Benchmarking Performance: Influence of Task Location on Cluster Throughput

Manuel Rodríguez-Pascual, José Antonio Moríñigo(✉), and Rafael Mayo-García

Centro de Investigaciones Energéticas, Medioambientales y Tecnológicas CIEMAT, Madrid, Spain
{manuel.rodriguez,josea.morinigo}@ciemat.es
http://rdgroups.ciemat.es/web/sci-track

Abstract. A variety of properties characterizes the execution of scientific applications on HPC environments (CPU, I/O or memory-bound, execution time, degree of parallelism, dedicated computational resources, strong- and weak-scaling behaviour, to cite some). This situation causes scheduling decisions to have a great influence on the performance of the applications, making difficult to achieve an optimal exploitation with cost-effective strategies of the HPC resources. In this work the NAS Parallel Benchmarks have been executed in a systematic way in a modern state-of-the-art and an older cluster, to identify dependencies between MPI tasks mapping and the speedup or resource occupation. A full characterization with micro-benchmarks has been performed. Then, an examination on how different task grouping strategies and cluster setups affect the execution time of jobs and infrastructure throughput. As a result, criteria for cluster setup arise linked to maximize performance of individual jobs, total cluster throughput or achieving better scheduling. It is expected that this work will be of interest on the design of scheduling policies and useful to HPC administrators.

Keywords: MPI application performance · Benchmarking
Cluster throughput · NAS Parallel Benchmarks

1 Introduction

The evolution in processors during the last decade has been oriented towards an increasing degree of parallelism and this fact has deeply impacted all levels of computing hardware and software. The design of clusters and supercomputers is also following this path. For example, the number of cores according to the TOP500 list [1] has grown exponentially since 1993. Current trends include the use of many-core processors, driving the number of computing units even further. An obvious way of getting the most of this is the usage of highly parallel applications. MPI and OpenMP continue being instrumental to create highly scalable applications suitable for this environment. Applications can be classified into CPU, I/O or memory-bound depending on which factor limits its execution speed. Other common issues regarding requirements are execution time, degree of parallelism and required computational resources, to cite some. This leads to many questions and specific scheduling decisions. Probably the first is what should we do with partially-filled multi-core processors? When a given job is only using some of the CPUs of a node, or just some of the cores of a CPU, what is the most efficient decision? On one side, executing some other job at the same time would lead to a most intensive usage of the resources; on the other, sharing the resources (namely memory at different levels and I/O) may force both jobs to compete for them and slow down, having in fact a negative impact on the total execution time. The problem is complex and has extra dimensions, and the scope of this work aims to shed some light on performance issues.

2 Related Work

Impact of MPI task locality has been investigated in [2] with three kernels of NPB and application codes running in a cluster of 32 CPUs. They show that an execution time saving of up to 25% is possible. Their number of used CPUs is small and the authors plan to extend the experiments to large-scale machines since it seems necessary to be conclusive about what happens in situations of many processors. In [3] it is summarized the results of mapping MPI tasks onto sockets, taking into account the machine topology. Results show that it is beneficial to map tasks onto as many sockets per node as possible (the bigger savings in execution time, up to 30%, are obtained precisely for those cases). Similar experiments are done in [4], reporting an improvement of about 15%. In particular, research dealing with multicore architectures has been focused in the last years. To this regard, [5] presents the gain in computational efficiency of a MPI-based production application that exhibits a performance peak improvement of about 9% (with averaged performance improvement of 6%), attributed to a better use of cache-sharing at the same node and to the high intra- to internode communication ratio of the cluster. Although it is a modest speedup, it is noticed that it is obtained with minor source code modifications. The work in [6] points to the same direction by evaluating the impact of multicore architectures in a set of benchmarks; but on the contrary, they conduct a non-straightforward adaptation of the original application. Their characterization of the inter- to intranode communications ratio throws a figure of 4 to 5 in the worst case. This kind of node mappings is an area where little to moderate efforts are required for significant gains in application performance. The impact of internode and intranode latency is analyzed in [7] using a parallel scientific application with MPI tasks mapped onto the CPUs of an infiniband-based cluster of 14 nodes. With the objective of improving the computational efficiency, [8] analyzes how many cores per node should be used for applications execution. Here, both NPB and a large-scale scientific application code are executed in three single-socket-per-node clusters. They identify that task mapping is an important factor on performance degradation, being the memory bandwidth per core the primary source of performance drop when increasing the number of cores per node that participate in the computation. Something similar concludes [9], showing a high sensitivity of the attained NAS kernels performance to the multi-core machines. In [10] it is detected that NPB exhibit high sensitivity to the cluster architecture. Also, MPI tasks mapping reveals that distributing them over the nodes is better from a computational standpoint in most cases. According to their experiments, an up to 120% speedup is attained for most of the NAS kernels. They explain this behaviour because by distributing the tasks they do not have to compete for node local resources, a scenario that seems to occur when running tasks are sharing a slot or are located in slot of the same node. In [11] a semi-empirical predictive model is formulated and tested with a large-scale scientific application code, which provides good results for weak scalability cases and show that it can lead to a 5% increase of the execution time. The study conducted in [12] on mapping MPI tasks to cores using micro-benchmarks and NPB, shows that it may affect significantly the performance of intranode communication, which is closely related to the inter- to intranode communication ratio.

