Benchmarking LAMMPS: Sensitivity to Task Location under CPU-based Weak-scaling

José A. Moríñigo[1], Pablo García-Muller[1], Antonio J. Rubio-Montero[1], Antonio Gómez-Iglesias[2], Norbert Meyer[3], and Rafael Mayo-García[1]

[1] Centro de Investigaciones Energéticas, Medioambientales y Tecnológicas CIEMAT, Madrid, Spain josea.morinigo@ciemat.es
http://rdgroups.ciemat.es/web/sci-track
[2] Oak Ridge National Laboratory, 1 Bethel Valley Rd, Oak Ridge, TN 37830, USA
[3] Poznań Supercomputing and Networking Center, Jana Pawla II 10, 61-139 Poznań, Poland

Abstract. This investigation summarizes a set of executions completed on the supercomputers Stampede at TACC (USA), Helios at IFERC (Japan), and Eagle at PSNC (Poland), with the molecular dynamics solver LAMMPS, compiled for CPUs. A communication-intensive benchmark based on long-distance interactions tackled by the Fast Fourier Transform operator has been selected to test its sensitivity to rather different patterns of tasks location, hence to identify the best way to accomplish further simulations for this family of problems. Weak-scaling tests show that the attained execution time is closely linked to the cluster topology and this is revealed by the varying time-execution observed in scale up to thousands of MPI tasks involved in the tests. It is noticeable that two clusters exhibit time saving (up to 61% within the parallelization range) when the MPI-task mapping follows a concentration pattern over as few nodes as possible. Besides this result is useful from the users standpoint, it may also help to improve the clusters throughput by, for instance, adding live-migration decisions in the scheduling policies in those cases of communication-intensive behaviour detected in characterization tests. Also, it opens a similar output for a more efficient usage of the cluster from the energy consumption point of view.

Keywords: Cluster throughput · LAMMPS benchmarking · MPI application performance · Weak scaling.

1 Introduction

In the last decade, computer science has evolved to a paradigm in which High Performance Computing (HPC) systems span thousand of nodes. Current architectures are basically following two major trends: clusters based on nodes with processors (CPUs) plus accelerators and multi-level memory; and clusters based on nodes with groups of equal low-power cores with a single-level memory. The first architecture usually has fewer nodes compared to the second one (consequently more parallel tasks may be allocated within a node), but both scenarios

exhibit a huge amount of cores. This fact can be easily checked in the TOP500 list [1] of the most powerful supercomputers, with number of cores growing exponentially since 1993. The CPUs evolution over the last years has driven an increasing degree of parallelism in the codes executed by the final users.

Aiming at an optimized combination of performance, cost and power, the consequence is that hardware and software stacks must be now built bearing also in mind the applications that fully exploit the computing infrastructure, leading to the so-called codesign [2] in which the different elements involved in supercomputing are tightly integrated. An efficient, optimum exploitation of this formidable computing power by highly parallel applications has required to the moment of message passing libraries (MPI, OpenMP,...), which have resulted to be cornerstone in order to implement highly scalable applications suitable for these environments. These ideas are being promoted by various exascale initiatives in the USA (Exascale Computing Project) [3], Europe (EuroHPC [4] and PRACE [5]), China (Tianjin Supercomputer Center) [6], and Japan (Post K) [7].

Nevertheless, relying on a perfectly implemented message passing strategy is not enough. Closely related to it is the computing platform topology, in particular the way the interconnections of the cluster have been designed, and lately may contribute to traffic bottlenecks and network contention. This is a delicate issue for massive parallel applications, as far as inter-node communication asymmetries may develop, thus contributing to a performance collapse (even supercomputers counting on either InfiniBand or OmniPath can exhibit a worse behavior as the number of used nodes and cores increases). Therefore, it is of value to analyze the performance sensitivity to the MPI-tasks mapping for a given application executed on the cluster, in order to assess the effect of task spreading over the nodes.

It is well known that applications speedup is constrained by several factors, mainly CPU-intensive, memory-bandwidth (or communication-intensive) and/or I/O-intensive behaviour. Other issues to be taken into account can be the available execution time and accessible computational resources, as well as the requested degree of parallelism. This context leads to many questions and specific scheduling decisions in HPC systems, being the first to come to mind, how to deal with partially-filled multi-core processors, i.e. what to do when a given job is only using some of the CPUs of a node, or just a subset of the cores of a CPU. As can be easily inferred, an adequate decision should lead to an improved computing and/or energy efficiency. One example could be the case in which resources are shared by two different jobs (namely I/O and memory at different levels), so it may lead to a competition and, consequently, to slow down the whole execution (that is, a negative impact). Unlike, executing some other job at the same time on the same node/CPU could imply a more intensive usage of the cluster. To clarify this complex disjunctive (where more coupled issues and extra dimensions could affect), the present investigation based on the solver LAMMPS following a weak-scaling perspective is intended to shed some light and is motivating this work. The article is structured as follows. The related

work is documented in the next section, whereas the description of the HPC clusters is briefly presented in Section 3. After that, the benchmark used with LAMMPS and its relevant information to understand the algorithmic issues is provided in Section 4. Section 5 summarizes the results and discussion. Finally some conclusions are given.

