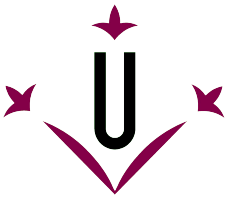
103084 - High Performance Computing

Parallel Becnhmarking Practice

Jordi Rafael Lazo Florensa

23 March 2023

Master's degree in Computer Engineering



Universitat de Lleida
Escola Politècnica Superior

1 Analysis of the chosen benchmark

The NAS Parallel Benchmarks (NPB) are a small set of programs designed to help evaluate the performance of parallel supercomputers. The benchmarks are derived from computational fluid dynamics (CFD) applications and consist of five kernels and three pseudo-applications in the original "pencil-and-paper" specification (NPB 1). The benchmark suite has been extended to include new benchmarks for unstructured adaptive meshes, parallel I/O, multi-zone applications, and computational grids. Problem sizes in NPB are predefined and indicated as different classes. Reference implementations of NPB are available in commonly-used programming models like MPI and OpenMP (NPB 2 and NPB 3).

- FT benchmark: FT (Floating-Point Test) benchmark is a benchmark that it is designed to measure the performance of floating-point operations.

  One of the primary reasons for choosing the FT benchmark is that this benchmark is designed to test the performance of various mathematical functions, such as sine, cosine, logarithms, and exponentials, which are commonly used in scientific and engineering applications. Therefore, measuring the FT benchmark can provide valuable insights into the performance of a computer system in these types of applications.

  Another reason for choosing the FT benchmark is that it is a standard benchmark that is widely recognized and accepted within the scientific and engineering communities. This means that results obtained from measuring the FT benchmark can be compared with results obtained from other systems, allowing for meaningful performance comparisons between different computer systems.

  Finally, the FT benchmark is a challenging benchmark that can stress the floating-point hardware of a computer system. This means that by measuring the FT benchmark, we can identify potential performance bottlenecks in the system and gain insights into how to optimize the system's floating-point performance.

- IS benchmark: The purpose of the Integer Sort benchmark is to measure the performance of a parallel computer system in sorting a large number of random integers.

  The IS benchmark involves generating a large set of random integers, distributing them among the processors in the parallel system, and then sorting them in parallel using a variation of the bucket sort algorithm. The benchmark measures the time required to sort the integers and the resulting throughput, which is the number of integers sorted per second.

  The performance of the IS benchmark is affected by a variety of factors, including the number of processors in the parallel system, the memory bandwidth, and the interconnect speed between the processors. As a result, the benchmark is a useful tool for evaluating the performance of different parallel systems and for identifying performance bottlenecks.

  I have chosen to measure this benchmark because it provides a measure of performance that is relevant to many real-world applications. Sorting is a common operation in many computational tasks, and the ability to efficiently sort large sets of data is an important factor in the performance of many parallel algorithms. By

measuring the performance of a parallel system on the IS benchmark, researchers can gain insights into the system's performance on a variety of real-world problems.

The classes that have been tested in these benchmarks have been the following:

- Class A: The class is designed to measure the performance of parallel computing systems on a relatively small problem size. It represents a workload that is suitable for testing the performance of small-scale parallel systems or individual nodes within a larger parallel system.
- Class B: This class of benchmarks is designed to test the performance of applications that have irregular communication patterns, such as those found in unstructured grid simulations.
- Class C: This class of benchmarks is designed to test the performance of applications that have both irregular communication patterns and irregular data access patterns.
- Class S: This class of benchmarks is designed to test the performance of small-scale parallel systems, with problem sizes that can fit into the memory of a single node. There are three benchmarks in this class, which test computation, communication, and memory access.
- Class W: This class of benchmarks is designed to test the performance of large-scale parallel systems, with problem sizes that require distributed memory. There are three benchmarks in this class, which test computation, communication, and memory access.

2 Description of the machine to compare

Moore is an homogeneous cluster with a total of 32 processing units and 32GB of main memory, distributed over 8 nodes with an Intel Core i5 processor with 4 cores at 3.1 GHz and 4GB of main memory.

Number of nodes:	8
Number of processing units (per node):	4
RAM Memory in GB (per node):	4
Processor (4 cores per node):	i5 3.1GHz

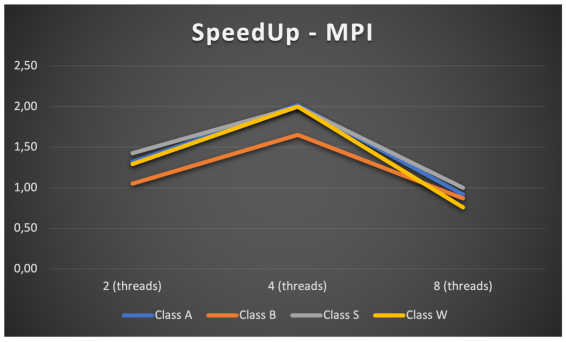
Figure 1: Cluster in which the benchmarks have been executed.