These previous investigations point out to the large sensitivity of the execution time to the task mapping. The impact of grouping or not MPI tasks "outside of the box" (out of the same node), over sockets or cores within the node is high as it is seen that execution time varies significantly. Also the lack of understanding of how to proceed in a systematic manner with an application in a specific cluster remains and further clarifications are needed to improve the cluster efficiency and usage. The present investigation summarizes the results of mapping MPI tasks onto cores in two different infiniband-based clusters at CIEMAT.

Then, processor mapping combinations have been tested to explore the impact on cluster throughput and hence, to build usage criteria aiming at feeding better scheduling strategies to support the scientific groups.

Hence, the present work explores the behaviour of different scientific kernels on modern infrastructures and the impact on clusters throughput. An analysis on how task location, network traffic and resource sharing affect their execution time has been carried out to infer a generalization of their behavior. This information can then be useful to improve scheduling algorithms and cluster setups. In the text, a *job* is composed by one or more *tasks*. The assignment of those tasks to computational resources (*mapping*) determining when and where to run each job constitutes the *scheduling* process.

3 Characterization of HPC Facilities

3.1 Benchmarking

The chosen applications for systematic benchmarking can be divided in two groups. The first one is system benchmarks, to measure the raw performance of the components of our clusters. The second one is application benchmarks (NPB), to test the behavior of the infrastructure when running real applications.

STREAM benchmark [13] measures sustainable memory bandwidth and tests the communication bandwidth between the socket and its RAM. In multicore sockets an OpenMP flag is set for building one thread/core during compilation.

OSU micro-benchmark [14] measures the latency and bandwidth of MPI libraries and interconnects. It implements a set of routines with various communication patterns. It measures intra- and internode communication bandwidths.

Bonnie++ [15] is a small yet powerful benchmark to measure disk performance. It provides a number of tests of hard drive and file system performance.

Intel Memory Latency Checker 3.1 [16] allows accessing memory chunks located in the elements of the memory hierarchy, measuring the latency time.

All together, these benchmarks allow characterizing the cluster raw performance. Hence, a better understanding of the results gathered with a set of scientific kernels is intended. The NAS Parallel Benchmarks (in short NPB) [17] is developed at the Numerical Aerodynamic Simulation (NAS) program at NASA. It has evolved as a group of kernels, set for a variety of problem sizes (classes) of increasing computing cost. All together are representative of algorithmic building blocks found in the aforementioned scientific. Among them are examples of memory-bound and CPU-bound, or weak-scaling and strong-scaling kernels. The present investigation uses NPB v2.0, which includes seven portable kernels (Fortran90, MPI parallelised) whose acronyms corresponds to: BT- Block Tridiagonal solver; CG- Conjugate Gradient; EP- Embarrassingly Parallel; FT- Discrete fast Fourier Transform; IS- Integer Sort; LU- Lower-Upper Gauss-Seidel solver; MG- MultiGrid on a sequence of grids.

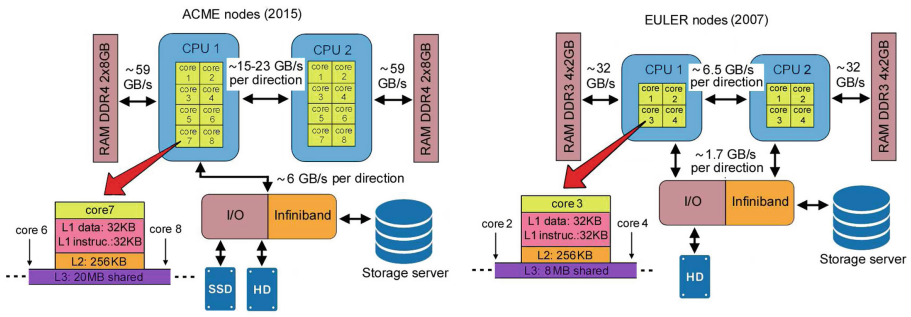

Fig. 1. Major architecture and communications for clusters ACME and EULER (bandwidths are approximate best cases sustained values).

