Multi-phased task placement of HPC applications in the Cloud

Emmanuell D. Carreno, Marco A. Z. Alves
UFPR - Brazil
{edcarreno, mazalves}@inf.ufpr.br

Matthias Diener
UIUC - United States
mdiener@illinois.edu

Eduardo Roloff, Philippe A. O. Navaux
UFRGS - Brazil
{eroloff,navaux}@inf.ufrgs.br

***Abstract*—Many high-performance computing applications present different phases during their execution. Nevertheless, thread and process placement techniques usually provide static-only methods to improve the data and thread locality. Similarly, cloud computing datacenters may present variations in terms of latency over the execution time of applications. To overcome these two problems, in this paper we analyze scientific applications that have different communication patterns along with its execution. For such applications, we evaluate the performance variation of traditional static placement techniques to our new approach that uses code annotations to perform the new placement of tasks, matching also the variations on network performance of Virtual Machines (VMs) during the run time. For our experiments, we use applications from the NAS parallel benchmark suite, running them on two VM sizes with 32 and 64 cores respectively, from the same family of instance types at the West US datacenter from Azure. Results show that compared to traditional static process mapping, our multi-phased placement mechanism achieves average performance gains of 13.57%, up to 28.32% in the evaluated scenarios. These results show that there is an opportunity to improve performance by correctly identifying the network variations and reacting by generating a new task-to-instance mapping.**

***Index Terms*—Cloud Computing, HPC, Task Mapping, MPI, NAS, Network Variability**

The advent of cloud computing allowed several new use cases for computing on demand. Storing, processing and transferring data became a utility charged by the number of resources consumed. For this reason, performance improvement techniques are critical to reducing costs.

The growth of cloud computing in the last years is creating new opportunities for different computing market segments [1]. One of those segments is the use of cloud computing for High-Performance Computing (HPC), moving from in-house data centers and clusters with homogeneous hardware and interconnections to a highly dynamic and shared environment. The users of cloud services have very little information about the hardware in which their virtual machines are allocated and have almost no information about the network topology and interconnection technology [2]. The amount of resources shared inside the physical machine or its level of utilization is also unknown. Other users' interference may impact applications from a tenant due to multiple users sharing some resources, causing contention for this resources on the same physical machine, and in some cases degrading network performance or impacting the availability of allocation for new virtual machine instances.

During cloud applications execution, the users cannot access the allocation information, counters, and metrics available on the VM; low-level metrics of the bare machine are used only by the provider to maintain the QoS offered in the contract or service level agreement (SLA). However, for some users, the available VM information may be not enough for their performance improvement mechanisms [3], [4], and until users gain access to dedicated performance counters, the best option for any tenant is to use any measurable metric about the underlying hardware or technology used by the provider.

In this context, previous work [5] implemented a mechanism to improve the communication performance of parallel scientific applications. This previous work measured the network performance at the data center in which the VMs are allocated and feed this information into a task mapping mechanism to reduce the benchmark's time to solution. The main difference to the previous work is the multi-phase analysis, the first step towards an online network-performance aware task placement solution.

However, by considering only the networking variations in the cloud datacenter, may not be enough to achieve the maximum performance for HPC applications. One reason is that HPC applications often present multiple phases during its execution. Each phase the application may present a different behavior in terms of resources usage and communication pattern. Thus, a static task mapping may lack opportunities in terms of performing the best task-to-instance mapping.

In this paper, we use information about the current network status to improve the communication speed of parallel applications. We discuss how gains obtained by a task mapping mechanism are affected by the metric selected to measure the network status. We also show how changes in the application communication behavior can degrade performance caused by the lack of adaptability of current task mapping approaches. The main contributions of this paper are three-fold:

- An analysis of which NAS benchmarks benefit more from different networking metrics used to perform task mapping on a public cloud service.
- A quantitative comparison of the impact of using the incorrect task mapping on the distinct computing phases of a synthetic scientific application.

- An approximated measurement of the performance gap on the execution of multi-phased applications.

Results show that we could improve the previous work approach by 13.57% when we consider the distinct application phases in the task mapping mechanism.

I. Motivation and Proposal

In order to improve the performance of scientific applications using its inherent communication network characteristics, it is necessary to correctly identify its underlying patterns and the current state of the cloud network. Measuring and correctly using such information by task mapping techniques could help reduce the impact of a sluggish network.

A task mapping algorithm may use the information available from two sources of heterogeneity affecting communication performance. The first source comes from the heterogeneity in hardware topology in the cloud network that causes that some elements of the environment to communicate faster than others. The second source comes from the communication behavior heterogeneity from each application, which causes that some pairs of tasks perform more communication between them compared to other processing tasks. However, in the cloud, not every hardware aspect is known to the tenant; for this reason, it is possible to retrieve only partial networking information.

