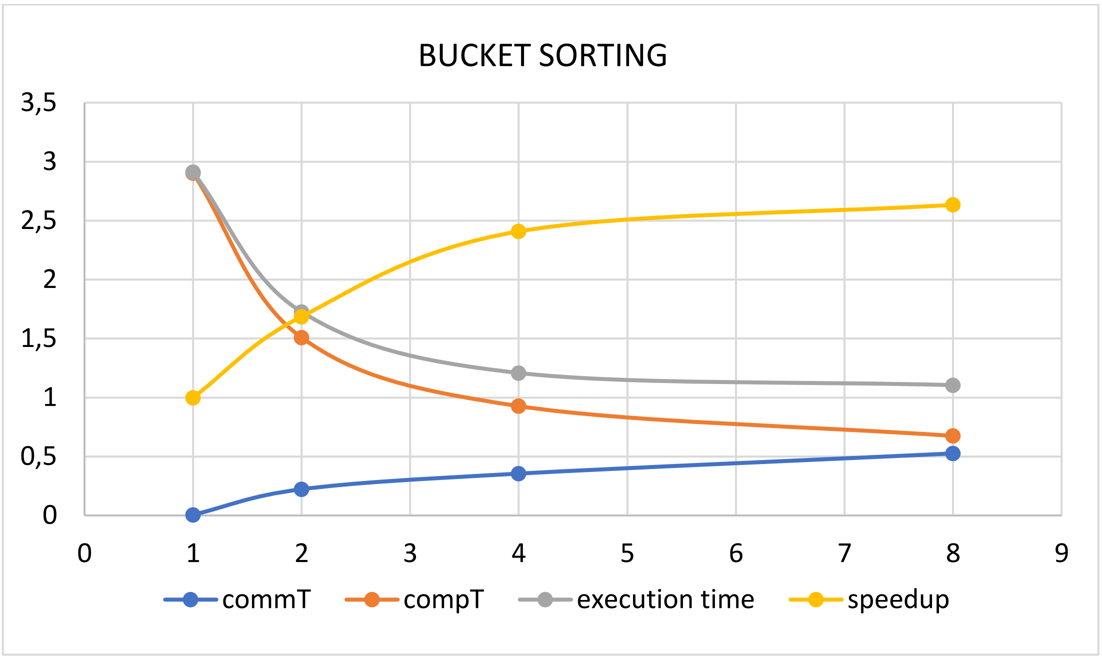
```
mpirun -np 1 ./MPI_Sort_bucket
Processor 0: commT = 0.005869, compT = 2.904857
Overall Exec time = 2.910751 with 1 procs

mpirun -np 2 ./MPI_Sort_bucket
Processor 0: commT = 0.036354, compT = 1.508169
Processor 1: commT = 0.222414, compT = 1.503687
Overall Exec time = 1.726125 with 2 procs
```

```
mpirun -np 4 ./MPI_Sort_bucket
Processor 2: commT = 0.276686, compT = 0.927215
Processor 3: commT = 0.354758, compT = 0.853512
Processor 1: commT = 0.314441, compT = 0.893991
Processor 0: commT = 0.147039, compT = 0.859503
Overall Exec time = 1.208452 with 4 procs

mpirun -np 8 ./MPI_Sort_bucket
Processor 7: commT = 0.525861, compT = 0.578714
Processor 5: commT = 0.447122, compT = 0.657528
Processor 6: commT = 0.467693, compT = 0.636920
Processor 1: commT = 0.476305, compT = 0.628489
Processor 4: commT = 0.430157, compT = 0.674695
Processor 2: commT = 0.457660, compT = 0.647197
Processor 3: commT = 0.438975, compT = 0.666020
Processor 0: commT = 0.213236, compT = 0.602662
Overall Exec time = 1.105088 with 8 procs
```

Several key observations can be made from the output results of the Bucket Sorting MPI program:

- When executed on a single processor, communication time remains non-zero since the processor is still invoking MPI functions within the code.
- Communication time is notably reduced for Processor 0 compared to the other processors, primarily because it is responsible for broadcasting the array at the beginning.
- Computation times remain consistent across all processors and consistently decrease as more processors are added for the parallelization of the sorting process.

processors	**commT**	**compT**	**execution time**
1	0.005869	2.904857	2.910751
2	0.222414	1.508169	1.726125
4	0.35475	0.927215	1.208452
8	0.525861	0.674695	1.105088



The Bucket Sorting program works well with more processors. The speedup is impressive, reaching about 2.63 with 8 processors. Communication time increases slightly with more processors but is still well managed and computation time consistently goes down when adding more processors.