3 Serial, OpenMP and MPI Results

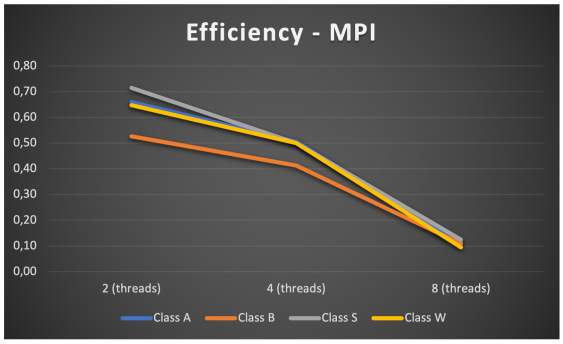
- FT Benchmark:

FT benchmark							
Execution times (s)							
CLASS	Serial	OMP (threads)			MPI (tasks)		
		2	4	8	2	4	8
A	3,76	2,18	1,26	1,33	2,85	1,86	4,09
B	38,31	27,63	16,12	16,68	36,42	23,18	43,96
S	0,10	0,06	0,03	0,04	0,07	0,05	0,10
W	0,22	0,12	0,08	0,09	0,17	0,11	0,29

Figure 2: Result of the execution time of the different classes in the FT benchmark.

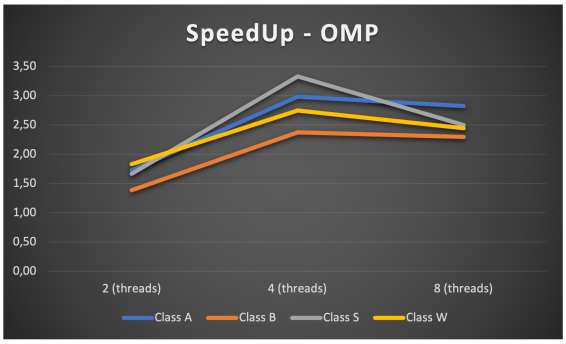


(a) FT - Speed up MPI

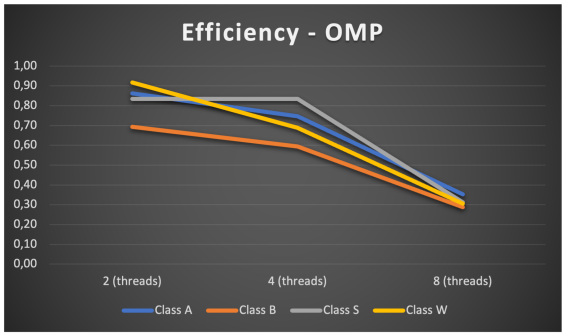


(b) FT - Efficiency MPI

Figure 3: Speed up and Efficiency using MPI in the FT Benchmark



(a) FT - Speed up OMP



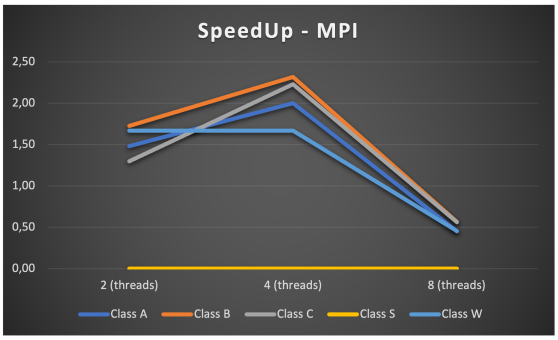
(b) FT - Efficiency OMP

Figure 4: Speed up and Efficiency using OMP in the FT Benchmark

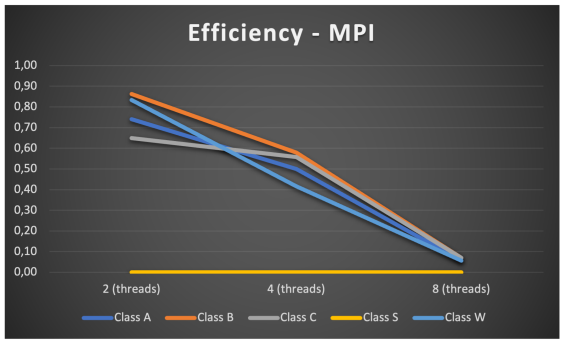
- IS Benchmark:

IS benchmark							
Execution times (s)							
CLASS	Serial	OMP (threads)			MPI (tasks)		
		2	4	8	2	4	8
A	0,40	0,26	0,15	0,16	0,27	0,20	0,89
B	1,83	1,15	0,63	0,64	1,06	0,79	3,24
C	7,23	6,03	3,12	3,13	5,57	3,24	12,90
S	0,00	0,00	0,00	0,00	0,00	0,00	0,01
W	0,05	0,04	0,02	0,03	0,03	0,03	0,11

Figure 5: Result of the execution time of the different classes in the IS benchmark.