A. Network Variability

A cloud data center may have several sources of heterogeneity notably by hardware differences and by application load from other tenants, creating variations on the network performance over time. Other factors of variability may include sudden access peaks, daily usage cycles, and load-balancers [6]. Because all of these factors depend on what other tenants deployed on the same data center do, we (as tenants too) are unable to determine when they are going to happen. To illustrate the behavior mentioned above, we profiled the network variations over bandwidth and latency over 23h in the network performance of 4 cloud instances at the same data center.

Figure 1 shows the measurements for two network metrics, bandwidth, and latency on a pair of instances. The communication measurement was performed VM to VM using MPIBench, an open source benchmarking tool [7]. The experiment shows that along a day the performance of the two metrics is variable and there are differences between instances, that is, we can see that at the start of the experiment the bandwidth performance of the pair VM01-VM02 was improving slowly while simultaneously the same metric was degrading quickly for the pair VM02-VM04. This behavior of heterogeneous networks is recurrent on public clouds [8], [9], [5].

B. Cloud Mapping

To improve the performance of applications under such variable network conditions, some techniques were proposed in the past [10], [11], [12], [13]. Using a mechanism developed

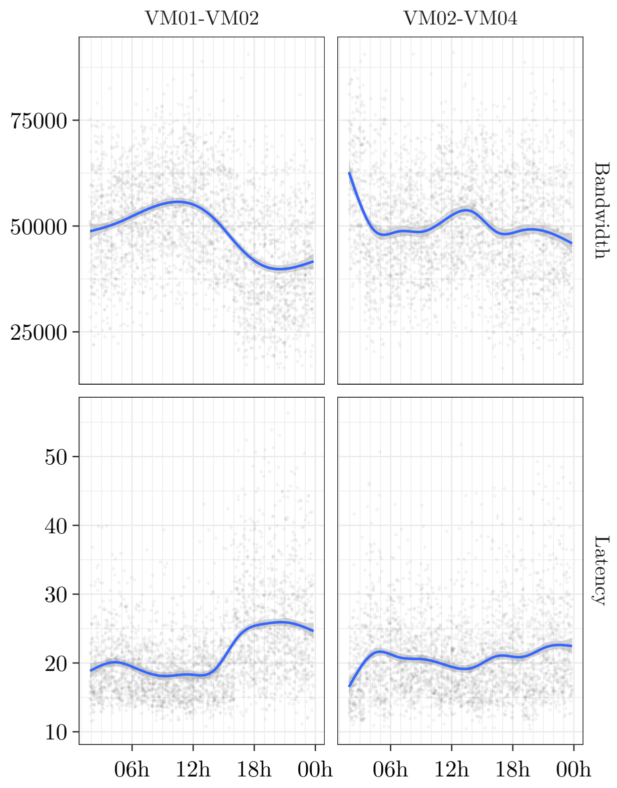


Fig. 1: Network variability of latency [ms] and bandwidth [Kbps] over 23 hours from two pairs of cloud instances, VM01 with VM02 and VM02 with VM04, the blue line shows an approximation of their behavior.

in a previous work [5], we performed experiments to motivate our proposal. This mechanism allows statically mapping processes with higher communication demands to nodes with better network performance.

For the analysis of the cloud mapping performance, we used a benchmark that simulates a Computational Fluid Dynamics (CFD) application. This benchmark (SP - Scalar Pentadiagonal solver) is part of the NAS parallel benchmark suite, which was developed by NASA to test computational characteristics of High-Performance Computing (HPC) clusters [14].

Figure 2 shows the average results of executing the SP benchmarks using the two base metrics for the tool, bandwidth, and latency, compared with running the benchmark without using any information about the current network performance.

Extracting information about the current state of the network allows improving the performance of an application [5]. Even with the limited information available for the tenant, limited compared with the bare metal information that the cloud provider has access to, it is possible to implement mechanisms to profile such state and improve upon that. However, this technique does not take into account changes on communication behavior during execution.

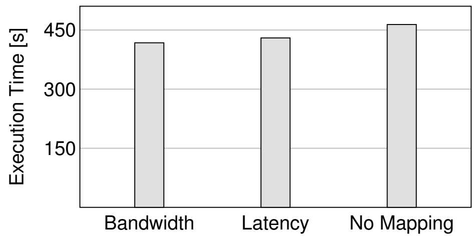


Fig. 2: Performance results using different mappings using 256 tasks and NAS size C.

This experiment also indicates that there are no significant improvements on network variability that may come from changes in infrastructure or updated datacenter hardware after three years from the initial release of CloudMap.