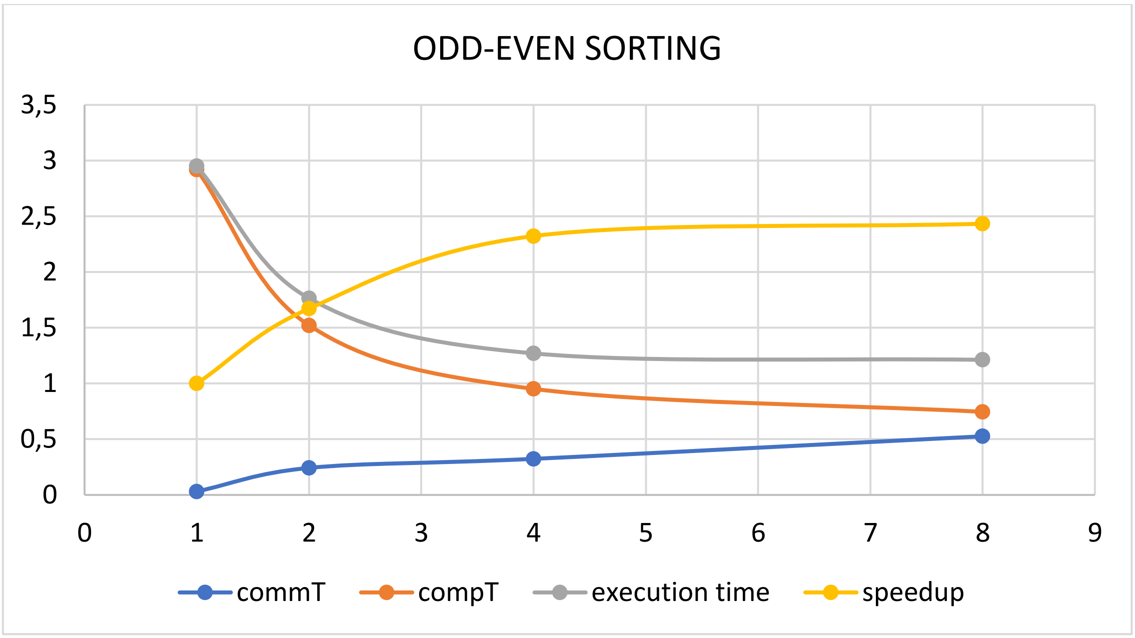
ODD-EVEN SORTING

```
mpirun -np 1 ./MPI_Sort_oddeven
Processor 0: commT = 0.031534, compT = 2.918537
Overall Exec time = 2.950115 with 1 procs

mpirun -np 2 ./MPI_Sort_oddeven
Processor 0: commT = 0.067220, compT = 1.518431
Processor 1: commT = 0.242373, compT = 1.522184
Overall Exec time = 1.764643 with 2 procs

mpirun -np 4 ./MPI_Sort_oddeven
Processor 1: commT = 0.308242, compT = 0.951757
Processor 3: commT = 0.323724, compT = 0.896022
Processor 0: commT = 0.131974, compT = 0.893144
Processor 2: commT = 0.317952, compT = 0.951453
Overall Exec time = 1.269966 with 4 procs

mpirun -np 8 ./MPI_Sort_oddeven
Processor 1: commT = 0.444432, compT = 0.741668
Processor 3: commT = 0.479880, compT = 0.699463
Processor 5: commT = 0.467063, compT = 0.712339
Processor 7: commT = 0.525879, compT = 0.663077
Processor 2: commT = 0.473377, compT = 0.713905
Processor 6: commT = 0.456974, compT = 0.745683
Processor 0: commT = 0.268740, compT = 0.651488
Processor 4: commT = 0.510135, compT = 0.693199
Overall Exec time = 1.212250 with 8 procs
```

Here are some observations based on the results of the MPI Odd-Even Sorting program:

- Communication time remains non-zero even when executed with a single processor, as it still invokes MPI functions within the code.
- Communication time is somewhat lower for Processor 0, primarily because it handles the scattering of the array among the other processors.
- Computation time is consistent across all processors and continues to decrease efficiently as more processors are added.

processors	commT	compT	execution time
1	0.031534	2.918537	2.950115
2	0.242373	1.522184	1.764643
4	0.323724	0.951757	1.269966
8	0.525879	0.745683	1.212250



In summary, the MPI Odd-Even Sorting program demonstrates effective parallelization with decreasing execution times and significant speedup (approximately 2.43 with 8 processors). Communication time increases with more processors but remains manageable, and computation time decreases uniformly, showcasing the program's efficient distribution of sorting tasks.

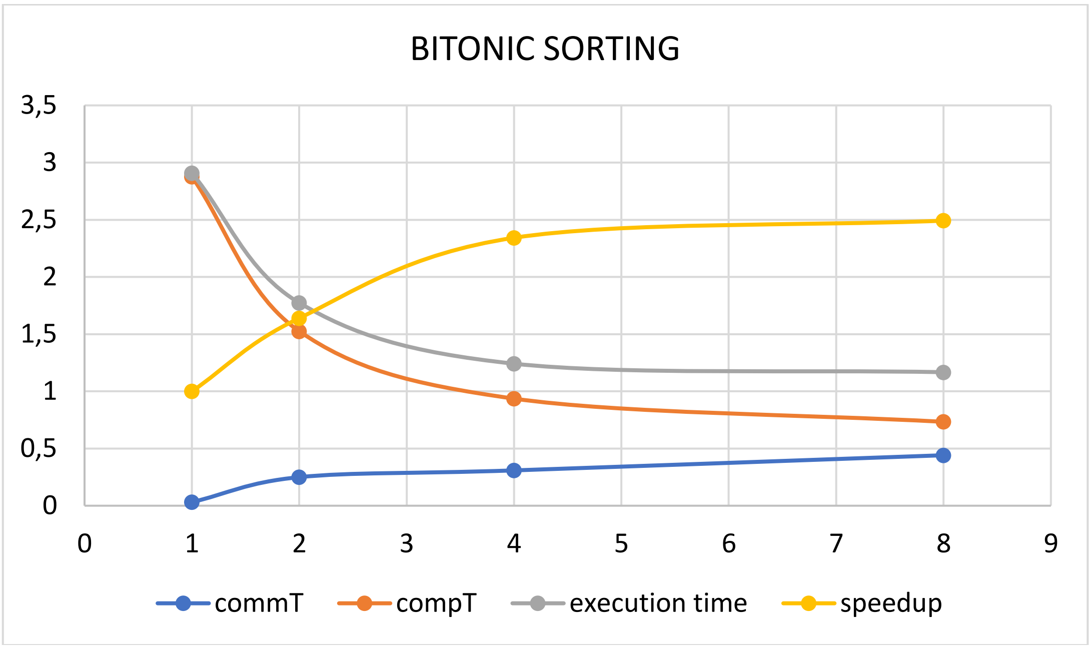
BITONIC SORTING

```
mpirun -np 1 ./MPI_Sort_bitonic
Processor 0: commT = 0.031434, compT = 2.877169
Overall Exec time = 2.908629 with 1 procs

mpirun -np 2 ./MPI_Sort_bitonic
Processor 0: commT = 0.064761, compT = 1.525566
Processor 1: commT = 0.249691, compT = 1.524100
Overall Exec time = 1.773834 with 2 procs

mpirun -np 4 ./MPI_Sort_bitonic
Processor 1: commT = 0.308867, compT = 0.923507
Processor 3: commT = 0.298851, compT = 0.936237
Processor 0: commT = 0.113570, compT = 0.923802
Processor 2: commT = 0.308810, compT = 0.932808
Overall Exec time = 1.241697 with 4 procs

mpirun -np 8 ./MPI_Sort_bitonic
Processor 7: commT = 0.425629, compT = 0.722740
Processor 5: commT = 0.424335, compT = 0.724755
Processor 1: commT = 0.427835, compT = 0.722028
Processor 3: commT = 0.417961, compT = 0.732216
Processor 2: commT = 0.423451, compT = 0.733497
Processor 6: commT = 0.435152, compT = 0.724829
Processor 0: commT = 0.165426, compT = 0.717049
Processor 4: commT = 0.441300, compT = 0.725627
Overall Exec time = 1.167180 with 8 procs
```

Here are some observations of the output results for running the MPI Bitonic Sorting program:

- Even when the program is executed with a single processor, the communication time remains non-zero because the program continues to make use of MPI functions in its code.
- Processor 0 experiences relatively lower communication time, primarily because it is responsible for scattering the array to the other processors.
- Computation time is consistent across all processors and efficiently decreases as more processors are added for parallel processing.