2 Related Work

Impact of MPI task locality has been investigated in [8] with scientific mini-kernels and application codes. They show that an execution time saving of up to 25% is possible with grouping tasks. Their investigation is limited to a small number of CPUs and the authors plan to extend the experiments to large-scale machines since it seems necessary to be conclusive about what happens with many processors. In [9], the results of mapping MPI tasks onto sockets are summarized 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 [10], reporting an improvement of about 15%. In particular, research dealing with multicore architectures has been focused in the last years and [11] presents the gain in computational efficiency of a MPI-based production application that exhibits a performance peak improvement of about 9%, attributed to a better use of cache-sharing at the same node and to the high intranode to internode communication ratio of the cluster. Although it seems a modest speedup, it is noticed that it is obtained with minor source code modifications.

In [12] several scientific kernels and production applications are analyzed. Albeit the scope of their study is the system noise in scale, some comparisons for mini-kernels are provided under weak-scaling. It is observed that a speedup degradation greater than 120% occurs when the same number of MPI tasks are distributed massively. The work in [13] points to the same direction by evaluating the impact of multi-core architectures in a set of benchmarks. 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 [14] 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, [15] analyzes how many cores per node should be used for applications execution. 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 [16], showing a high sensitivity of the attained scientific kernels performance to the multi-core machines. In [17] it is detected that the tested mini-kernels 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 experiments. 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. The influence of the resource sharing by different jobs is focused in [18] using a set of scientific mini-kernels. The results show significant dependence to the cluster setup (further details on the clusters setups used in those tests are given in [19]). The study conducted in [20] on mapping MPI tasks to cores using a set of benchmarks and scientific mini-kernels, 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 distributing the MPI tasks over the nodes is high as it is seen that execution time varies significantly.

The sensitivity of LAMMPS to the cluster architecture is analyzed in [21] under strong-scaling tests. The author shows the drop of the performance of two benchmarks (one is the rhodopsin protein, focused in the present investigation) with a range of parallelization up to 1024 cores in three clusters. The strong-scaling performance of same benchmark is also analyzed in [22] and comparisons provided in three clusters but for a small number of cores. In this study, the authors explore the sensitivity of the results to re-configuring the topology of one of the clusters, which exhibits execution differences of up to 9%.

Since having enough computing time and/or range of parallelization in supercomputers seems to be often critical for systematic performance assessments of scientific kernels, an alternative, promising via is the use of clusters emulators (which take into account their network topology in detail), as it is described in [23] with a modeled Stampede platform and running the High Performance LINPACK to evaluate the attainable performance.

The present work tries to fill this gap and explores the behaviour of the molecular dynamics solver LAMMPS compiled for CPUs in three modern computing infrastructures: Stampede at TACC (USA), Helios at IFERC (Japan), and Eagle at PSNC (Poland). Focusing on weak-scaling tests and using a communication-intensive benchmark, this study analyzes the sensitivity of the speedup to varying the MPI-tasks location over the nodes by the assignment (that is, *mapping*) of those tasks to computational resources and how this is related to the supercomputer network topology and resource sharing. These results pave the way for further and deeper studies regarding not only cluster throughput, but also energy consumption.

Furthermore, this information can then be useful to build usage criteria to proceed in a systematic manner with the execution of an application in a specific cluster. Also it aims at feeding better scheduling strategies to support scientific groups.

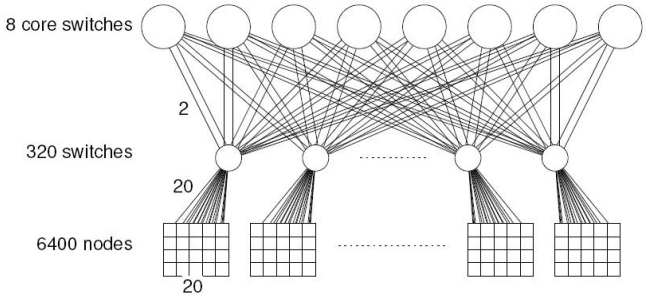

Fig. 1. The fat-tree network topology [23] of Stampede at TACC.