3.2 Infrastructure Characterization

Two computing facilities of different generations based at CIEMAT have been employed. The first one, EULER, is a production HPC shared among the scientific groups. It consists in 480 Xeon®5450 quadcore@3.0 GHz, 2 GB RAM/core, mounted on Dell PowerEdge M610 blades. When EULER was installed in Autumn 2008, its performance according to LINPACK (23 TFlops peak) would have ranked it around position 300 of TOP500. Because it is under full usage, the experiments have been done while sharing the resource and the restriction of accessing to a reserved set of 8 nodes within a limited timeframe (this matches with how the analyzed applications behave in real environments).

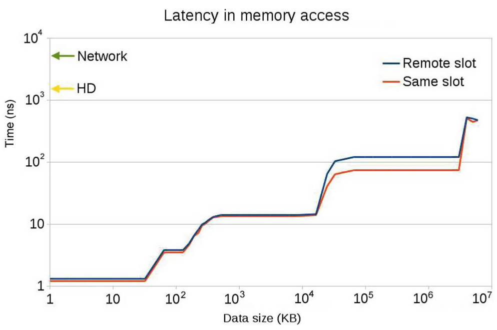

Fig. 2. Latency of a core accessing increasingly larger data blocks in ACME, corresponding to cache, RAM and disk. 'Same slot' and 'Remote slot' refer to the location of the processor to which the accessed memory is connected in the NUMA system.

The second one, ACME, is a smaller, state-of-the-art HPC for research purposes and fully devoted to this project, which counts with 10 nodes (8 of them are computing nodes): 2 Bull R424-E4 chassis with 4 nodes each, plus another two devoted to host accelerators and fast network storage. Each node consists in a Supermicro X10DRT-P motherboard with two 8-core Xeon®E5-2640 v3@2.60. Each node counts with 32 GB DDR4 RAM memory @2133 MHz, in the form of 4×8 GB modules. Two of these modules are connected to each processor (NUMA). Each core accesses to half the memory with rather smaller access time than to the other half, as show in Fig. 2. Network is provided by Infiniband FDR. The MPI library mvapich2-2.2b is installed in both. Figure 1 displays the hardware of ACME and EULER nodes. It includes some performance metrics obtained with the benchmarks and from the official documentation. It is worth noticing that most the results match their counterparts provided by the hardware vendors. ACME has higher intranode communication bandwidth (3 to 4 times higher than EULER's); 2.5 times larger shared L3 cache; and 3 times higher infiniband bandwidth. Both clusters have a ratio of intra- to internode bandwidth within 3 to 3.5.

3.3 Influence of Node Sharing on Memory Access Time

In multi-core CPUs and in multi-CPUs nodes, there is a decision concerning whether it is better or not to share the resources among pending jobs, or should a sole application be executed at the same time. It can be inferred that sharing a node between two or more jobs may increase RAM misses, as the available memory is shared between the running jobs, so they have less space for data and executable. This same issue happens when sharing a multi-core CPU, leading to an increased number of misses in L3 cache. Hence, it is necessary to measure the access time to the different memory layers in order to quantify its influence.

Figure 2 shows latency in ACME when a given core is accessing to memory. As expected, L1 cache is the fastest one but only stores up to 32 KB of data; then comes L2 with 128 KB, L3 with 20 MB, and after that the RAM memory. In this case, as pointed out before, there is a significant difference (60%) between accessing the modules connected to the same processor and those connected to the other one in the other slot of the same motherboard. The last step corresponds to the sizes between 32 and 64 GB, where both RAM modules are accessed to store/read. The last step of the memory hierarchy is represented by the persistent storage. It counts with a HD (1 TB) for scratch and temporary files and network storage for the users' home directory and the non-OS applications (scientific codes). Measured latency is 1.5 µs for the HD and 7.35 µs for the network storage, roughly an order of magnitude larger than RAM latencies.

Summing up, results show that a miss in any cache level implies accessing an upper layer of the memory hierarchy with a penalization of about an order of magnitude in latency. Resource sharing will then have an impact on the job execution due to the influence on the access time of the different cache levels.

4 Results

4.1 Cluster Performance

The experiments with NPB have been repeated under four Slurm setups:

- Dedicated Cores: one-to-one assignment of cores to MPI tasks of the parallel job (a core executes only one MPI task of that job; set in ACME and EULER).
- Dedicated Sockets: a socket may only execute MPI tasks of the same job. Once the socket is occupied by at least one MPI task of a job, no other part of another job may be executed on it in the meanwhile (set in ACME).
- Dedicated Nodes: an entire node is assigned in exclusivity to execute MPI tasks of the same job (set in both ACME and EULER).
- Dedicated Network (reference case): the whole cluster executes only one parallel job at the same time, thus avoiding any overhead due to network congestion (set in ACME). This is a scenario devoted to obtain reference execution times.