C. Application phases

Various HPC applications, such as Poisson equation solver, relies on various algorithms during its execution. In such cases, a static placement technique waste opportunities because it will be unable to obtain the maximum performance from the matching between application communications pattern and network metrics, as it varies along the time.

To simulate the behavior of a multi-phased application, an application needs to change its communication pattern along its execution time. Figure 3 shows the communication matrices for three different scenarios with 64 processes, where darker color represent points where more communication happened between two processing tasks. The first matrix on Figure 3a present the communication between processes over the complete application execution, calculated as the matrix addition of the communication matrices of both benchmarks. Thus, considering a theoretical application performing CFD simulation with two phases. Figures 3b and 3c present the distinct behavior for each phase of such application.

In this section, we showed that network variability is still an issue that needs to be addressed. One way of mitigating its impact may be by using the available metrics improving communication performance. Nonetheless, regarding the application executed we were looking for improvements by analyzing which best metric suites better each type of application and, also by examining what happens when the underlying communication pattern changes.

II. Methodology

In this section, we will detail each part of our proposal and explain their duty on the solution.

A. NAS benchmarks

To evaluate the impact of a proposal on different applications we ran our experiments using the MPI implementation of the NAS parallel benchmark suite [14], version 3.3.1. Each one of the benchmark applications was developed to represent specific computational and data movement aspects encountered on scientific applications [15].

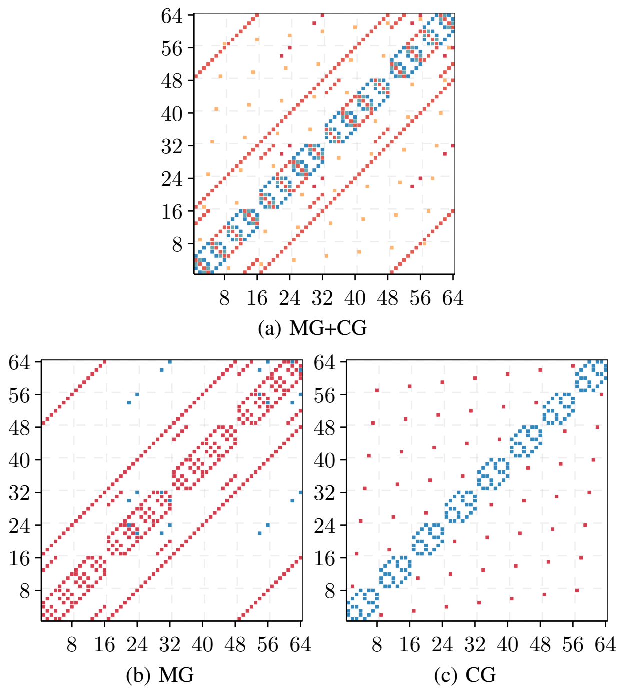


Fig. 3: Communication matrices for MG and CG kernels from the NAS benchmarks. (a) shows the communication matrix for an application with two different processing phases. (b) and (c) are the two phases that compose (a).

Regarding the communication of each benchmark, they can be divided into three categories using the type of messages interchanged between their processes. BT, CG, LU, MG, and SP benchmarks use mostly point-to-point messages. FT and IS use mostly collective messages to communicate between their processes. EP performs communication between its tasks only at the end to compute the final result. These eight benchmarks were executed for all the experiments.

To obtain the communication matrix for each NAS benchmark we used EZTrace [16], which automatically instruments MPI-based applications. We created the communication matrices using two metrics, the number of interchanged messages between tasks and the volume of such messages. This step was performed offline, and its result was reused for every execution of the same benchmark.

B. Mapping strategies

The evaluation was conducted using seven different mapping strategies. The first three use a reorganization of the rank allocation without any additional information about the network conditions:

default: The default mapping allocation performed by OpenMPI uses a round-robin fashion, where every task is allocated on the first available node until all slots available are full in that node and then goes to the next node available.

random: The algorithm generates a rank file that allocates all the tasks on any of the available nodes without any defined priority. This mapping strategy also helps in finding any abnormality on the other allocation proposals.

interleave: The tasks are allocated in order by node, that means the first node in the order defined by the MPI framework receives the first for allocation on it, then, the second node receives the second task, and so on until all task are allocated on all the available slots. Sometimes also called *Cyclic*.

The next four mapping strategies use CloudMap with the actual measurements of the current network performance using one of two different sources of variability, latency (*lat*) or bandwidth (*bw*). For each source of variability, we used the values from the two metrics of the profiled communication patterns available for each benchmark, the number of messages (*num*) or volume of data interchanged between tasks (*vol*), this allows us to test the followings strategies:

bw-num: Uses bandwidth and number of messages.

bw-vol: Uses bandwidth and volume of messages.

lat-num: Uses latency and number of messages.

lat-vol: Uses latency and volume of messages.