3 Supercomputers Architecture

Next, a brief description of the supercomputers involved in the executions of LAMMPS is provided. To mention that Helios supercomputer is already decommissioned and Stampede has evolved towards the Stampede-2 architecture.

3.1 Stampede at TACC (USA)

Stampede supercomputer [24] was ranked 6th in the TOP500 list (June 2013) by achieving 5,168.1 TFlop s^{-1} and was still ranked 20th in June 2017. In 2017, this machine got upgraded and a portion available during its deployment. At present it is in full operation, renamed Stampede-2. The Stampede platform consisted of 6,400 Sandy Bridge nodes, each with two 8-core Xeon E5-2680 and one Intel Xeon Phi KNC MIC co-processor. The nodes were interconnected through a 56 Gbit s^{-1} FDR InfiniBand 2-level Clos fat-tree topology built on Mellanox switches. Its fat-tree network topology is sketched in Fig. 1. The 6,400 nodes are divided into groups of 20, with each group being connected to one of the 320 36-port switches (4 Tbit s^{-1} capacity), which are themselves connected to 8 648-port core switches (each with a capacity of 73 Tbit s^{-1}). The peak performance of the 2 Xeon CPUs per node was approximately 346 GFlop s^{-1}. The theoretical peak performance of the platform was therefore 8,614 TFlop s^{-1}.

3.2 Helios at IFERC (Japan)

Helios supercomputer [25] was owned by the Computational Simulation Centre (CSC) and ranked 38th in the TOP500 list when it was in fully operation status in November 2014 as it provided a performance peak value of 1,524.1 TFlop s^{-1}. After several upgrades, it finally counted on 4,500 nodes (∼72,000 CPU cores), which were complemented with 180 MIC nodes (∼21,600 co-processors cores). The tests presented in this work were carried out on the major Helios general purpose configuration, i.e. without Xeon Phi included. The processor forming Helios was Sandy-Bridge EP (16-core nodes), which were connected by QDR

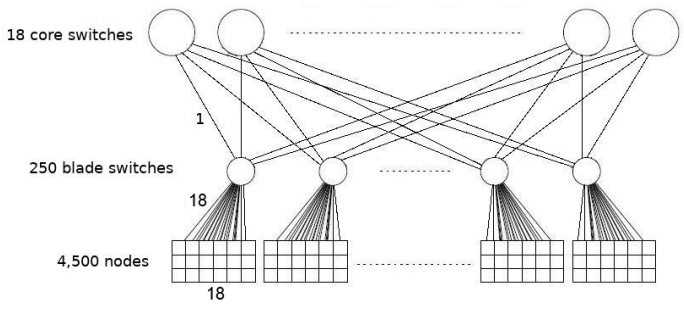

Fig. 2. Architecture of supercomputer Helios at IFERC.

InfiniBand. This connection grouped the computing nodes in sets of 10 which were connected to storage to either 109 Gbit s^{-1} (direct storage) or 24 Gbit s^{-1} (medium storage). The InfiniBand network also compromised the 8 login nodes and their bandwidth characteristics were 3.2 Gbit s^{-1} throughput and 30 million message/s rate. The whole cluster connected to auxiliary servers such as backup, NFS, etc. via an Ethernet backbone provided of 10 Gbit s^{-1} links (ranging from 8x to 4x). The whole network scheme is depicted in Fig. 2.

3.3 Eagle at PSNC (Poland)

Eagle cluster [26] was commissioned in late 2015 at new PSNC DataCenter facility. Initially, the machine consisted of 1032 nodes, each with two 14-core Xeon E5-2697 (Haswell) CPUs and 56 Gbit s^{-1} FDR InfiniBand interface. It was ranked at the 79th position on TOP500 in November 2015. Inter-cluster InfiniBand network is built on fat-tree topology with a variable blocking factor. All worker nodes are divided into 6 groups which are connected with off-the-shelf 1U 36-port FDR InfiniBand switches, which give 1:4 and 1:2 blocking factors. It depends on tree depth (see Fig 3). After the upgrade, which took place in December 2016 and consisted of additional 55 nodes with two Xeon E5-2682 (Broadwell) CPU, peak performance of the Eagle cluster is 1.4 PFlop s^{-1}. Due to extensive use of DLC (Direct Liquid Cooling) modules and freecoolig capability, PUE (Power Usage Effectiveness) parameter achieves value 1.05.