Table 1 compares the total execution time of a set of NAS kernels. To mimic real-life workloads, all jobs corresponding to different kernel classes (sizes of computed problems) and degrees of parallelism (about 4,000 jobs sent for each cluster setup) were submitted at the same time, letting Slurm scheduler to decide where and when to execute them using 8 nodes (16 cores/node) in ACME and 16 nodes (8 cores/node) in EULER. There was no indication of any job maximum execution time, thus no preemption techniques were employed. Table 2 shows

Table 1. Total execution time of NAS kernels in ACME with different cluster setups (time ratio is referred to the Dedicated Network setup)

Setup	Execution time (s)	Time ratio (%)
Dedicated nodes	36533	32.6
Dedicated cores	19442	17.5
Dedicated sockets	28989	25.8

Table 2. Submitted jobs of all NAS kernels partitioned per degree of parallelism.

Degree of parallelism	1	2	4	8	16	32	64	128
Number of jobs	210	360	630	720	840	540	400	170
Percentage of jobs (%)	5.4	9.3	16.3	18.6	21.7	14	10.3	4.4

the jobs according to their degree of parallelism; note that 128 is the number of cores in the cluster ACME. This way, the impact of MPI tasks location inside the clusters has been analysed under the Slurm setups. It is noted that the Dedicated Sockets setup is the 2nd more efficient after the Dedicated Cores setup (about 30% of the jobs counts for 8 or even less MPI tasks), which is able to allocate more than one job at the same time.

4.2 NAS Benchmarking

The number of nodes (nN) times the number of MPI tasks per node (nT), in short $nNxnT$ (see Fig. 3), that define the configuration of each experiment (say, a definite kernel of a given class, executed under a cluster setup) has been repeated 10 times, with their average referred in what follows as a computed case. The experiments conducted under Dedicated Network setup include the kernels of class D to enlarge the population of computed cases. For the other

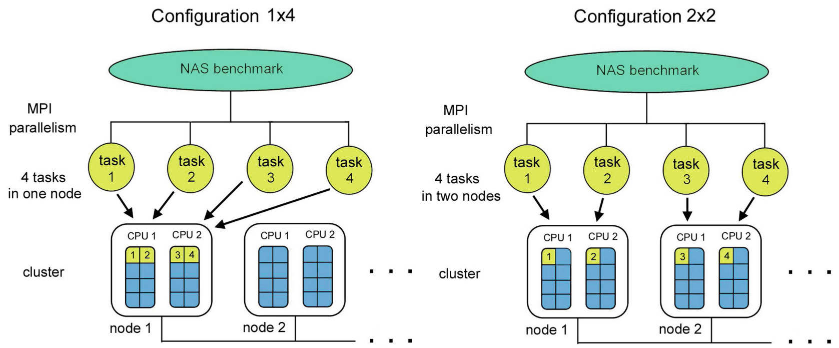

Fig. 3. Example of 1×4 and 2×2 MPI tasks mapping in cluster ACME.

cluster setups, only experiments with the A, B and C classes of the kernels have been conducted to guarantee that the running MPI processes fit into the RAM memory as well as to bound the workload computing time. All exhibit standard deviation <1%.

4.3 Dedicated Nodes Cluster Setup

Bearing in mind that the Dedicated Cores setup is a realistic scenario of production clusters, a general trend can be stated out of Fig. 4, which depict the execution time of the seven NAS kernels (sectors of the circles) for ACME. The nondimensional execution time (referred to the execution time obtained in the cluster configuration of the fewest number of nodes, which corresponds to the circle's centers) is shown in the figure. Figure 4 shows that most computed cases takes more execution time as far as more nodes are involved.

