PERFORMANCE DROP AT EXECUTING COMMUNICATION-INTENSIVE PARALLEL ALGORITHMS

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

[1] Centro de Investigaciones Energéticas, Medioambientales y Tecnológicas (CIEMAT), Avda. Complutense 40, Madrid 28040, Spain
[2] Oak Ridge National Laboratory, 1 Bethel Valley Rd, Oak Ridge, TN 37830, USA
[3] Poznan Supercomputing and Networking Center, Jana Pawla II 10, 61-139 Poznan, Poland

* Corresponding author: josea.morinigo@ciemat.es

Abstract. This work summarizes the results of a set of executions completed on three fat-tree network supercomputers: Stampede at TACC (USA), Helios at IFERC (Japan), and Eagle at PSNC (Poland). Three MPI-based, communication-intensive scientific applications compiled for CPUs, have been executed under weak-scaling tests: the molecular dynamics solver LAMMPS; the finite element-based mini-kernel miniFE of NERSC (USA); and the three-dimensional Fast Fourier Transform mini-kernel bigFFT of LLNL (USA). The design of the experiments focuses on the sensitivity of the applications to rather different patterns of task location, to assess the impact on the cluster performance. The accomplished weak-scaling tests stress the effect of the MPI-based application mappings (concentrated vs. distributed patterns of MPI tasks over the nodes) on the cluster. Results reveal that highly distributed task patterns may imply a much larger execution time in scale, when several hundreds or thousands of MPI tasks are involved in the experiments. Such a characterization serves users to carry out further, more efficient executions. Also researchers may use this experiments to improve their scalability simulators. In addition, these results are useful from the clusters administration standpoint since tasks mapping has an impact on the cluster throughput.

Keywords: Cluster throughput, communication-intensive algorithms, MPI application, weak scaling.

1. INTRODUCTION

Current High Performance Computing (HPC) architectures span thousand of nodes, 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 trend usually has fewer nodes compared to the second one and it allows more allocated parallel tasks per node. Anyway, both scenarios exhibit a huge amount of cores (see the TOP500 list [1] of the most powerful supercomputers). The evolution of CPUs over the last years has driven the inclusion of an increasing degree of parallelism within the codes executed by the final users.

Extreme computing demands the optimization of performance, cost and power consumption. Hardware and software stacks must be built and operated bearing in mind the applications that will run as well as the next available computing infrastructure. This approach is known as codesign [2], in which the different elements involved in supercomputing are tightly integrated.
At present these issues are being tackled by a variety of initiatives in the USA (Exascale Computing Project) [3], Europe (EuroHPC [4] and PRACE [5]), China (Tianjin Supercomputer Center) [6], and Japan (Post K) [7]. In particular, they are boosting the further development of MPI libraries, not only to take advantage of new technologies, but to address the current networks complexity in supercomputers with thousands of nodes. However, a careful implementation of a message passing strategy is not enough since the computing platform topology (in particular the way in which the cluster interconnections have been designed) may contribute to traffic bottlenecks and network contention, which leads to the speed down of the system. This turns to be problematic for massive parallel applications, as far as inter-node communication asymmetries may develop, thus contributing to a performance drop. Causes of performance degradation must be analyzed in detail (even

supercomputers designed with either InfiniBand or OmniPath can exhibit a worse behavior as the number of involved nodes and cores increases) following an adequate setup of experiments. Therefore, it is cornerstone to assess the performance sensitivity to the MPI-tasks mapping for a given application executed on the cluster under weak-scaling with an approximately constant workload –per-node, in order to assess the effect of task spreading over the nodes for increasing degree of parallelism of the running application.

Applications may be CPU-intensive, memory-bandwidth demanding (communication-intensive) and/or I/O-intensive, such that leads the scalability and speedup to stagnate. Typically, users have access to finite resources and constrains derived from several factors. The maximum degree of parallelization attainable or the total execution time granted in a computing resource, arise in practical cases. The context is complex and scheduling decisions in HPC systems have also an impact on the applications performance. For instance, one situation could be a given job which only uses some of the CPUs of a node, or just a subset of the cores of a CPU. An opposite scenario would be having the resources shared by two or more different jobs, so it may lead to a competition in terms of memory-bandwidth and, consequently, to slow down the whole execution. Unlike, executing some other job at the same time on the same node/CPU could imply a more intensive usage of the cluster. Anyway, what it is expected independently of sharing CPUs of a node or performing executions with dedicated nodes, is the existence of colliding network traffic to some extent, due to its concurrent usage by the supercomputer community. To this regard, the supercomputer topology plays a major role on the network traffic effects.

The present work extends our previous investigation [8] and explores the behavior of three communication-intensive applications (that is, a benchmark of the production solver LAMMPS; and two mini-kernels miniFE and bigFFT) executed on three fat-tree network based supercomputers under a weak-scaling perspective. The results show that runtime strongly depends on the chosen task-mapping pattern and that optimal task-mapping at extreme scale deserves careful attention to attain a better computational efficiency of the communication-intensive applications. Besides, the conducted experiments are useful to assess the correctness of scalability simulators

at predicting performance drop or cluster efficiency in these types of executions. The article is structured as follows. The related work is summarized in the next section, whereas the description of the HPC clusters is briefly presented in Section 3. After that, a description of the experiments conducted with the miniFE, LAMMPS and bigFFT codes, including the major statistical data of the experiments, is provided in Section 4. Section 5 summarizes the results and discussion. Finally some conclusions are given.

2. RELATED WORK

MPI task mapping effects have been investigated in [9] with scientific mini-kernels and application codes. They show that an execution time saving of up to 25% is possible by grouping the tasks. In [10] machine topology is taken into account and the 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 [11], reporting an improvement of about 15%. In particular, in [12] multicore architectures show the gain attained in computational efficiency of a MPI-based production application, which exhibits a performance peak improvement of about 9%, 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 seems a modest speedup, it is noticed that it is obtained with minor source code modifications and mostly using an optimized task mapping over the nodes.

Most of the above mentioned studies are limited to a moderate number of CPUs and additional research is needed to be conclusive about what happens with many participating processors in weak-scaling benchmarking. In particular, various scientific kernels and production codes have been tested in scale in [13] under weak-scaling. It is observed that runtime degradation greater than 120% occurs when the same number of MPI tasks are distributed massively (typically, one task per node).

The work in [14] 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. The impact of the

different internode and intranode latencies is analyzed in [15] for an InfiniBand-based cluster running a MPI-based scientific application. Aiming at improving the computational efficiency, [16] analyzes how many cores per node should be used for applications execution. These studies identify that performance degradation in multi-core machines is rather dependent on task mapping, 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. Indeed, by distributing the tasks over the nodes, they do not have to compete for node local resources (a scenario that seems to occur when running tasks are sharing a slot of the same node). The influence of this resource sharing by different jobs is focused in [17] using a set of scientific kernels. The results show significant dependence to the cluster setup (further details on the clusters setups used in those tests are given in [18]). Several studies points at this performance sensitivity to the cluster architecture [19] and specifically to the inter- to intranode communication ratio [20], hence according to their experiments, they recommend to distribute the MPI tasks over the nodes to attain the best results from a computational standpoint (an improvement of about 20% for most of the experiments).

Previous work on assessing communication-intensive scientific applications examines the role played by the network topology and communication bandwidth on speeding performance up at scale. This is analyzed in [21] with two benchmarks of the solver LAMMPS (one of them corresponds to the rhodopsin, focused on the present investigation), executed with a range of parallelization up to 1024 cores in three clusters. The strong-scaling behavior of the same benchmarks executed in three clusters but for a small number of cores, is analyzed in [22]. This investigation quantifies the variation of the execution time to re-configuring the topology of the clusters, which is of about 9%. The effect of changing the cluster interconnections has been also examined in [23-25]. In particular, the comparison of the attained speedup by means of improving the bandwidth using multi-rail InfiniBand networks [26], as well as implementing variants of the fat-tree topology [25] is investigated. These quantifications performed with micro-benchmarks and scientific applications like LAMMPS and three-dimensional FFTs libraries, are helpful in the sizing process of a cluster, bearing in mind that the attained performance will be quite dependent on the type of benchmarked application. Other promising via to quantify the expected

performance given a suite of codes is the usage of cluster emulators, which take into account the network topology in detail, as it is described in [27] with a modeled HPC resource. Since access to computing time and range of parallelization constrains may be scarce, a methodology to gather large-scale simulation statistics is the usage of emulators to shed light onto the effect of varying the supercomputer topology on the performance and on the communication requirements of scientific applications. Besides emulators, trace-based simulation has been applied to study the impact of MPI communications on the performance of LAMMPS and miniFE [28], which are two of the three codes analyzed in the present investigation. Corresponding data of executions in cluster, eed these kind of simulations and the discussion on major aspects of the abovementioned approaches (accuracy and feasibility, to mention some).

It is clear that an adequate cluster topology is a key element to improve applications performance and it remains an open arena in the frame of optimizing the computing resources. Besides and from the standpoint of exploiting a given cluster (that is, a given topology) it is important to identify the best way of mapping tasks of parallel applications, hence to speed the execution up. In [29], a topology-aware resource allocation policy is analyzed, to optimize the allocation of nodes to jobs with different communication-intensive patterns in a fat-tree cluster. It shows that topology-aware node allocation for production workloads can provide interference-free executions with a minimum deterioration of the cluster's quality of service when isolated job partitions are implemented.

To this respect, the behavior of a suite of communication-intensive MPI-based scientific applications, among others, have been analyzed in scale in [13, 23, 30]. A large variation of the execution time when executed highly distributed and concentrated over the nodes is reported and the identification of the weight of the point-to-point and collective operations is pointed out as a major cause of the performance drop in those highly distributed scenarios.

All these previous investigations show that the execution time varies significantly because there is a large sensitivity to the MPI tasks mapping over the nodes, driven by multiple architectural factors. It is stressed that most of these experiments have been conducted in clusters with hyperthreading disabled (one thread per core), as our present investigation does, so they do not analyze possible advantages derived

from multiple-thread based executions, which is a topic analyzed in [31]. Nevertheless, one major objection related to multi-threading is the likely lack of an efficient implementation of the applications (both full-fledge and mini-applications) to improve scalability. In addition, the selection of a group of production applications as representative benchmarks remains nowadays a hot topic: many suites are available for benchmarking, albeit no unified, metric-based procedure of HPC systems has been clearly stated [32].