4 LAMMPS Benchmark Description

LAMMPS (acronym for Large-scale Atomic/Molecular Massively Parallel Simulator) is a well-established molecular dynamics research code that models an ensemble of particles in a liquid, solid, or gaseous state. Its capabilities span atomic, polymeric, biological, metallic, granular, and coarse-grained systems using a variety of force fields and boundary conditions. It is available as open source code written in C++ and supports the MPI message-passing library. Both CPUs and GPUs versions can be compiled (see [27] for further reference).

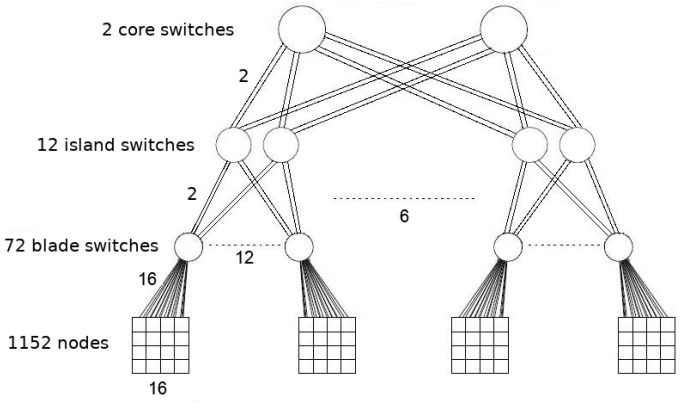

Fig. 3. Architecture of cluster Eagle at PSNC.

4.1 Effect of Atomistic Interactions

Regarding the various LAMMPS benchmarks available as part as the code distributions and other supplied by the research community, some of them correspond to the so-called short-range interaction at the atomistic level. That is, for instance, the case of the LJ-benchmark (which models the three-dimensional rapid melting of an atomic fluid, in which atomistic forces follow a Lennard-Jones (LJ) potential description), in which the atoms interaction caused by the short-range potential (implemented as a 2.5 sigma cutoff distance) involves only 55 neighbors per atom for accurate results. This local physics can be related to the existence of a moderate algorithmic coupling in terms of network traffic among the MPI tasks, which manage the portions (set as cubic bins of atoms) of the whole computational domain, which is partitioned across processors using spatial decomposition. Even in those situations of many bins (that is, tasks) with a rather small number of atoms inside, the information exchange caused by the short-range interactions of those atoms near the bins boundaries does not impact to a great extend the network traffic during the time-integration, so reasonable good weak-scaling properties remains in scale [21]. Table 1 shows the different execution contributions (time spent in the major sections of the code) for the LJ-benchmark with 4,000 atoms in the entire computational domain, executed in Helios. It is seen that communications dominate the whole figure. This is explained by the fact that force interactions are computed very quickly in relative time (a small contribution to the total execution time) and the network traffic cost linked to a bin is mainly caused by the operations performed with its immediate neighboring bins.

4.2 Benchmark RHODO

This system consists of a rhodopsin protein simulated in solvated lipid bilayer with the chemistry molecular simulation module CHARMM [28] for the force

Table 1. LAMMPS algorithmic sections timing breakdown for the LJ-benchmark with 256 MPI tasks in Helios (*Pair*: pairwise atoms interactions. *Neigh*: to compute new neighbours list for each atom. *Comm*: communications time. *Output*: time to output the restart and atom position, velocity and atom forces files).

Algorithmic sections, %					
Pair	Neigh	Comm	Output	Modify	Other
8.77	2.70	87.32	0.12	0.40	0.69

field, besides the long-range solver Particle-Particle Particle-Mesh (PPPM) [29] of LAMMPS to include the Coulombics interaction for accurate results. The 32,000 atom system is made up from counter-ions and a reduced amount of water. The Lennard-Jones cut-off distance is 10 Angstrom and each atom has 440 neighbors. Both force field and long-range solvers differs from the LJ-benchmark case and the model of atomistic interactions has important implications, as long-range interactions lead to many more particle neighbors to be taken into account at each time-step of the integration, which means a more communication-intensive problem to compute.

4.3 Setup and Execution Issues

The installed LAMMPS version corresponds to the November 2016 distribution. The MPI library is the MVAPICH2 distribution already available in Stampede and Helios; and MPIch v.3.1.2 in the case of Eagle. The portable FFTW v3.3.6 library [30] has been installed and linked to the code instead of the native FFT [31], to accomplish the executions. The inclusion of Coulombic interactions requires to build LAMMPS with the KSpace package as it provides the PPPM solver, which in turn executes the FFT operator.