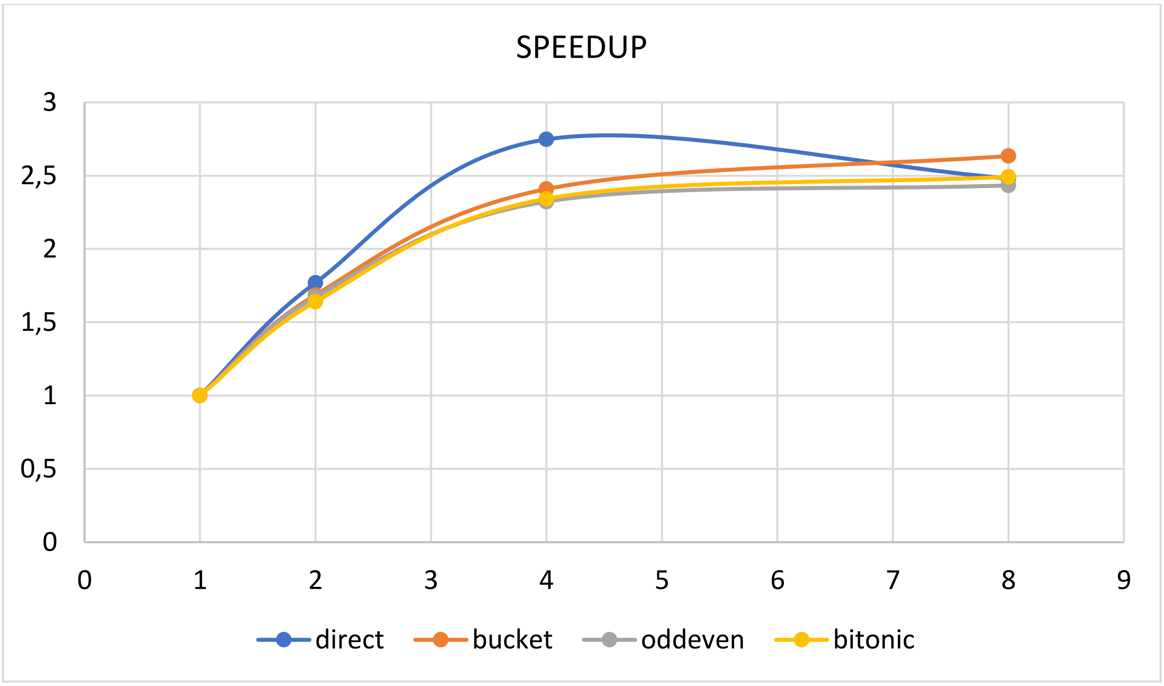
processors	**commT**	**compT**	**execution time**
1	0.031434	2.877169	2.908629
2	0.249691	1.525566	1.773834
4	0.308867	0.936237	1.241697
8	0.441300	0.733497	1.167180



We can notice a resemblance between the graph for Bitonic sorting and the one for Odd-Even sorting. The execution, communication, and computation times are slightly lower, yet they follow a similar pattern.

FINAL NOTES

The Direct sorting algorithm is the only one that is not suitable for parallelization because the speedup is higher for 4 processors than for 8. Nevertheless, it is the algorithm that achieved the best time for sorting 10,000,000 data when executed with 4 processors.



The other three algorithms - bucket, odd-even, and bitonic - benefit greatly from parallelization. In all three graphs, we can observe that their communication, computation, and execution times follow a similar trend. Despite their similar performance, the algorithm that worked most efficiently when sorting 10,000,000 data for all executions with different numbers of processors is bucket sorting, followed by bitonic, and lastly, odd-even. It's worth noting that, bucket sorting is the only one among the three algorithms that has a limitation, which is that we need to know the range of values we are going to sort.