The present work extends the analysis conducted in [8] and explores the speedup sensitivity to varying the MPI-tasks of three communication-intensive benchmarks mapped over the nodes and executed under a weak-scaling approach. The scientific applications are the production code LAMMPS and the mini-kernels miniFE and bigFFT, compiled for CPUs in three modern computing infrastructures (Stampede at TACC (USA), Helios at IFERC (Japan), and Eagle at PSNC (Poland)) with different fat-tree topology configurations.

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.

3. SUPERCOMPUTER ARCHITECTURES

Next, a brief description of the supercomputers involved in the executions of the scientific application LAMMPS and the kernels miniFE and bigFFT is provided. It is noted that these three supercomputers rely on the fat-tree network, which is partly a consequence of its easily extensible assembly from off-the-shelf commodity components. Furthermore, the fat-tree topology gives high bisection bandwidth and a relatively low diameter among the available options for a given node count. It is noticed that these topologies necessarily are constrained by technological, room and economic factors, resulting in tapered networks with extra hops. These are distinguishing features among the supercomputers, which imply an impact on the

weak-scaling behavior of every application. Nevertheless, it is not the only aspect to consider. Memory and processor's cache availability, bus clocking or the intra-node communication among processors are architecture-related issues wh drive the global performance.

3.1 Stampede at TACC (USA)

Stampede supercomputer [33] was ranked 6th in the TOP500 list (June 2013) by achieving 5168 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 decommissioned and has evolved towards Stampede-2 architecture. The Stampede platform consisted of 6400 Sandy Bridge EP nodes, each with two 8-core Xeon E5-2680v1, which offers 2.7GHz 5GHz with Turbo Boost) connected between them through two Intel QuickPath links. At least 2GB memory per core is guaranteed. Additionally, every node had one Intel Xeon Phi KNC MIC co-processor (not used in the experiments of this study).

The nodes were interconnected through a 56 Gbit s^{-1} FDR InfiniBand two-level Clos fat-tree topology built on Mellanox switches. This fat-tree had a blocking factor of 1,25:1 and its network topology is sketched in Figure 1. The 6400 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 8614 TFlop·s^{-1}.

3.2 Helios at IFERC (Japan)

Helios supercomputer [34] was owned by the Computational Simulation Centre (CSC) and ranked 38th in the TOP500 list when it was in full operation status in November 2014 as it provided a performance peak value of 1524 TFlop s^{-1}. After several upgrades, it finally counted on 4500 nodes (72 000 CPU cores), which were complemented with 180 MIC nodes (21600 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 nodes were based on the Sandy-Bridge configuration with two Xeon E5-2680v1 processors (16-core nodes). In

contrast to Stampede, Helios nodes were connected to QDR InfiniBand fat-tree with a blocking factor 1:1 (that is, non-blocking), but nodes exhibited double latency and half bandwidth compared to Stampede. A sketch of its topology is in Fig. 1. This connection grouped the computing nodes in sets of 18 which were connected to storage to either 109 Gbit·s^{-1} (direct storage) or 24 Gbit·s^{-1} (medium storage). The InfiniBand network also comprised 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. Helios supercomputer is already decommissioned.

3.3 Eagle at PSNC (Poland)

Eagle cluster [35] was deployed in late 2015 at new PSNC Data Center facility. Initially, the machine consisted of 1032 nodes, each with two 14-core Xeon E5-2697v3 (Haswell-EP) CPUs and 56 Gbit·s^{-1} FDR InfiniBand interface.

Albeit these processors are the third evolution of the ones installed in Stampede and Helios, they offer a little improvement: 2.6GHz (3.6GHz with Turbo Boost); same L2 cache per core; nearly double L3 cache; and at about 15% and 33% of acceleration through QPI links and main memory, respectively.

It was ranked at the 79th position TOP500 in November 2015. The main difference with the other two clusters is its InfiniBand network, which comprises a variable tapering. All worker nodes are divided into 6 groups (e.g. islands or pods, similar to [13, 36]). which are connected with off-the-shelf 1U 36-port FDR InfiniBand switches, which give 4:1 (inside pods) and 24:1 (inter-pod) blocking factors. Thus, it depends on the tree depth (see Fig. 1). After the upgrade, which took place in December 2016 and consisted of additional 55 nodes with two Xeon E5-2682 (Broadwell) CPU, the peak performance of the Eagle cluster is 1.4 PFlop·s^{-1}.

4. SCIENTIFIC APPLICATIONS

A short description of the MPI applications whose sensitivity to different task mapping pattern is examined, is given. The chosen applications implement communication-intensive algorithms (basically, multidimensional Fast Fourier Transforms (FFTs) and

implicit discretization of partial differential equations) to test their effect on runtime variability caused by network traffic when the MPI tasks are highly concentrated and highly distributed over the nodes. A priori, their performance will depend to a high extend on the available bandwidth provided by the links and switches of the network.

4.1 LAMMPS

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 (see [37] for further reference). The installed LAMMPS version in the three clusters corresponds to the November 2016 distribution for CPU and MPI (it is noticed that, in order to check the sensitivity to the LAMMPS release, additional tests have been carried out with the February 2015 distribution in Stampede, for comparison purposes).

Among the various benchmarks provided in LAMMPS distribution, the one analyzed in this work is the rhodopsin, which simulates a protein in a solvated lipid bilayer. It is complemented with the chemistry molecular simulation module CHARMM [38] to tackle the force field, besides the long-range Particle-Particle Particle-Mesh (PPPM) solver [39, 40] 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. This benchmark models a long-range interaction of Coulombic nature at the atomistic level, which needs from frequent FFTs transformations. Both force field and long-range solver differ from those short-range interaction benchmarks, also available in LAMMPS (for instance, see the LJ-benchmark, that models the 3D rapid melting of an atomic fluid, in which atomistic forces follow a Lennard-Jones (LJ) potential), in the atoms interaction, here based on a short-range potential and a small number of involved neighbors per atom. This rather small number of atoms inside each computational bin implies that the information exchange does not impact to a great extent the network traffic during the time-integration, so reasonable good weak-scaling properties remains in scale [21].

On the contrary, the inclusion of a long-range interaction leads to many more particle neighbors (each rhodopsin atom is surrounded by 440 neighbors, that is, at about one order of magnitude greater than the short-range case) to be taken into account at each time-step of the integration, which means a more communication-intensive problem to compute (force interactions are computed very quickly and they are a small contribution to the total execution time. In addition, network traffic cost linked to a bin is mainly caused by the operations performed with its immediate neighboring bins). The consequence of this local physics is a strong algorithmic coupling, which implies higher network traffic among the MPI tasks, linked to the cubic bins of atoms in the partitioned computational domain across processors using spatial decomposition.

4.2 miniFE

Release 1.4 of this scientific mini-application, developed at Sandia National Laboratory (USA), comprises the algorithm complexities of the unstructured implicit finite elements or finite volume solvers, within a size of less of 8,000 lines of code.
It simulates the steady-state heat conduction on a brick-shape problem domain. The discretization of the involved partial differential equation is accomplished with linear 8-node hex-elements, assembled as a sparse linear system.

The numerical approach follows a simple un-preconditioned conjugate-gradient (CG) algorithm [41], which implies sparse matrix-vector products during the CG iteration. These data are set in local memory over the participating nodes. Implicitness of the numerical method and data locality leads to a communication-intensive scenario as the problem size increases [23, 30]. The miniFE has support for OpenMP (not used in the present computations) and it is parallelized with MPI. The physical brick-shape size (discretized into a prescribed number of grid cells per spatial dimension in the setup of the problem) serves to control the weak-scaling properties for a given number of participating cores in the simulation.

4.3 bigFFT

Three-dimensional FFTs are one of the most compute- and communication-intensive schemes in applications from a variety of fields, ranging from direct numerical

simulations of turbulence, astrophysics, molecular dynamics, material sciences or tomography, among others.

Release 1.0 of this scientific mini-application [42] developed at Lawrence Livermore National Laboratory (LLNL, USA), implements a real-to-complex MPI-based parallel 3D FFT using a 2D virtual grid in which the tasks are arranged (an alternate version with OpenMP threading is available from the developers on request, not analyzed in the present work). The code itself is a test of a pencil-based algorithm, which is a 2D domain-decomposition FFT, in which each core / task is responsible for a rectangular column (pencil) of the data array. The kernel exploits the FFTW library [43] to perform the 1D FFT needed for the three dimensions in turn. The pencil-based implementation theoretically allows scaling up to the number of tasks squared. Initialization of the 3D data array of specified dimensions (N_xxN_yxN_z, being N_x the pencil length) is done by Gaussian randomization. A sequence of forward and backward 3D FFT is completed at execution with a processor grid of specified size (np_{row} x np_{col}, equal to the total number of MPI tasks). Communications are driven by transposes of large arrays.

5. EXECUTIONS

5.1 LAMMPS

Rhodopsin benchmark has been selected as an example of strongly-coupled behavior, in contrast to other short-range algorithms included in the LAMMPS release, The inclusion of long-range Coulombic interactions requires building the code with the KSpace package as it provides the PPPM solver (it executes the FFT operator). In this sense, instead of using the native FFT available in LAMMPS [34], the portable FFTW library has been installed and linked to accomplish the executions. Compilation is carried out with the Intel compiler.

Weak-scaling tests imply the periodic replication of the rhodopsin database (atoms system inside a computational bin), then each core manages 32,000 atoms at the simulation start. This is done by scripting a replication pattern for the three spatial dimensions in the input file. To balance the load, the replication pattern has been shaped as a cubic computational domain. Parallelization ranges from 2^3 to 13^3 (2197 MPI tasks). The set of experiments carried out is shown in Table 1.

Table 1. List of LAMMPS experiments executed. The range of parallelization is equal to the total number of replicated bins (each assigned to a dedicated core).

Range of parallelization	Bins replication (in each spatial dim.)	Number of participating nodes
8	2	1, 4, 8
64	4	16, 32, 64
125	5	25, 75, 125
216	6	16, 32, 108
512	8	64, 128, 256
1000	10	50, 125, 250
2197	13	140, 200, 256

Table 2. List of miniFE experiments executed. Brick-shaped computational domains of size N x N x N are specified.