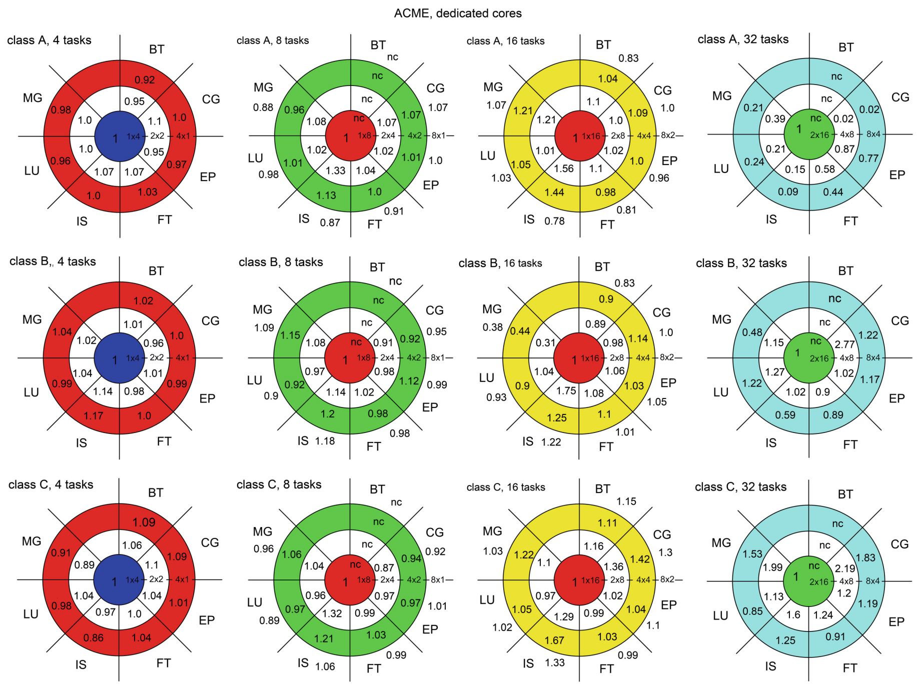

Fig. 4. Relative execution time for the Dedicated Cores setup in ACME. Nondimensional computing time is referred to the case of all processes running in one node.

Hence, it can be said that grouping MPI tasks within as few nodes as possible is good to achieve shorter execution times. This seems a general rule inferred out of the plot after examining the behaviour of the kernels as a whole. But two points must be made. On one hand, a case by case examination reveals that there are exceptions to this rule, as it is the case of the LU kernel (class B - 8 tasks); IS kernel (class C - 4 tasks); and others. So awareness of non consistent tendencies of the kernels has to be considered. And on the other hand, non monotone behaviour occurs in several computed cases. Some of these may be explained by the kernel requirements (CPU- or memory intensive,...), but also it is suggested that the execution is affected by the MPI tasks of rather different behaving kernels located in neighboring cores by the scheduler, which compete for resources (RAM and traffic). However, it is interesting to notice such a pattern (and criterium) related to grouping MPI tasks in fewer nodes, which seems to be more effective in saving execution time as the degree of parallelism and size (class) increase. On the contrary, computed cases of low number of tasks (see column of 4 MPI tasks in Fig. 4) show a small variation of the execution time (within 5–10%) with the number of nodes. This general trend observed in ACME, it is not so definite in EULER, which shows greater sensitiveness of the execution time (the equivalent figure is not included because of space constrains). The number of computed cases in EULER with some speedup when the MPI tasks are distributed among nodes dominates. Hence, EULER shows a somehow opposite behaviour compared to ACME (due to extension restrictions, its execution time plots are not included). As a result, how to proceed in EULER to speedup kernels execution is unclear and a more in depth kernel-by-kernel analysis seems necessary. A comparison of the different behaviour found in ACME and EULER in terms of the execution time for the two MG and EP kernels (memory- and CPU-intensive, respectively) is depicted in Fig. 5 for Dedicated Nodes setup. The plots for the MG kernel with classes B and C show that the speedup increases with monotone trend as more nodes are involved in the $nNxnT$ configuration (nT = 4, 8, 16 and 32). And it is seen that this speedup is significantly greater in EULER, which can be explained because EULER nodes are more memory-bound than ACME's. The EP kernel in ACME exhibits also a rather small speedup when tasks are distributed over nodes. But on the contrary, EULER shows that the EP kernel runs slower when it is taken "out of the box" (that is, when the tasks are partially migrated from all being grouped in one node). It is visible in Fig. 5 that a big jump in execution time occurs as the EP kernel goes from $1xnT$ to $2xnT$ (with $nT = 4$ and 8). It is noticed that the computed cases in EULER corresponding to $nT = 16$ and $nT = 32$ start at configurations 2×8 and 4×8, respectively, thus it is not possible to have evidence of the "out of the box" effect in these cases (but it is plausible that the wavy pattern observed in these be similar to the wavy one observed for $nT = 8$ from the 2×4 configuration on). In resume, the rule derived for the EP kernel in EULER is that MPI task grouping makes sense as the execution time drops. And besides, the reversed behaviour seen in ACME for the EP kernel can be justified because of the much higher internode bandwidth, which compensate the "out of the box" effect observed in EULER.