Weak-scaling tests imply the periodic replication of the rhodopsin database (atoms system), then each core manages 32,000 atoms at the start of the simulation. This is done by scripting a replication pattern for the three spatial dimensions in the input file. To balance the load, the bins pattern has been shaped as a cubic computational domain for the tested parallelizations. System replications range from 2^3 to 13^3 (2197 MPI tasks).

Executions for each range of parallelization encompasses three different grouping patterns regarding how the MPI tasks of a prescribed execution distribute over the available nodes: strongly concentrated (all the MPI tasks are allocated within as few nodes as possible, with no more than one task per core); strongly distributed (when possible, the number of nodes involved matches the number of MPI tasks); and something in between, the so-called "intermediate" in the figures legend. The maximum participating number of nodes in the experiments has been 256 in Stampede and 512 in Helios and Eagle supercomputers.

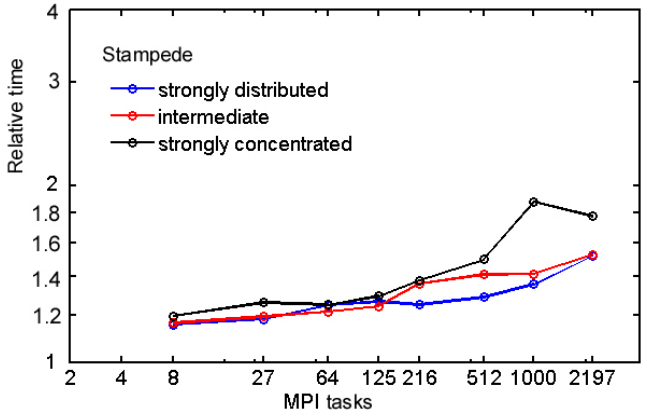

Fig. 4. Benchmark RHODO of LAMMPS executed in Stampede. MPI range is: 8, 27, 64, 125, 216, 512, 1000 and 2197. And MPI processes have been allocated over the nodes with three different grouping patterns (see legend).

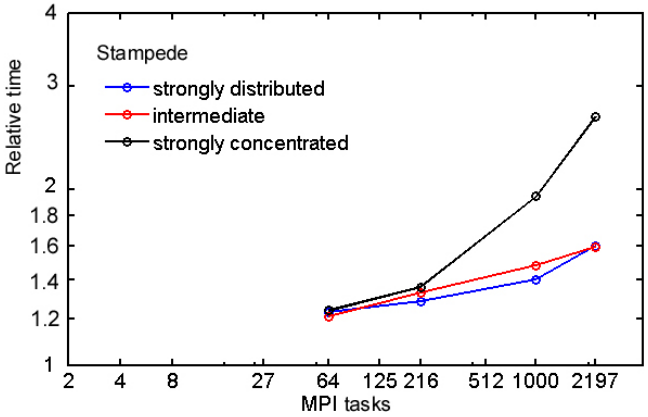

Fig. 5. Benchmark RHODO of LAMMPS executed in Stampede, this time using the local binary of the code (February 2015 release). MPI range is: 64, 125, 216, 512, 1000 and 2197. And MPI processes have been allocated over the nodes with three different grouping patterns (see legend).

5 Results

Non-dimensional execution time obtained in Stampede is compared in Figure 4 for the three grouping patterns. Besides the progressive deterioration of the weak-scaling property with the parallelization range, it is visible that the pattern of distributing MPI tasks over as many nodes as possible provides a shorter execution time (improvement of about 12-18%) compared to the pattern of task concentration for ranges greater than 512 MPI tasks.

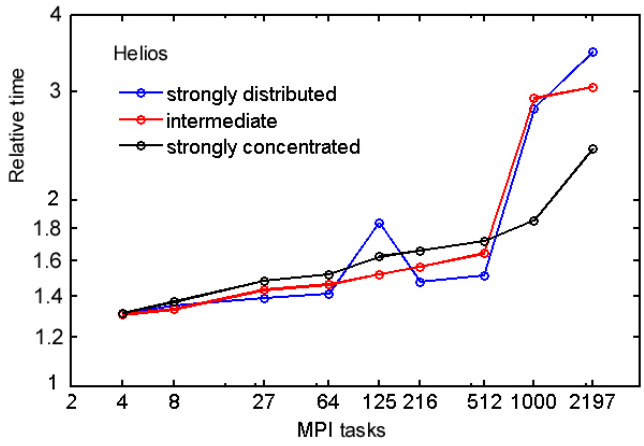

Fig. 6. Benchmark RHODO of LAMMPS executed in Helios. MPI range is: 4, 8, 27, 64, 125, 216, 512, 1000 and 2197. And MPI processes have been allocated over the nodes with three different grouping patterns (see legend).