Range of parallelization	Grid size N x N x N	Number of participating nodes
8	504 x 504 x 504	1, 4, 8
16	635 x 635 x 635	1, 8, 16
32	800 x 800 x 800	4, 8, 32
64	1008 x 1008 x 1008	8, 16, 32
128	1270 x 1270 x 1270	16, 32, 64
256	1600 x 1600 x 1600	32, 64, 128
512	2016 x 2016 x 2016	64, 128, 256
1024	2540 x 2540 x 2540	128, 256, 512 (1)
2048	3200 x 3200 x 3200	128, 256, 512 (1)

(1) Limited to 256 nodes in Stampede. Participating nodes: 128, 169, 256 because of queue constrains.

5.2 miniFE

Finite-element discretization implies to assemble a sparse system of equations, implicitly solved. The physical domain corresponds to a cube. The cube sizing for two consecutive ranges of parallelization (RP) is: $RP_{i+1} / RP_i = 2 \approx (N_{i+1}/N_i)^3$, to fulfill the load balance in the weak-scaling tests (see Table 2).

5.3 bigFFT

Each pencil of the 2D domain-decomposition FFT is assigned to a dedicated core. Theoretical load-balance of the weak-scaling tests is enforced by doubling the $N_y x N_z$ dataset size following the doubling of the range of parallelization. As in the previous cases, the grouping pattern (strongly-concentrated, intermediate and strongly-distributed) is identified in Table 3 with the number of participating nodes.

Table 3. List of bigFFT experiments executed. Datasets correspond to $N_x x N_y x N_z$ (the range of parallelization is equal the number of pencils np_{row} x np_{col}).

Range of parallelization	Size			Pencils		Number of participating nodes
	N_x	N_y	N_z	np_{row}	np_{col}	
8	1024	512	512	4	2	2, 4, 8
16	1024	1024	512	4	4	4, 8, 16
32	1024	1024	1024	8	4	8, 16, 32
64	1024	2048	1024	8	8	16, 32, 64
128	1024	2048	2048	16	8	32, 64, 128
256	1024	4096	2048	16	16	64, 128, 256
512	1024	4096	4096	32	16	82, 128, 256
1024	1024	8192	4096	32	32	128, 169, 256
2048	1024	8192	8192	64	32	128, 167, 256

Due to computing time restrictions followed by the decommissioning of the Helios supercomputer, bigFFT tests have been completed only in Stampede and Eagle.

5.4 Setup and Statistical Issues

The MPI library used in Stampede and Helios is MVAPICH2. MPIch v3.1.2 has been used in the Eagle tests. Their compilation was done with the Intel compiler.
The set of executions for every range of parallelization encompasses three different grouping patterns regarding how the MPI tasks have been mapped 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 or "intermediate".

Executions have been accomplished requesting nodes in exclusivity, so allocated tasks are not perturbed by resource-competing jobs running within them. The participating nodes and task-to-core mapping are chosen automatically by the resource manager (Slurm in the three supercomputers), hence the cores and nodes of an experiment are set according to the resource manager setup parameters, not requesting neither excluding specific node locations. The maximum participating number of nodes in the experiments has been 256 in Stampede and 512 in Helios and Eagle supercomputers.

These supercomputers have been exploited by multiple users while the tests of the present investigation have been completed and gathered, such that a variety of applications with diverse requirements, have been executed at the same time. Average workload has been frequently over 90%, thus affecting the applications performance. It seems clear that this high resource occupacy scenario implies that communications among distributed tasks of a given application, will be affected to some extent by the instantaneous traffic (mostly in those applications that spend a significant portion of communication time). Furthermore, sensitivity to system noise will be also an issue.

Therefore, a number of repetitions, set to 30 in the present study, have been accomplished for every specified case (see Tables 1-3) in order to quantify the variations among runs and to obtain average execution times as a metric. Table 4 shows an example of the observed variability for the LAMMPS benchmark ranges of parallelization ran in Stampede. More details on this issue will be discussed in the results section.

Table 4. Execution time variability with LAMMPS driven by the network traffic in Stampede.

Number of MPI tasks	216	512	1000	2197
Release Nov. 2016	± 11.8%	± 11.2%	± 10.7%	± 12.8%
Release Feb. 2015	± 11.0%	± 9.9%	± 11.2%	± 12.3%

Table 5. Intranode execution time (in seconds) for the application LAMMPS, miniFE and bigFFT in the three clusters.

	Stampede	Helios	Eagle
	LAMMPS		
8 MPI tasks	579	654	548
	miniFE		
4 MPI tasks	84	127	219
8 MPI tasks	123	343	266
	bigFFT		
8 MPI tasks	289	-	133

Table 6. Execution time for bigFFT, parallelized with 16 and 2048 processes (tasks) in cluster Eagle. PPN stands for processes (MPI tasks) per node.

16 processes		2048 processes	
PPN	Time (s)	PPN	Time (s)
1	153	8	3219
2	205	12 -13 (167 nodes)	1982
4	251		
8	411		
16	446	16	1410

6. RESULTS

Table 5 shows the attained execution time of the benchmarks in the three clusters, ran under a small parallelization (8 tasks). It corresponds to the job mapped within one dedicated node, so inter-node communications are avoided. This execution time is taken as the reference execution time and used to non-dimensionalize the respective range of parallelizations tested. Table 5 shows that the reference execution time exhibits a significant dependence with the involved supercomputer. Interestingly, the variation of the reference execution time is notably smaller in the case of the production code LAMMPS, than for the mini-kernels miniFE and bigFFT, which exhibit a wider sensitivity to the local environment where have been compiled under identical optimization flags. Besides, supercomputers are multi-user environments and as a result, their network traffic superimposed to the system noise affects the benchmarks execution to some extent, which must be quantified. Since in most cases it is unfeasible to have access to a supercomputer in exclusivity, a set of repetitions of a given benchmark execution should be done to estimate average runtimes of the scientific applications under analysis. Following some initial tests, the number of repetitions has been set to 30 times, which turns to be adequate to estimate the average execution time and variability. An example of the sensitivity of

this approach is shown in Fig. 2 for the rhodopsin benchmark of LAMMPS ran in Stampede. Albeit scalability behavior is not homogeneous for this subset of scientific applications, which will be shown in what follows, the sensitivity of miniFE and bigFFT to the number of repetitions of the tests is similar to LAMMPS, hence 30 repetitions is also prescribed as adequate and balanced with the available computing time in these platforms for the required accuracy.

The statistical dispersion observed for the runtime of the three applications is mostly attributed to the network traffic and congestion. Also switches hops are a plausible cause to explain this variability. To this respect, fat-tree network visualization tools [45] and traffic simulation [46] (it should be stressed that performance depends on the job placement and network rooting on fat-tree clusters, besides it is driven by the application communication pattern), may provide insight in the involved fat-tree network congestion and job interference by other communication-heavy jobs in highly utilized HPC clusters.

The examination of the LAMMPS executions on Stampede, reveals that the maximum variability is of about ± 13%. Table 4 includes estimates of the variability for the largest ranges of parallelization tested with two releases of LAMMPS. Execution of LAMMPS benchmark on the other clusters Helios and Eagle implies an interval of variability (see Fig. 3) of about ±17.6% and ±21.6%, respectively.

The combination LAMMPS-Stampede shows that those tests corresponding to the largest ranges of parallelization, run with a strongly-distributed pattern, are faster by 17-22% according to Fig. 4 (above) and the more recent version of LAMMPS tested. Comparing this speedup with the variability interval, these are numbers of the same order, so a cancellation of such speedup advantage might occur in some runs. However, there is a net gain of about 20% in average, as it is shown in Fig. 4. It is noted that a set of analogous runs have been carried out with the native version of LAMMPS available in Stampede for the users community, in order to check the sensitivity of the results to the code version and compilation options. In that case, even a faster execution takes place (see Fig. 4, below).

LAMMPS executions in Helios and Eagle show that the strongly-concentrated pattern runs faster for large ranges of parallelization (>512 and >216, respectively. See Figs.

5 and 6). This is opposite to the behavior observed in Stampede (see Fig. 4). That is, Helios and Eagle results imply that it is better to allocate the job within as few nodes as possible. For both clusters, an abrupt deterioration of scalability occurs at the same time the tendency observed in Stampede is reversed. This sensitivity of the runtime to changes in the cluster topology has been reported in [22] also for the rhodopsin benchmark, which gives an estimate of about 9%.

A more detailed information is provided in Fig.7, where the averaged contribution of the computing cost of the major algorithmic sections of LAMMPS to the total, is shown. It is important to notice that the KSpace-section comprises the FFTs, so its contribution to the communications must be added to the Comm-section, to correctly size the total communications cost. It is seen for Stampede that total communications contribution is significant at the largest ranges of parallelization, nevertheless it remains bounded under ~30%. Indeed, the strongly-distributed pattern exhibits a quite scalable total communications contribution, which explains the shape of the corresponding curve in Fig. 4 (on the contrary, the communication deterioration which happens in the strongly-concentrated case from 512 nodes on, justifies the observed rise of its execution time). The greater weight of communications, visible in Fig. 7 for Helios (~60%) and Eagle (over 90%), explains the quick drop of performance at the higher ranges of parallelization. A straightforward implication of such communication-intensive behavior is that task concentration over fewer nodes means shorter runtimes and a better scaling of the benchmark.

Out of this behavior observed in Helios and Eagle, one strategy to fight against the drop in computational efficiency (as the range of parallelization increases), is to enforce concentrated-task mappings. Doing so, the faster runtime will compensate a portion of the weak-scaling deterioration. As an example, the parallelization of 2197 tasks with the LAMMPS-Helios combination shows that applying the strongly-concentrated mapping leads to a ~52% speedup in runtime. It is visible in the plots that the slower execution due to the weak-scaling drop is the dominant effect in net performance loss, but the faster execution by applying a more efficient task-mapping, palliates this deterioration to some extent. Computational efficiency of the rhodopsin benchmark in several clusters has been already reported in the literature [37], but

this study includes its sensitivity to the task-mapping pattern for the first time, to the authors' knowledge.