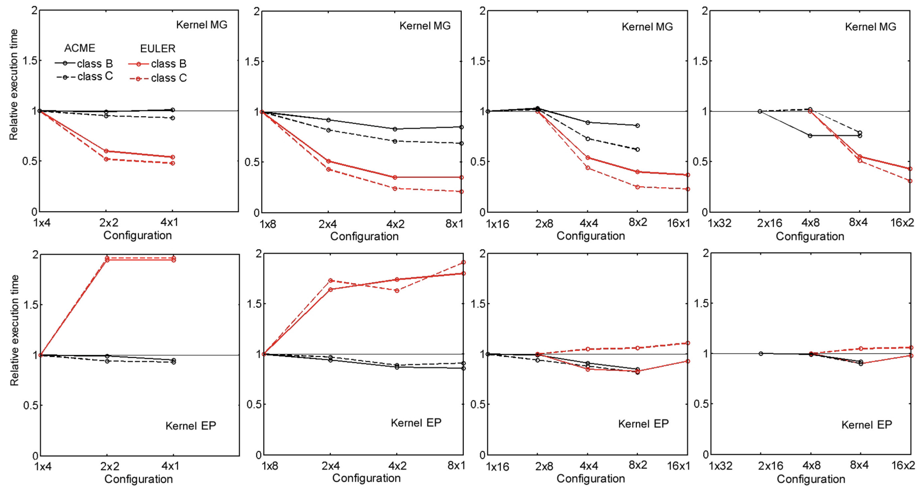

Fig. 5. Relative execution time for the Dedicated Nodes setup. NAS kernels MG (upper row) and EP (lower row) are shown for clusters ACME and EULER.

4.4 Sensitivity to the Clusters Setup

The performance of kernels MG and EP is plotted in Fig. 6 corresponding to 8 MPI tasks and the four clusters setups analysed for the classes A, B and C. This plot is relevant because it provides four $nNxnT$ configurations that start at 1×8 in both clusters, so the "out of the box" effect can be focused, if any. For both kernels, Dedicated Network and Dedicated Nodes setups provide very

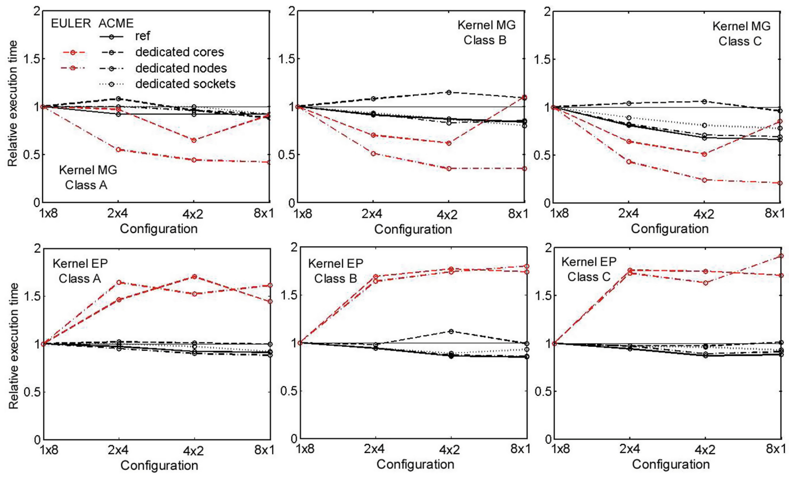

Fig. 6. Impact of cluster setup on the relative execution time for MG (memory-intensive) and EP (CPU-intensive) kernels with 8 MPI tasks in clusters.

similar execution time, showing a monotone, quasi-linear drop with the number of nodes, which suggests the benefit (but small) of adding nodes to the computation. The plot for Dedicated Sockets is similar, but even it is seen a smaller drop of the execution time. This pattern is observed for the MG and EP kernels with all classes (it is noted that Dedicated Network and Dedicated Sockets setups are included only for ACME since EULER is a production cluster and only a portion of it was assigned to this research). Again, for the MG kernel in EULER under Dedicated Nodes, it is observed a higher speedup with the number of nodes, compared to ACME. In particular, for the execution of the EP kernel in EULER, it is visible the "out-of-the-box" effect under both setups: a large increase of the execution time when the EP kernel goes from 1×8 to a 2×4 configuration, followed by a saturation of the drop when additional nodes are included. A different conclusion is derived for EULER: while for the memory-bound MG kernel is beneficial to distribute the MPI tasks over so many nodes as possible (the smaller host memory of its sockets may explain the significant improvement), the CPU-bounded EP kernel demands to group them into one node to attain the best performance. In resume, the MG and EP kernels under Dedicated cores setup in ACME, points out to the dominance of a performance drop (but not monotone) when additional nodes are added to the computation (the execution time shows a pattern of either a moderate increase of up to 20%, or a slight drop in some cases). Comparison of the speedup obtained with the Dedicated Nodes and Dedicated Cores setups for the whole set of experiments conducted in ACME and EULER is given in Figs. 7 and 8, respectively. The plotted boundary lines indicate the unused portion of the cluster due to the Slurm setup itself, which serves to build criterium about how much speedup is possible