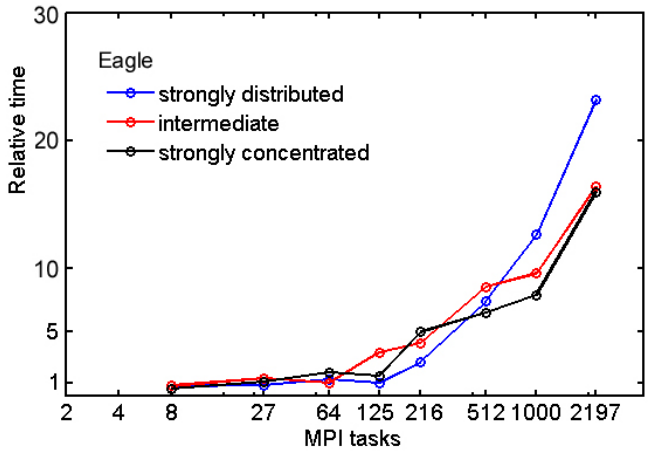

Fig. 7. Benchmark RHODO of LAMMPS executed in Eagle. MPI range is: 8, 27, 64, 125, 216, 512, 1000 and 2197. And MPI processes have been allocated over the nodes with three different grouping patterns (see legend).

For comparison purposes and to check the sensitivity of the results to the particularities of the LAMMPS version and compilation options, similar tests have been carried out with the version of LAMMPS available in Stampede for the users community. These results are plotted in Fig. 5 for the range of interest. Besides the benchmark weak-scaling is worse (especially for concentration), it is seen in the plots that differences between the distributed and concentrated patterns result to be even greater. Analogous plots for non-dimensional execu-

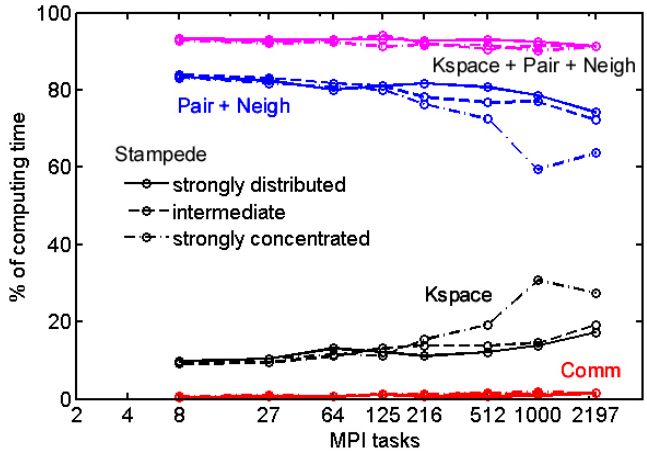

Fig. 8. Benchmark RHODO executed in Stampede. Relative contributions of the major algorithmic sections (Pair, Neigh, KSpace and Comm), executed in Stampede.

tion time have been obtained with Helios (see Fig. 6) and Eagle supercomputers (Fig. 7) and the three grouping patterns. Interestingly, both figures show the opposite behaviour in scale compared to Stampede: now, for a large enough parallelization range (1000 and 2197 MPI tasks), the strong-concentration pattern is the one that provides the shorter execution time (with a saving of up to 31% in Helios, and 61% in Eagle), which points out to the interest of applying this computing strategy. Figures 8, 9 and 10 show the major contributions of the algorithmic sections of the solver LAMMPS to the total computing time for the executions. It is noticed that the FFT operations are performed within the KSpace section, which quickly occupies a large portion of the computing time when the parallelization range is beyond 512 MPI tasks for the strong-distributed pattern.

It is noticed that the KSpace-section involves also data communications in addition to the communications accounted for by the Comm-section in the plots. The impact of the net communications on the execution time observed in Helios and Eagle seems to be driven by the KSpace-section as it peaks, which occurs at about 512 and 216 MPI tasks (respectively) for the strongly distributed pattern: the KSpace-section becomes very computing intensive from that amount of MPI tasks on, so grouping task in as few nodes as possible improves the execution time compared to Stampede (which exhibits a much more smooth KSpace-section behaviour in scale, as seen in Fig. 8).