Prior to introduce the results obtained with the code miniFE executed on the three supercomputers, some interesting results inferred from the executions addressed in [13] deserve attention. That work, which encompasses the mini-application miniFE among a bunch of solvers, is circumscribed to the context of sensitivity to system noise. The key aspect is to rebuild the executions information to focus on the sensitivity to task-mapping and to show evidence of the tight dependence between the execution time and the task-mapping pattern. Figure 8 shows the miniFE executed in the CAB cluster (LLNL, USA) [24] under almost ideal conditions (the cluster is dedicated and it counts with 1296 compute nodes, so highly distributed task-mappings can be achieved). The two curves in Fig. 8 show the weak-scaling behavior of miniFE in two opposite scenarios: 2 Processes (tasks) Per Node (in short, 2-PPN) and 16-PPN (this last corresponding to a strongly-concentrated pattern). At small range of parallelization both curves remain close, but for a large number of MPI-tasks, the 2-PPN case experiences a quick rise of the runtime, so a performance drop is linked to strongly-distributed patterns. For a 2048-task execution, a difference of more than 200% in runtime occurs and the tendency of the curves indicates that it would be even bigger at greater scales.

Besides it should be noted that miniFE is an application that weak-scales poorly and heavily relies on collective communications [13]. Thus, an adequate task-mapping must be identified for the best computational efficiency. Comparison of the executions of miniFE carried out on Stampede, Helios and Eagle (see Figs. 9, 10 and 11) shows that only Eagle exhibits similar behavior to the CAB cluster. On the contrary, Stampede and Helios show that strongly-distributed patterns provide higher computational efficiency. In Eagle, for a range of parallelization over 128 tasks, concentration of task over fewer nodes leads to an improvement. In particular, the 2048-task experiment shows that the concentrated pattern leads to 250% faster executions in average.

Analogous results are obtained with the mini-kernel bigFFT (see Figs. 12 and 13): the weak-scaling behavior differs in Stampede and Eagle at scale and as with the

miniFE kernel, it runs faster by mapping the jobs (of >256 tasks) in as few nodes as possible in Eagle supercomputer. Interestingly, tests in Stampede show that weak-scaling deteriorates very quickly when the range of parallelization goes over 512 tasks, as it is visible in Fig. 12. This local, abrupt rise in slope of the curves – quite close for the three mappings- may be explained by the inefficient management of such number of pencils in the FFTs, or even the participation of extra switch hops during runtime.

Executions of bigFFT in Eagle exhibit a steeper loss of computational efficiency over the range of parallelization (more evident for the strongly-distributed pattern), which peaks over 256 tasks. The strongly-distributed pattern overpasses the runtime of the strongly-concentrated pattern in more than ~140%. Sensitivity to the mapping is summarized in Table 6 for two scenarios: very small and large range of parallelization, say 16 and 2048 MPI-task jobs.

While the 16-task jobs scenario exhibits an increase of about 200% in the execution time when a strongly-concentrated pattern is applied (16-PPN), the opposite behavior happens for the 2048-task jobs, which saves more than half the computing time when the same strongly-concentrated pattern (16-PPN) is imposed. This runtime overhead at scale is mostly linked to the two major communications bottlenecks inherent to the pencil-based FFT algorithm, that is: heavy All-to-All communications and Point-to-Point calls, which dominate the computing cost.

A clarification must be done regarding the initial slope of the weak-scaling curves computed in the bigFFT-Stampede combination. The theoretical computational intensity of the 3D FFT is $O(N^3 \log(N))$. That is, weak-scaling requires, in a first approximation, to increase the core count 8 times with each two-fold increase in the grid side for each dimension. Or a core count of 2 times for one two-fold increase in one dimension of the 3D dataset, which is the strategy here adopted. It is clear that the factor log(N) of the scaling rule impedes an ideal weak-scaling and explains the non-flat slope shown in Stampede over the range 8 to 512 tasks.

The possibility of applying strongly-concentrated patterns to attain shorter runtimes in clusters within some ranges of the parameters involved, suggests a generalization in

terms of two major parameters, say the dataset size (domain, grid…), managed by each core under weak-scaling; and the range of parallelization. This would lead to a kind of characterization for each given application-cluster combination, as shown in Fig.14. This information results to be useful at both users and administrator levels, to improve the applications performance and throughput of a cluster.

7. CONCLUSIONS

Weak-scaling tests have been used to analyze the role of MPI communications on the performance of scientific applications and to identify execution bottlenecks. Three applications have been into focus: LAMMPS and two mini-kernels (miniFE and bigFFT). Experiments have been designed and run in three fat-tree topology-based supercomputers: Stampede at TACC (USA), Helios at IFERC (Japan) and Eagle at PSNC (Poland).

The results show that applications runtime strongly depends on the chosen task-mapping pattern, being the results non-homogeneous for the three clusters. Factors that drive this behavior are the cluster topology (rather different regarding the fat-tree levels, count of nodes connected to the switches or number of switches), scale of execution, network traffic leading to congestion, and load imbalance among others. This leads the application to behave in a particular manner in terms of performance.
In addition, it is difficult to maintain scalable sustained bandwidth on clusters as scale goes further because of the topology and network traffic. One plausible cause of the differences found in scalability and runtime, is the number of switch hops during execution: the larger the number of hops at execution, the higher the runtime variation across the tests. The observed effect, which the present investigation shows to occur at hundreds of nodes and in the range a couple of thousands of MPI-tasks for communication-intensive algorithms, is likely to be exacerbated in the coming exascale era.

Hence, optimal mappings at extreme scale deserve careful attention to attain better computational efficiency in the runs of communication-intensive applications. In this

context, topology-aware mapping of tasks is an idea to explore in more detail, to seek for the cause of performance drop at scale. Hence, tools for sophisticated job placing and task mapping, tailored at particular cluster topologies, would be an asset to improve the computational efficiency. In particular, cluster throughput could benefit from the implementation of task live-migration in the scheduling policies, which will help to clarify if it is worth or not to concentrate the tasks on as few nodes as possible. In either case, the conducted experiments are useful to assess the correctness of scalability simulators at predicting performance drop or cluster efficiency in these types of executions. Further investigation must address a better quantification of the compute-to-communication ratio with profilers, as well as the weight of the all-to-all and point-to-point communications calls contributions.

Last but not least, LAMMPS is an example of a full-fledged application in comparison to the kernels miniFE and bigFFT. These last are very informative since they, besides being building blocks of larger, production codes, serve to guide hardware and software components decisions. However, the behavior observed in the mini-applications should be taken cautiously and not to extrapolate it to a production code which incorporate the mentioned algorithm as part of its design.

ACKNOWLEGMENT

This work was partially funded by the Spanish Ministry of Economy and Competitiveness CODEC-OSE project (RTI2018-096006-B-I00) and the Comunidad de Madrid CABAHLA project (S2018/TCS-4423), both with European Regional Development Funds (ERDF). It also profited from funds from H2020 co-funded projects Multiscale Modelling for Fusion and Fission Materials (M4F, No. 755039), Energy oriented Centre of Excellence for computing applications II (EoCoE-II, No. 824158), and Supercomputing and Energy in Mexico (Enerxico, No. 828947). Access to resources of CYTED Network RICAP (517RT0529) and Poznan Supercomputing and Networking Center, in particular the support of Marcin Pospieszny, system administrator at PSNC, is acknowledged.

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), pp. 22-30 (2011). https://doi.org/10.1109/MC.2011.300

[3] Exascale Computing Project (ECP) homepage, https://www.exascaleproject.org

[4] EuroHPC homepage, 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] Moríñigo J.A., García-Muller P., Rubio-Montero A.J., Gómez-Iglesias A., Meyer N., Mayo-García R.: Benchmarking LAMMPS: Sensitivity to Task Location Under CPU-Based Weak-scaling. In:.High Performance Computing, Proc. 5th Latin American Conference (CARLA 2018), Bucaramanga, Colombia – Comm. In Computer and Information Sci., vol. 979, pp. 224-238 (2019). https://doi.org/10.1007/978-3-030-16205-4_17

[9] Jeannot E., Mercier G., Tessier F.: Process Placement in Multicore Clusters: Algorithmic Issues and Practical Techniques. IEEE Transactions on Parallel and Distributed Systems 25(4), pp. 993-1002 (2014). https://doi.org/10.1109/TPDS.2013.104

[10] Chavarría-Miranda D., Nieplocha J., Tipparaju V.: Topology-aware Tile Mapping for Clusters of SMPs. In: Proc. 3rd Conference on Computing Frontiers, Ischia, Italy (2006). https://doi.org/10.1145/1128022.1128073

[11] Smith B., Bode B.: Performance Effects of Node Mappings on the IBM BlueGene Machine. In: Euro-Par 2005 Parallel Processing. Lecture Notes in Computer Science, vol. 3648, pp. 1005-1013, (2005). https://doi.org/10.1007/11549468_110

[12] 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: Proc. IEEE Symposium Computers and Comm., pp. 811-817, Sousse, Tunisia (2009). https://doi.org/10.1109/ISCC.2009.5202271

[13] León E.A., Karlin I., Moody A.T.: System Noise Revisited: Enabling Application Scalability and Reproducibility with SMT. In: Proc. IEEE International Parallel and Distributed Processing Symposium, pp. 596-607, Chicago, USA (2016). https://doi.org/10.1109/IPDPS.2016.48

[14] 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

[15] 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. Conf. Parallel and Distributed Computing and Systems, pp. 455-460 (2011). https://doi.org/10.2316/P.2011.757-005

[16] 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: Compute The Future, Proc. Cray User Group (CUG 2009), Atlanta, USA (2009).

[17] Rodríguez-Pascual M., Moríñigo J.A., Mayo-García R.: Effect of MPI Tasks Location on Cluster Throughput Using NAS. Cluster Computing 22(4), pp. 1187–1198 (2019). https://doi.org/10.1007/s10586-018-02898-7

[18] 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

[19] Xingfu W., Taylor V.: Processor Partitioning: An Experimental Performance Analysis of Parallel Applications on SMP Clusters Systems. In: 19th IASTED Conference Parallel Distributed Computing and Systems (PDCS07), pp. 13-18, Cambridge, USA (2007).

[20] Zhang C., Yuan X.: 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, University of Edinburgh, EPCC (2007)

[22] Liu J.: LAMMPS on Advanced SGI Architectures. White Paper SGI (2010)

[23] León E.A., Rosenthal E.: Characterizing Applications Sensitivity to Network Performance. In: Supercomputing Conference (SC'14), Poster. New Orleans, USA (2014).