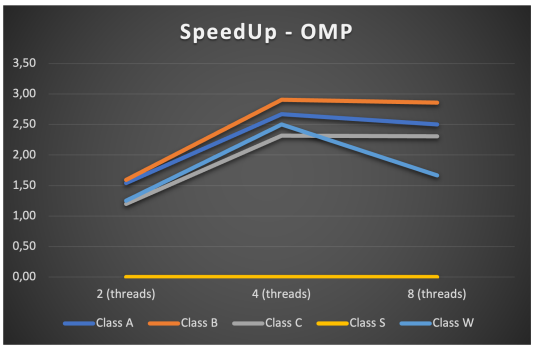


(a) IS - Speed up MPI

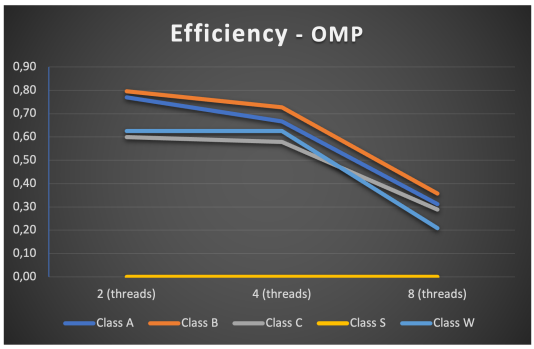


(b) IS - Efficiency MPI

Figure 6: Speed up and Efficiency using MPI in the IS Benchmark



(a) IS - Speed up OMP



(b) IS - Efficiency OMP

Figure 7: Speed up and Efficiency using OMP in the IS Benchmark

4 Analysis of the benchmarking results

Regarding the results obtained from the IS benchmark (OMP), for the speedup values, it appears that in general, increasing the number of threads leads to an improvement in performance, with the exception of Class S, which appears to have no speedup at all because it is designed to test the performance of small-scale parallel systems, with problem sizes that can fit into the memory of a single node making the load on the nodes negligible. It also appears that the performance improvement varies by class, with Class B showing the highest speedup values and Class C showing lower speedup values.

For the efficiency values, it appears that the parallel program is not as effective at utilizing the available resources, as the values are generally below 1, indicating that there is room for improvement in terms of efficiency. The efficiency values also show that as the number of threads increases, the efficiency decreases, which is likely due to issues related to scalability and overhead.

Regarding the results obtained from the IS benchmark (MPI), overall, increasing the number of threads does not always lead to an improvement in performance. For Class A, B, and C, the speedup values increase with the number of threads up to a certain point (4 threads for Class A and B, 8 threads for Class C), after which they start to decrease. For Class S, the same problem occurs when using OMP. For the efficiency values are generally low, indicating that the parallel implementation is not making efficient use of the available

resources. The efficiency values decrease as the number of threads increases, which is expected due to issues related to scalability and overhead. The fact that the speedup values decrease after a certain number of threads could indicate that the implementation is not able to efficiently scale beyond a certain point.

Regarding the results obtained from the FT benchmark (OMP), overall increasing the number of threads leads to an improvement in performance for all classes. The speedup values vary by class, with the highest speedup values observed for Class S and Class W, and the lowest speedup values observed for Class B. The speedup values are relatively consistent for all classes, with little decrease in speedup as the number of threads increases. Regarding the efficiency values are generally higher than those observed for the MPI implementation of the IS benchmark, indicating that the parallel implementation is making better use of the available resources.

Regarding the results obtained from the FT benchmark (MPI), the speedup values vary by class, with the highest speedup values observed for Class S and the lowest speedup values observed for Class A and Class W. The speedup values are not as consistent as the OMP implementation of the FT benchmark, with a significant drop in speedup observed for some classes as the number of threads increases. Regarding the efficiency values are lower than those observed for the OMP implementation of the FT benchmark, indicating that the parallel implementation is not making as efficient use of the available resources. Class A and Class W have lower efficiency values compared to Class B and Class S, indicating that the implementation is not making as efficient use of resources for these classes.

5 Conclusions

The OMP implementations showed better performance than the MPI implementations. Class S, in the FT benchmark, showed the highest speedup values, indicating good performance for certain types of calculations. In the MPI implementations of the benchmarks, after analyzing the data, it can be said that: increasing the number of threads generally led to an improvement in performance, but with less consistency than in the OMP implementations. Overall, the results indicate that parallel implementations can lead to significant improvements in performance for certain types of calculations, but the choice of implementation (OMP vs. MPI) can have a significant impact on performance. Finally, the OMP implementation of the FT benchmark appears to be more scalable and efficient compared to the MPI implementation of the IS benchmark. The high efficiency values and consistent speedup values indicate that the parallel implementation is making good use of the available resources and is able to efficiently scale across different numbers of threads.

6 References

[1] NAS Parallel Benchmarks.
`https://www.nas.nasa.gov/software/npb.html`

[2] Wikipedia - NAS Parallel Benchmarks.
`https://en.wikipedia.org/wiki/NAS_Parallel_Benchmarks`