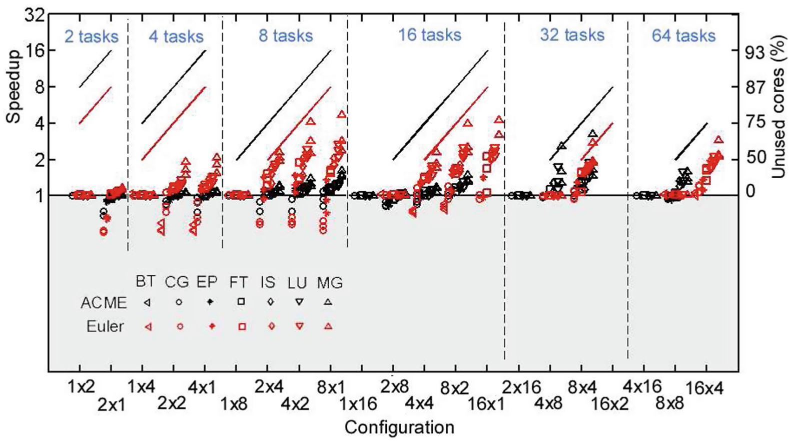

Fig. 7. Map of speedup vs. usage of computational resources for NAS under the Dedicated Nodes setup (locus of % of unused cores is plotted for each number of MPI tasks. Line color corresponds to the symbols), depicted for clusters ACME and EULER. (Color figure online)

and which is the extra cost due to not using a portion of the machine (e.g.: say an ACME 2×4 configuration with Dedicated Nodes setup. This implies 4 tasks running on a socket of 8 cores. Since each node has 2 sockets, the occupation reads 4 of a total of $8 + 8 = 16$ cores, which means a 75% of unused cores).

The vertical scales in the plots relate speedup and % of unused cores (i.e. speedup of 2 corresponds to a 50% of unused cores; speedup of 4 to a 75% of unused cores; and so on). This criterium remarks the importance of searching for a balance between significant speedups and not having too many unused cores. Obviously, it is a matter of settling a sweet point for users and cluster administrators. But under the Dedicated Nodes setup, it is seen that few points are over the boundary lines. Only in the Dedicated Cores setup, all boundary lines collapse into the 0%-unused cores situation (full occupation).

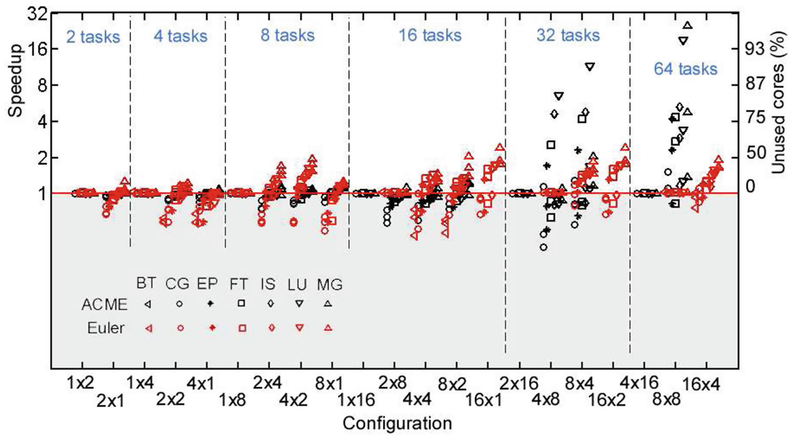

Fig. 8. Map of speedup vs. usage of computational resources for NAS under the Dedicated Cores setup depicted for clusters ACME and EULER.

5 Conclusions

The NAS Parallel Benchmarks have been executed in a systematic way on two clusters with rather different internode and intranode bandwidth properties, to identify dependencies between MPI tasks mapping and execution time speedup or resource occupation. The study comprises jobs up to 128 MPI tasks, bounded accordingly to our clusters size and usage constraints, as well as justified by the limited strong-scaling properties of NAS kernels.