[24] León E.A., Karlin I., Bhatele A., Langer S.H., Chambreau C., Howell L.H., D'Hooge T., Leininger M.L.: Characterizing Parallel Scientific Applications on Commodity Clusters: An Empirical Study of a Tapered Fat-Tree. In: Proc. International Conference for High Performance Computing, Networking, Storage and Analysis (SC'16), Salt Lake City, USA (2016)

[25] Jain N., Bhatele A., Howell L.H. Böhme D., Karlin I., León E.A., Mubarak M., Wolfe N., Gamblin T., Leininger M.L.: Predicting the Performance Impact of Different Fat-Tree Configurations. In: Proc. International Conference for High Performance Computing, Networking, Storage and Analysis (SC'17), pp. 50:1-50:13, Denver, USA (2017). https://doi.org/10.1145/3126908.3126967

[26] Choi D.J., Lockwood G., Sinkovits R.S., Tatineni M.: Performance of Applications Using Dual-Rail InfiniBand 3D Torus Network on the Gordon Supercomputer. In: Conference on Extreme Science and Engineering Discovery Environment (XSEDE'14), pp. 43:1-43:6, Atlanta, GA, USA (2014). https://doi.org/10.1145/2616498.2616541

[27] 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)

[28] Ferreira K., Grant R.E., Levenhagen M.J., Levy S., Groves T.: Hardware MPI Message Matching Behaviour to Inform Design, Corrurrency and Computation, Practice and Experience (2019). https://doi.org/10.1002/cpe.5150

[29] Pollard S.A., Jain N., Herbein S., Bhatele A.: Evaluation of an Interference-free Node Allocation Policy on Fat-tree Clusters, in Proceedings of the ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis, (SC'18), Dallas, USA (2018). https://doi.org/10.1109/SC.2018.00029

[30] León E.A., Chambreau C., Leininger M.L.: What Do Scientific Applications Need? An Empirical Study of Multirail Network Bandwidth. In: 7th Int. Conference on Advanced Communications and Computations (INFOCOMP 2017), pp. 35-39, Venice, Italy (2017)

[31] Dang H.V., Snir M., Gropp W.: Towards Millions of Communicating Threads, in Proceedings of the 23nd European MPI Users' Group Meeting (EuroMPI 2016), pp. 1-14, Edinburgh, UK (2016). https://doi.org/10.1145/2966884.2966914

[32] Radulovic M., Asifuzzaman K., Carpenter P., Radojkovic P., Ayguadé E.: HPC Benchmarking: Scaling Right and Looking Beyond the Average, in Proceedings of the 24th International European Conference on Parallel and Distributed Computing (EuroPAR 2018), LNCS vol. 11014, pp.135-146 (2018). https://doi.org/10.1007/978-3-319-96983-1_10

[33] Stampede supercomputer: https://www.tacc.utexas.edu/systems/stampede

[34] Helios supercomputer: http://www.iferc.org/CSC_Scope.html#Systems

[35] Eagle supercomputer: https://wiki.man.poznan.pl/hpc/index.php?title=Eagle

[36] Taffet P., Rao S., León E.A., Karlin I. Testing the Limits of Tapered Fat Tree Networks. In: IEEE/ACM Performance Modeling, Benchmarking and Simulation of High Performance Computer Systems (PMBS), pp.47-52, Denver, USA (2019). https://doi.org/10.1109/PMBS49563.2019.00011

[37] LAMMPS homepage, https://lammps.sandia.gov/bench.html#rhodo

[38] Brooks B.R., Brooks III C.L., Mackerell Jr. A.D., Nilsson L., Petrella R.J., et al.: CHARMM: The Biomolecular Simulation Program. Journal of Computational Chemistry, 30 (10). pp. 1545-1614 (2009). https://doi.org/10.1002/jcc.21287

[39] 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

[40] Brown W.M., Kohlmeyer A., Plimpton S.J., Tharrington A.N.: Implementing Molecular Dynamics on Hybrid High Performance Computers - Particle-Particle Particle-Mesh, Comp. Phys. Comm. 183(3), pp.449-459 (2012). https://doi.org/10.1016/j.cpc.2011.10.012

[41] Lin P.T., Heroux M.A., Barrett R.F., Williams A.B.: Assessing a mini-application as performance proxy for a finite element method engineering application. Concurrency and Computation, 27 (17). pp. 5374-5389 (2015). https://doi.org/10.1002/cpe.3587

[42] Richards D.F., Glosli J.N., Chan B. Dorr M.R., Draeger E.W. et al.: Beyond homogeneous decomposition: scaling long-range forces on Massively Parallel Systems. In: Proc. International Conference on High Performance Computing Networking, Storage and Analysis (SC'09), art. nº 60. Portland, USA (2009). https://doi.org/10.1145/1654059.1654121

[43] Fast Fourier Transform of the West homepage, http://www.fftw.org

[44] Plimpton S., Pollock R., Stevens M.: Particle-Mesh Ewald and rRESPA for Parallel Molecular Dynamics Simulations. In: SIAM 8th Conference on Parallel Processing for Scientific Computing (1997).

[45] Bhatia H., Jain N., Bhatele A., Livnat Y., Domke J., Pascucci, V., Bremer P.: Interactive Investigation of Traffic Congestion on Fat-Tree Networks Using TreeScope, Computer Graphics Forum (37) pp. 561-572 (2018). https://doi.org/10.1111/cgf.13442

[46] Qiao P., Wang X., Yang X., Fan Y., Lan Z.: Preliminary Interference Study About Job Placement and Routing Algorithms in the Fat-Tree Topology for HPC Applications. In: IEEE International Conference on Cluster Computing (CLUSTER), pp. 641-642, Honolulu, USA, (2017). https://doi.org/10.1109/CLUSTER.2017.90

LIST OF FIGURES

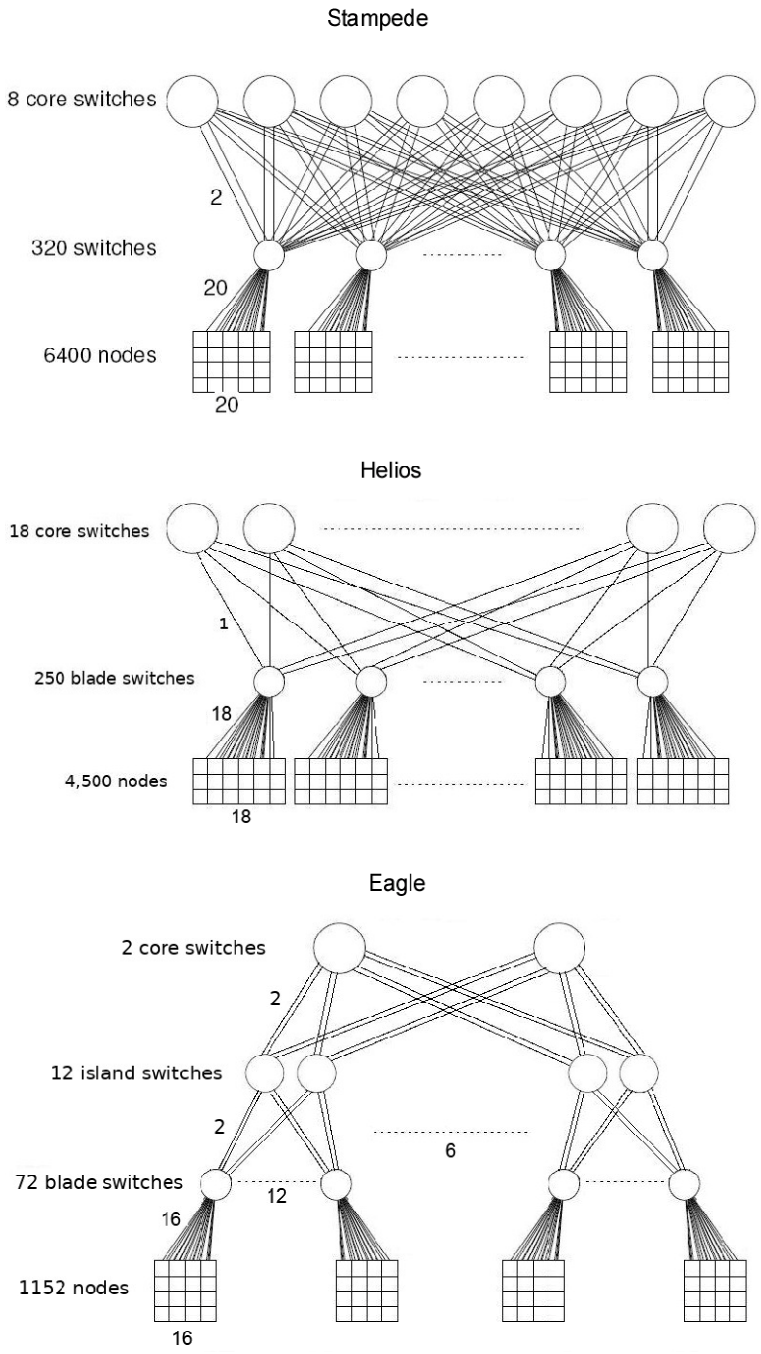

Fig. 1 Fat-tree network of supercomputers Stampede (up) at TACC (USA) [27]; Helios (middle) at IFERC (Japan) [34]; and Eagle (bottom) at PSNC (Poland) [35].