6 Conclusions

The weak-scaling tests of LAMMPS presented in this work stress the weight of the communications on the solver behavior as a function of the grouping pattern of MPI tasks over the nodes. The experiments have been accomplished

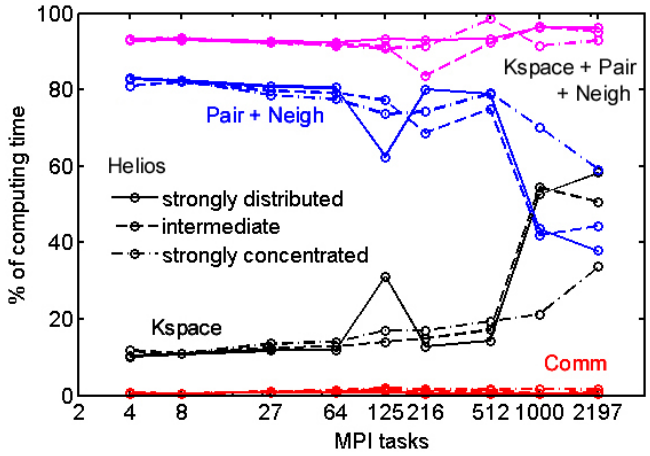

Fig. 9. Benchmark RHODO executed in Helios. Relative contributions of the major algorithmic sections (Pair, Neigh, KSpace and Comm), executed in Helios.

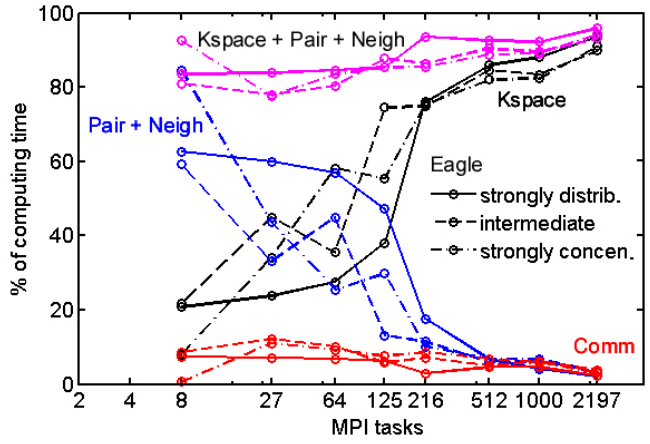

Fig. 10. Benchmark RHODO executed in Eagle. Relative contributions of the major algorithmic sections (Pair, Neigh, KSpace and Comm), executed in Eagle.

on three supercomputers: Stampede at TACC (USA), Helios at IFERC (Japan), and Eagle at PSNC (Poland). For this, the physics linked to long-distance molecular interactions has been chosen, which in terms of modeling requires the frequent solution of parallelized FFTs. Algorithmically, it implies a strong coupling among MPI tasks. Besides, counting on an available prefixed number of nodes and/or cores in a given cluster, a decision on the problem size (i.e. the setup of the amount of atoms per core) influences whether the simulation will be communication-intensive. This fact, jointly with the knowledge of the supercomputer network topology, indicates if it will worth concentrate or not tasks

on as few nodes as possible. The conducted study paves the way for a better usage of the whole cluster. In this sense, it may also help to improve the clusters throughput by, for instance, adding live-migration decisions in the scheduling policies in those cases of communication-intensive behavior. Also, it opens a similar output from the energy consumption point of view, profiting from the same live migration decisions. As a consequence, future work can be designed to including other codes which mimics similar communication-intensive behavior (FFTs, implicit algorithms, etc), such that would provide additional outcomes and a more precise evaluation of the impact of this family of codes.

Acknowledgment

This work was partially funded by the Spanish Ministry of Economy, Industry and Competitiveness project CODEC2 (TIN2015-63562-R) with European Regional Development Fund (ERDF) as well as carried out on computing facilities provided by the CYTED Network RICAP (517RT0529) and Poznań Supercomputing and Networking Center.