The findings can be related to two scenarios. When the clusters are configured to run parallel jobs in exclusivity, the results show that in most cases the execution time drops as the MPI tasks are distributed over the nodes (this agrees with previous investigations) and it seems efficient to distribute a given parallel job over cores located in different nodes. However, the other important scenario found in production HPC clusters corresponds to the need of sharing resources among set of jobs, such that the socket cores execute MPI tasks of different jobs. In this situation, a rather different behaviour is observed, much more sensitive to the type of cluster. In our state-of-the-art cluster ACME, many of the carried out experiments show a speedup when MPI tasks run in the fewest number of nodes. This is opposite to what is found in our older cluster EULER, where execution time trends are more sensitive to the algorithm properties and part of the experiments points out to task distribution over nodes to shorten the execution time. This major result found in ACME feeds the discussion about possible computational efficiency benefits by tailoring live tasks migration and scheduling policies in modern clusters. In production clusters which share a significant load of serial jobs while running parallel jobs (a 7-year analysis of the executed tasks in our cluster EULER showed that more than half were serial), the question of to which extent serial tasks may act as perturbations to the execution of parallel jobs arises and deserves consideration to clarify the best live task migration policies within the context of optimizing clusters occupation. Other interesting aspect is how different the results would be in the case of hybrid MPI/OpenMP tasks. These open issues are part of the ongoing research within our group.

Acknowledgment. This work was supported by the COST Action NESUS (IC1305) and partially funded by the Spanish Ministry of Economy and Competitiveness project CODEC2 (TIN2015-63562-R) and EU H2020 project HPC4E (grant agreement n 689772).

References

1. Top 500. www.top500.org
2. Jeannot, E., Mercier, G., Tessier, F.: Process placement in multicore clusters: algorithmic issues and practical techniques. IEEE Trans. Parallel Distrib. Syst. **25**(4), 993–1002 (2014)
3. Chavarría-Miranda, D., Nieplocha, J., Tipparaju, V.: Topology-aware tile mapping for clusters of SMPs. In: Proceedings of the 3rd Conference on Computing Frontiers (CF 2006), pp. 383–392. ACM (2006)
4. Smith, B.E., Bode, B.: Performance effects of node mappings on the IBM Blue-Gene/L machine. In: Cunha, J.C., Medeiros, P.D. (eds.) Euro-Par 2005. LNCS, vol. 3648, pp. 1005–1013. Springer, Heidelberg (2005). https://doi.org/10.1007/11549468_110
5. Rodrigues, E.R., Madruga, F.L., Navaux, P.O.A., Panetta, J.: Multi-core aware process mapping and its impact on communication overhead of parallel applications. In: Proceedings of the IEEE Symposium on Computers and Communications, pp. 811–817 (2009)
6. Chai, L., Gao, Q., Panda, D.K.: Understanding the impact of multi-core architecture in cluster computing: a case study with Intel dual-core system. In: Proceedings of the 7th IEEE International Symposium on Cluster Computing and the Grid, CCGrid, pp. 471–478 (2007)

7. Shainer, G., Lui, P., Liu, T., Wilde, T., Layton, J.: The impact of inter-node latency versus intra-node latency on HPC applications. In: Proceedings of the IASTED International Conference on Parallel and Distributed Computing and Systems, pp. 455–460 (2011)
8. Xingfu, W., Taylor, V.: Using processor partitioning to evaluate the performance of MPI, OpenMP and hybrid parallel applications on dual- and quad-core Cray XT4 systems. In: Cray UG Proceedings (CUG 2009), Atlanta, USA, pp. 4–7 (2009)
9. Ribeiro, C.P., et al.: Evaluating CPU and memory affinity for numerical scientific multithreaded benchmarks on multi-cores. IJCSIS **7**(1), 79–93 (2012)
10. Wu, X., Taylor, V.: Processor partitioning: an experimental performance analysis of parallel applications on SMP clusters systems. In: 19th International Conference on Parallel Distributed Computing and Systems (PDCS 2007), CA, USA, pp. 13–18 (2007)
11. Wu, X., Taylor, V.: Performance modeling of hybrid MPI/OpenMP scientific applications on large-scale multicore. J. Comput. Syst. Sci. **79**(8), 1256–1268 (2013)
12. Zhang, C., Yuan, X., Srinivasan, A.: Processor affinity and MPI performance on SMP-CMP clusters. In: IEEE International Symposium on Parallel & Distributed Processing, Workshops and Ph.D. Forum (IPDPSW), Atlanta, USA, pp. 1–8 (2010)
13. McCalpin, J.D.: Memory bandwidth and machine balance in current high performance computers. In: IEEE Computer Society Technical Committee on Computer Architecture (TCCA) Newsletter, pp. 19–25 (1995)
14. OSU Micro-Benchmarks. http://mvapich.cse.ohio-state.edu/benchmarks
15. Bonnie++. www.coker.com.au/bonnie++
16. Intel Memory Latency Checker 3.1. www.intel.com/software/mlc
17. Bailey, D., et al.: The NAS parallel benchmarks. Technical report (1994)