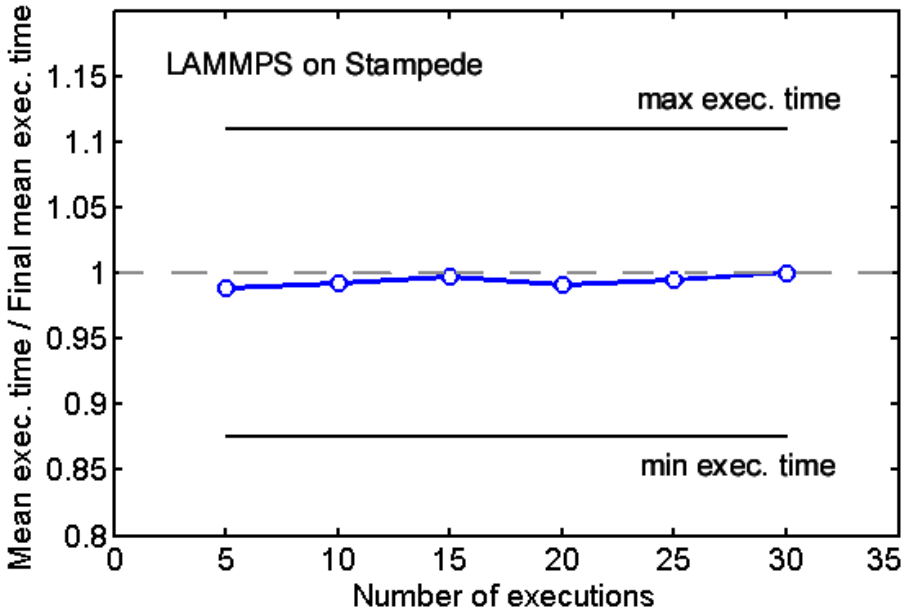

Fig. 2 Mean execution time over the number of executions of LAMMPS in Stampede. Execution time is non-dimensionalized with the final mean execution time corresponding to 30 realizations. Maximum and minimum non-dimensional execution time boundaries are plotted.

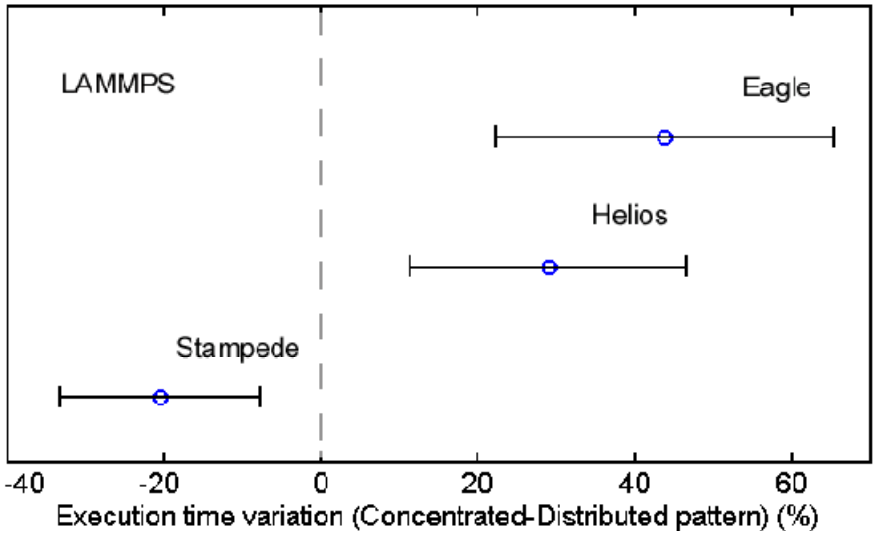

Fig. 3 Variation of the mean execution time when a concentrated pattern is mapped instead of a distributed pattern for LAMMPS benchmark in the three clusters: Stampede, Helios and Eagle.

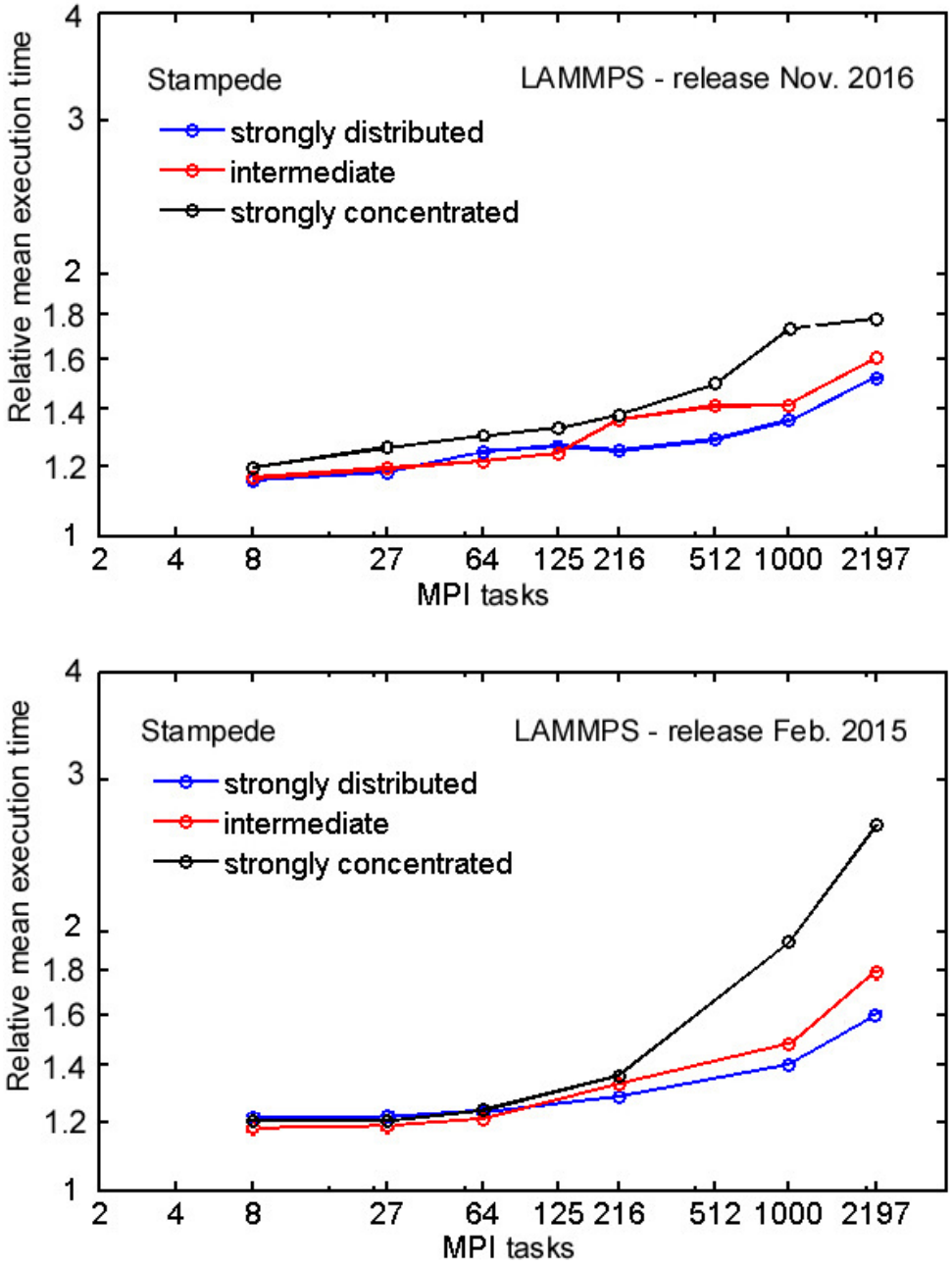

Fig. 4 Relative mean execution time of LAMMPS in Stampede corresponding to MPI ranges: 8, 27, 64, 125, 216, 512, 1000 and 2197. The MPI processes have been allocated over the nodes with three different mapping patterns (see legend). Execution time is compared for two versions of the code: release of November 2016 (up) and release of February 2015 (bottom). Reference execution time: 500sec.

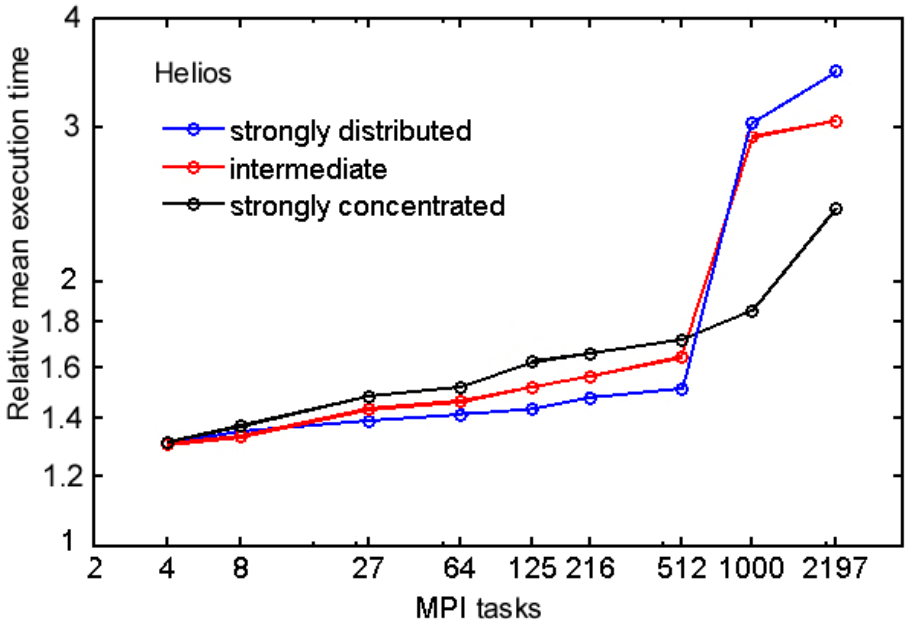

Fig. 5 Relative mean execution time of LAMMPS in Helios, corresponding to the MPI ranges: 4, 8, 27, 64, 125, 216, 512, 1000 and 2197. The MPI processes have been allocated over the nodes with three mapping patterns (see legend). Reference execution time: 500sec.