References

1. TOP500 Supercomputers homepage, http://www.top500.org
2. Shalf J., Quinlan D., Janssen C.: Rethinking Hardware-Software Codesign for Exascale Systems. Computer **44**(11), 22–30 (2011). https://doi.org/10.1109/MC.2011.300
3. Exascale Computing Project (ECP) homepage, https://www.exascaleproject.org
4. http://eurohpc.eu
5. Partnership Research for Advance Computing in Europe, http://www.prace-ri.eu
6. National Supercomputing Center in Tianjin homepage, http://www.nscc-tj.gov.cn
7. Post-K supercomputer: www.fujitsu.com/global/Images/post-k-supercomputer.pdf
8. Jeannot E., Mercier G., Tessier F.: Process Placement in Multicore Clusters: Algorithmic Issues and Practical Techniques. In: IEEE Transactions on Parallel and Distributed Systems **25**(4) 993–1002 (2014). https://doi.org/10.1109/TPDS.2013.104
9. Chavarría-Miranda D., Nieplocha J., Tipparaju V.: Topology-aware Tile Mapping for Clusters of SMPs. In: Procs. Third Conference on Computing Frontiers, Ischia, Italy (2006). https://doi.org/10.1145/1128022.1128073
10. Smith B., Bode B.: Performance Effects of Node Mappings on the IBM BlueGene/L Machine. In: Cunha J.C., Medeiros P.D. (eds.) Euro-Par 2005 Parallel Processing. Lecture Notes in Computer Science, vol 3648, pp. 1005–1013, Springer, Berlin, Heidelberg (2005). https://doi.org/10.1007/11549468_110
11. 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 Applica- tions. In: Proc. IEEE Symposium Computers and Comm. pp. 811–817, Sousse, Tunisia (2009). https://doi.org/10.1109/ISCC.2009.5202271
12. León E.A., Karlin I., Moody A.T.: System Noise Revisited: Enabling Application Scalability and Reproducibility with SMT. In: IEEE International Parallel and Distributed Processing Symposium, pp. 596–607. IEEE, Chicago, USA (2016). https://doi.org/10.1109/IPDPS.2016.48

13. 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: Proc. 7th IEEE Int. Symposium Cluster Computing and the Grid (CCGrid), pp. 471–478, Rio De Janeiro, Brazil (2007). https://doi.org/10.1109/CCGRID.2007.119
14. Shainer G., Lui P., Liu T., Wilde T., Layton J.: The Impact of Inter-node Latency versus Intra-node Latency on HPC Applications. In: Proc. IASTED Int. Conference Parallel and Distributed Computing and Systems, pp. 455–460 (2011). https://doi.org/10.2316/P.2011.757-005
15. 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), pp. 4–7, Atlanta, USA (2009).
16. Ribeiro C.P. et al.: Evaluating CPU and Memory Affinity for Numerical Scientific Multithreaded Benchmarks on Multi-cores. International Journal on Computer Science and Information Systems **7**(1), pp. 79–93 (2012)
17. Xingfu W., Taylor V.: Processor Partitioning: An Experimental Performance Analysis of Parallel Applications on SMP Clusters Systems. In: 19th Int. Conference Parallel Distributed Computing and Systems (PDCS07), pp. 13–18, Cambridge, Massachusetts, USA (2007)
18. Rodríguez-Pascual M., Moríñigo J.A., Mayo-García R.: Benchmarking Performance: Influence of Task Location on Cluster Throughput. In: Communications in Computer and Information Science 796, pp. 125–138, Springer, Heidelberg (2017). https://doi.org/10.1007/978-3-319-73353-1_9
19. Moríñigo J.A., Rodríguez-Pascual M., Mayo-García R.: Slurm Configuration Impact on Benchmarking. In: Slurm User Group Meeting, Athens, Greece, (2016). https://slurm.schedmd.com/publications.html
20. Chi Zhang, Xin Yuan A.S.: Processor Affinity and MPI Performance on SMP- CMP Clusters. In: IEEE Int. Symposium Parallel and Distributed Processing, Workshops and PhD Forum, pp. 1–8, Atlanta, USA (2010). https://doi.org/10.1109/IPDPSW.2010.5470774
21. McKenna G.: Performance Analysis and Optimisation of LAMMPS on XCmaster, HPCx and BlueGene. MSc, University of Edinburgh, EPCC (2007).
22. Liu J.: LAMMPS on Advanced SGI Architectures. White Paper SGI (2010)
23. Cornebize T., Heinrich F., Legrand A., Vienne J.: Emulating High Performance Linpack on a Commodity Server at the Scale of a Supercomputer, HAL-id: hal-01654804 (2017)
24. Stampede supercomputer: https://www.tacc.utexas.edu/systems/stampede
25. Helios supercomputer: http://www.iferc.org/CSC_Scope.html#Systems
26. Eagle supercomputer: https://wiki.man.poznan.pl/hpc/index.php?title=Strona_glowna
27. LAMMPS homepage, http://lammps.sandia.gov
28. CHARMM homepage, https://www.charmm.org
29. Plimpton S., Pollock R., Stevens M.: Particle-Mesh Ewald and rRESPA for Parallel Molecular Dynamics Simulations. In: Eighth SIAM Conference on Parallel Processing for Scientific Computing (1997)
30. Fast Fourier Transform of the West homepage, http://www.fftw.org
31. Plimpton S.: Fast Parallel Algorithms for Short-Range Molecular Dynamics. Journal of Computational Physics, **117**(1), pp. 1–19 (1995). https://doi.org/10.1006/jcph.1995.1039