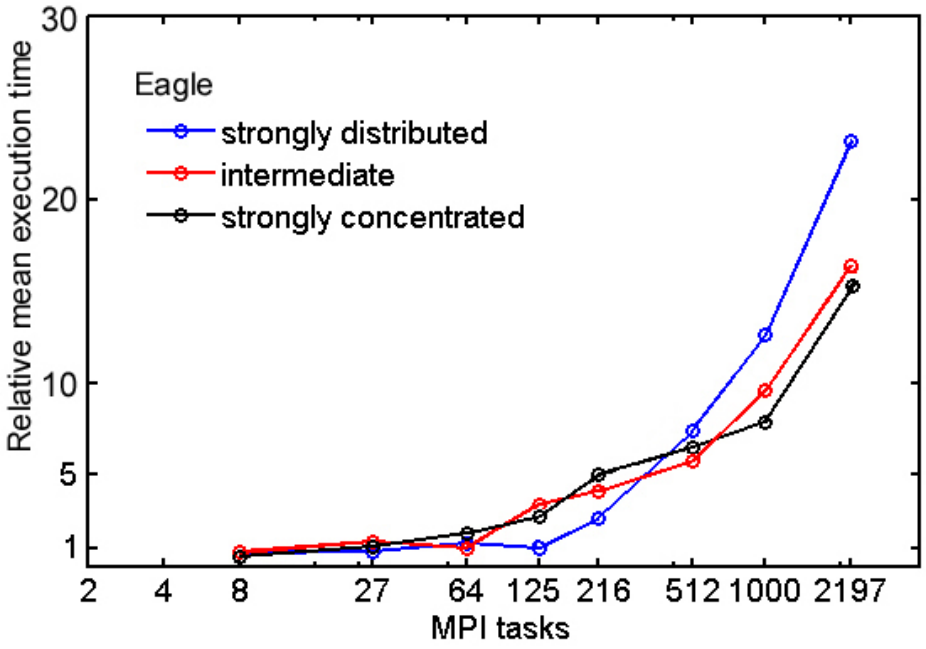

Fig. 6 Relative mean execution time of LAMMPS In Eagle, corresponding to MPI ranges: 8, 27, 64, 125, 216, 512, 1000 and 2197. The MPI processes have been allocated over the nodes with three mapping patterns (see legend). Reference execution time: 500sec.

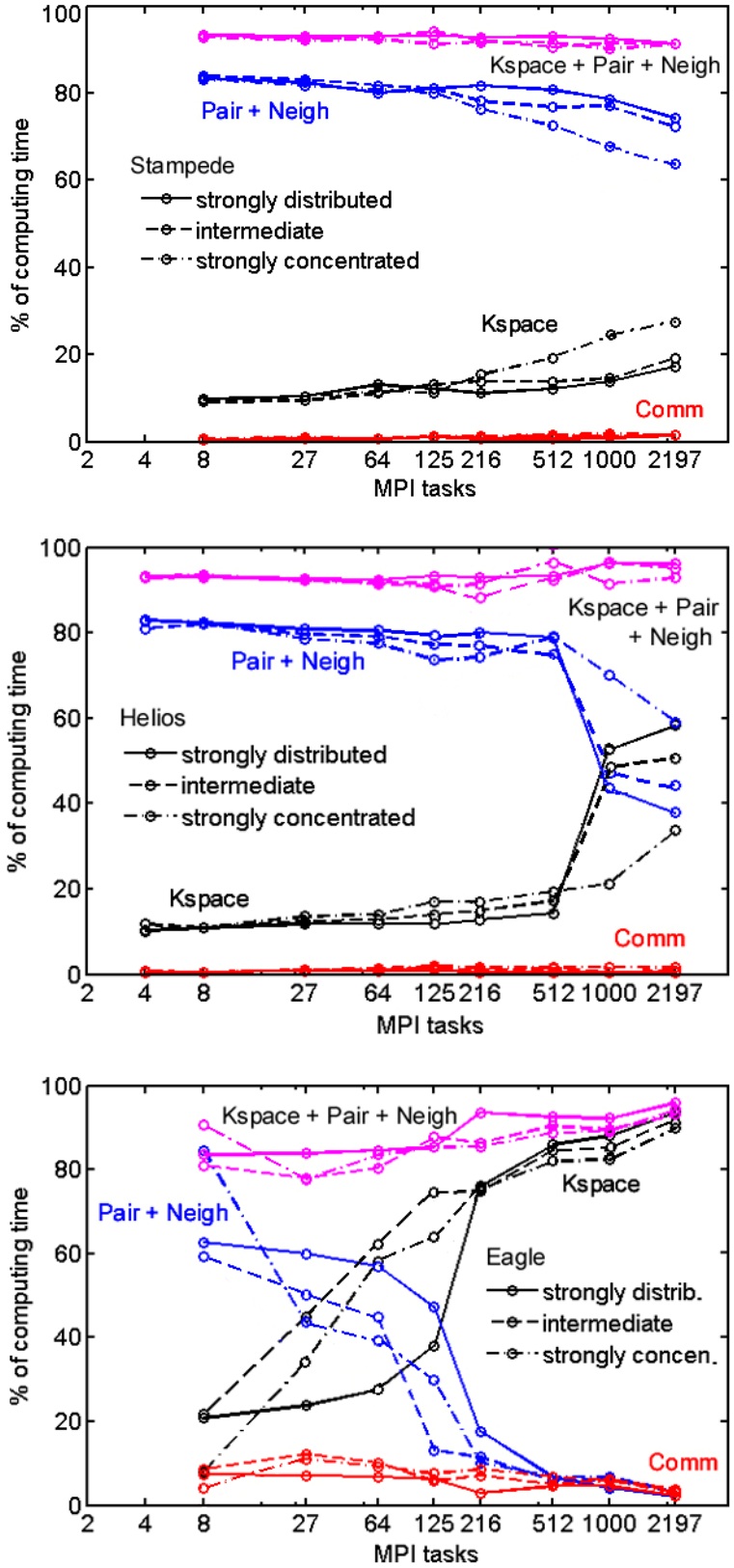

Fig. 7 Relative mean contributions of the major algorithmic sections (Pair, Neigh, KSpace and Comm) to the execution time of the LAMMPS benchmark in the 3 clusters.

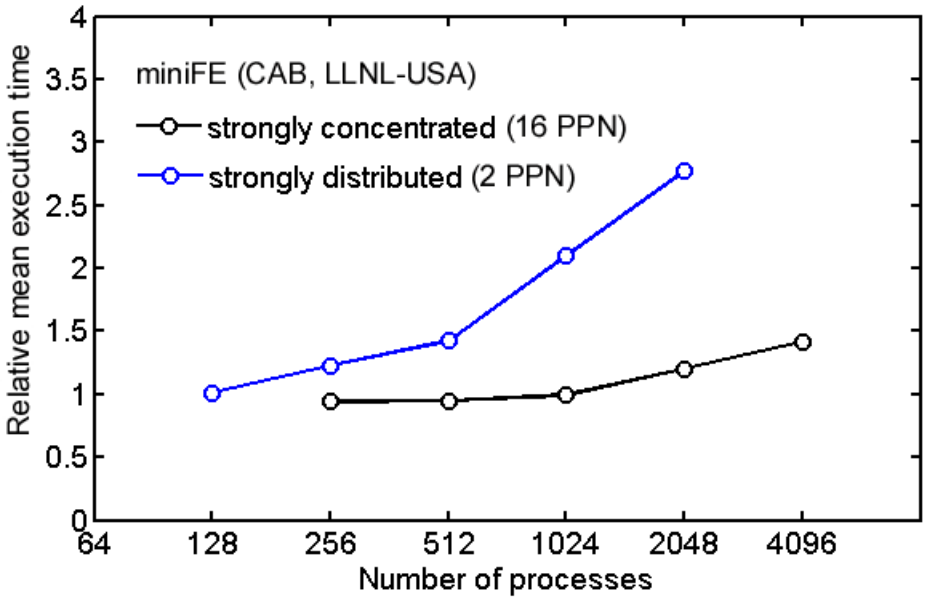

Fig. 8 Relative mean execution time of mini-application miniFE in cluster CAB at LLNL (USA) corresponding to two process (MPI-tasks) mappings: concentrated pattern (of 16 process-per-node: 16 PPN); and distributed pattern (2 PPN). (Data post-processed from [13]. Reference execution time: 25sec).

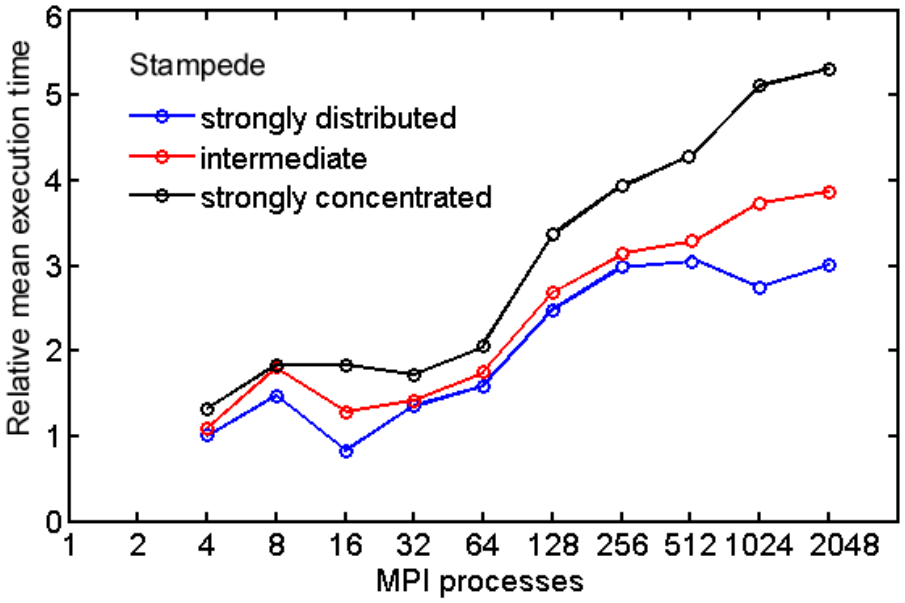

Fig. 9 Relative mean execution time of miniFE in Stampede, corresponding to MPI ranges: 4, 8, 16, 32, 64, 128, 256, 512, 1024 and 2048. The MPI processes have been allocated over the nodes with three mapping patterns (see legend). Reference execution time: 84sec.

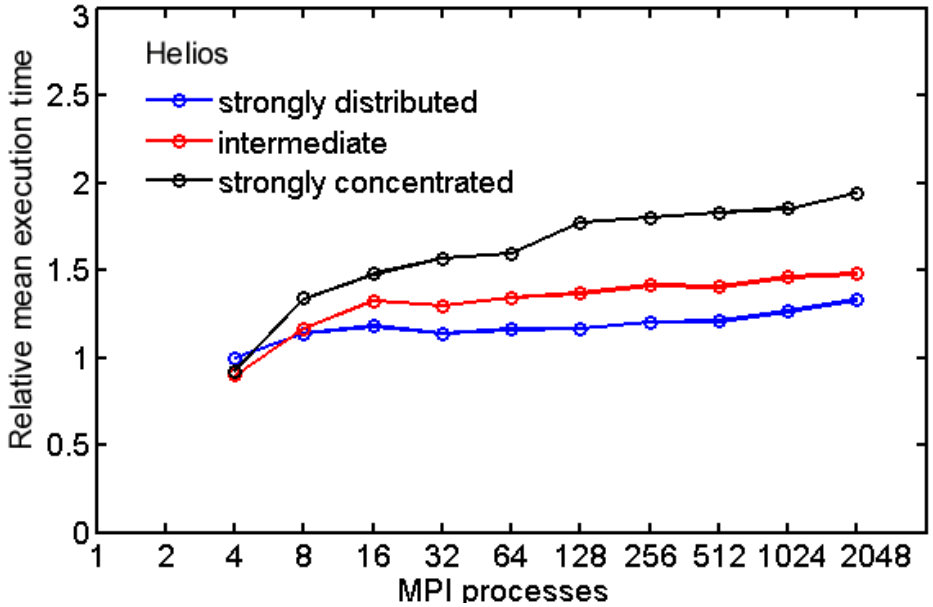

Fig. 10 Relative mean execution time of miniFE in Helios, corresponding to MPI ranges: 4, 8, 16, 32, 64, 128, 256, 512, 1024 and 2048. The MPI processes have been allocated over the nodes with three mapping patterns (see legend). Reference execution time: 343sec

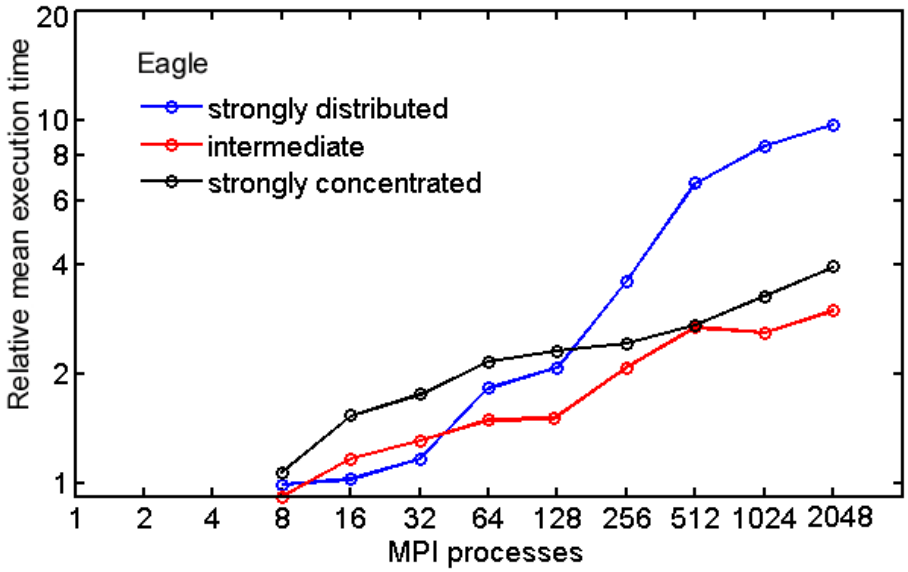

Fig. 11 Relative mean execution time of miniFE in Eagle, corresponding to MPI ranges: 8, 16, 32, 64, 128 256, 512, 1024 and 2048. The MPI processes have been allocated over the nodes with three mapping patterns (see legend). Reference execution time: 266sec.

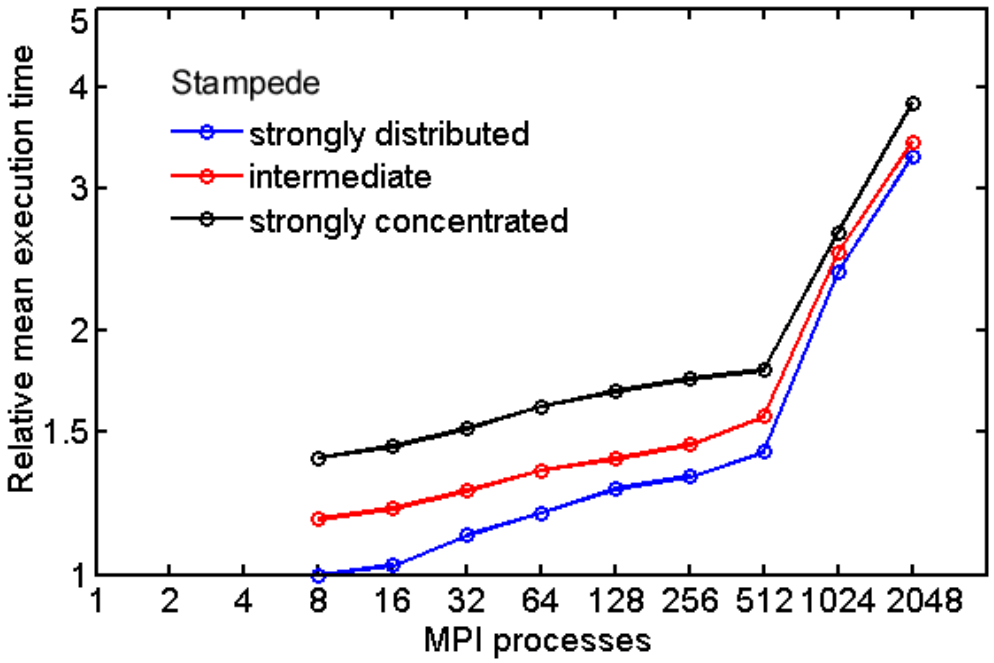

Fig. 12 Relative mean execution time of bigFFT in Stampede, corresponding to MPI ranges: 8, 16, 32, 64, 128 256, 512, 1024 and 2048. The MPI processes have been allocated over the nodes with three mapping patterns (see legend). Reference execution time: 289sec.

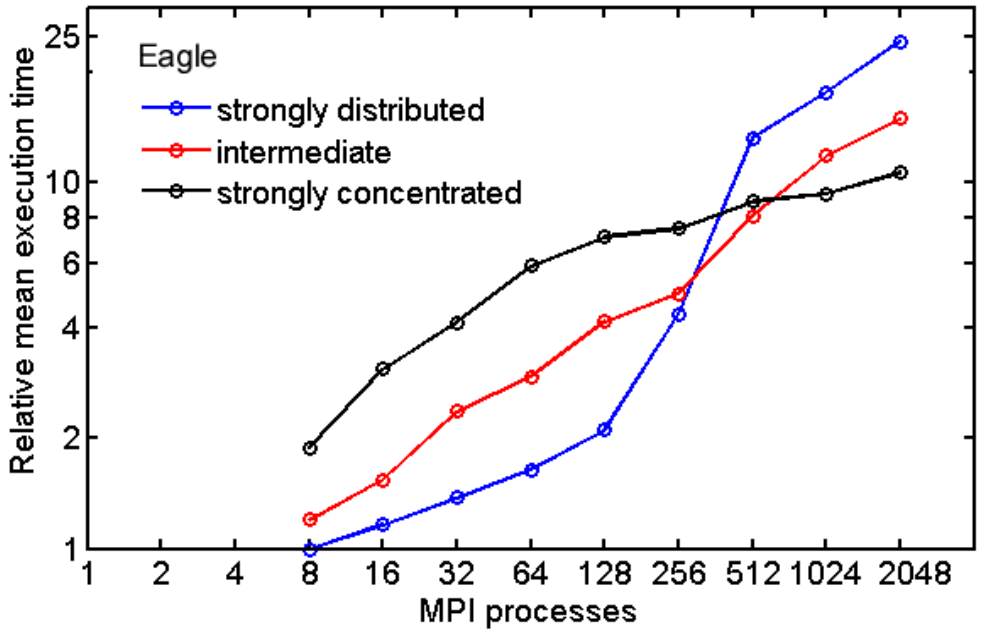

Fig. 13 Relative mean execution time of bigFFT in Eagle, corresponding to MPI ranges: 8, 16, 32, 64, 128 256, 512, 1024 and 2048. The MPI processes have been allocated over the nodes with three mapping patterns (see legend). Reference execution time: 133sec.

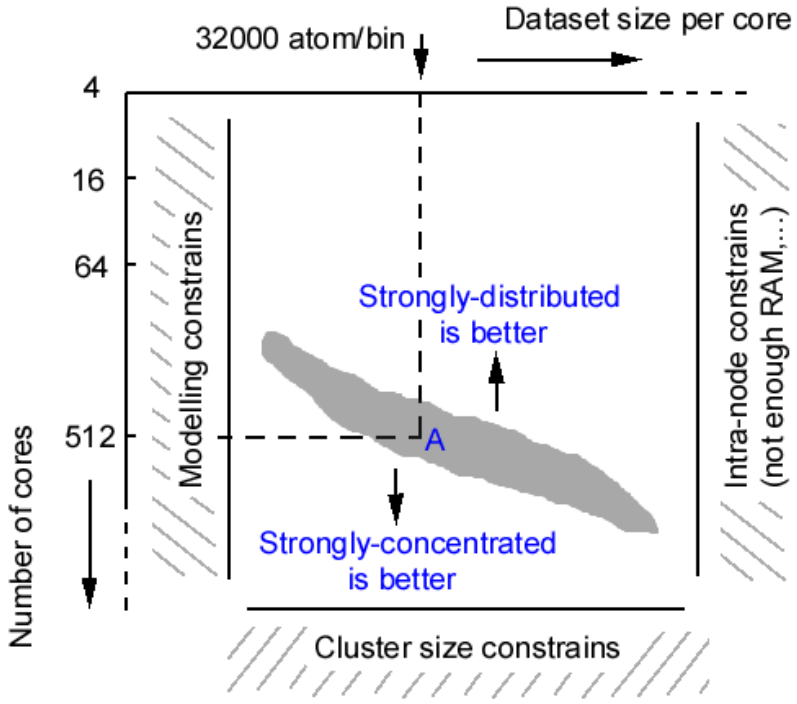

Fig. 14 Schematic characterization map of a generic combination Application-Cluster under weak-scaling. Task-mapping based on strongly concentrated or distributed patterns for performance improvement is indicated (point A is set for illustration purposes, it corresponds to the LAMMPS-Helios case tested).