C. Deployments

Previous work [5] showed that network variability also depends on particular aspects of the data center in which the VM instance is allocated. To reduce this additional source of variability, that regional usage aspects of every data center location may introduce, we evaluated our proposal at only one data center located on the west of the United States (West US Region).

The VMs selected are part of the most up-to-date offered by the cloud provider. We used two different instance sizes from the same performance series shown in Table I. Benchscore is a metric to compare the performance of VMs. The metric is the resulting value of a resource-intensive benchmark provided by Microsoft for each one of their VM sizes [17]. The selected VM sizes have a good balance of memory, CPU performance and networking specifications, suitable for the requirements of the NAS benchmarks. We will use a short version of the name for every VM instance in this paper. The specifications from the cloud provider indicate that D64v3 is, in essence, a virtual machine with twice the computing power and network bandwidth than D32v3.

The process of instantiating and deallocating the VMs was performed using in-house developed scripts that allowed reproducibility of the experiments. We used the Linux distribution Ubuntu 16.04 LTS, Kernel version 4.15.0-1037 and OpenMPI version 1.10.2. Every experiment was initiated and controlled using a central VM instance that did not take part in the processing an also acted as a cloud front-end to allow monitoring the status of the experiment without creating interference.

Network communication between VMs is performed using flat network topology. In total, we used 256 CPU cores for computing on each deployment, eight instances of 32 cores for D32v3 and four instances of 64 cores for D64v3.

TABLE I: Main characteristics of the Cloud Instances used.

VM Size	VMs	Benchscore	Cores/VM	Network speed
Standard_D32_v3	8	309021	32	16000 Mbps
Standard_D64_v3	4	613424	64	30000 Mbps

D. Mechanism

To perform the actual task mapping we used CloudMap [5]. CloudMap is a tool that allows analyzing the current state of network performance using both bandwidth and latency between all the user's allocated VMs. The information gathered by the tool creates a network performance matrix, using this information and a previously obtained application communication pattern can generate an MPI rank file. Using this approach, we can create groups of tasks that communicate more into the VMs that have better network performance between them. The mechanism was modified to receive a specific static application communication matrix at runtime, the profiled number or volume of messages. Current state measuring is performed between all nodes, in an all to all fashion. A drawback for the scalability of this approach is that it is limited by the network bandwidth and by the number of nodes in the deployment. The results obtained by combining this four-part methodology are discussed in the next section.

III. Results

First, we will show the results of executing each NAS benchmark using the seven task mapping strategies. Then, we will discuss the impact on applications phases performing task mapping on three scenarios. Previous work [5] showed that the overhead of the mapping strategy is 0.69 seconds on average for eight instances, the calculated execution time does not include this value. After that, we will show how those benchmarks can perform when executing in a sequential fashion that we call *bundle*, simulating different stages of a scientific application. And finally, we use these results to calculate improvements in potential task migration scenarios.

A. NAS benchmarks

As mentioned in Section II-B, the mapping scenarios are Default, Random, Interleaved, Bandwidth and number of messages (*bw-num*), Bandwidth and volume of messages (*bw-vol*), Latency and number of messages (*lat-num*) and Latency and volume of messages (*lat-vol*). Performance results of using these mapping strategies are shown in Table II while the analysis of each scenario will be explained with the Figures.

Figure 4 shows the results normalized to *Default*, there are ten executions for every task mapping scenario, along almost 127 hours for D32v3 and 115 hours for D64v3 VMs. This interval of time may include daily variations caused by tenants usage on the local data center region.

We can observe that for some of the benchmarks the mechanism improves the performance by reducing the execution time of the application. However, not all of the task mapping approaches are reducing execution time by the same amount; some of them even impact their performance negatively.

TABLE II: Performance results of the NAS benchmarks with input data size C and D on cloud deployments of D32-v3 and D64-v3 instances. Values shown are the geometric mean of at least 10 executions in seconds.

Instance Size	NAS Size	Rank Mapping	Profile Type	Comm Metric	Benchmark							
					BT	CG	EP	FT	IS	LU	MG	SP
D32-v3	C	default	none	none	29.7536	23.9289	1.3286	20.4254	5.9924	36.1243	4.6449	39.8769
		random	none	none	32.3274	48.8427	1.3484	18.1698	3.7386	54.8401	4.4137	49.3949
		interleaved	none	none	29.3538	44.7635	1.3357	17.9333	3.9023	44.4580	4.0735	43.3714
		scotch	num	bandwidth	29.7676	46.6722	1.3384	18.2950	3.7910	52.7764	4.4362	45.3896
				latency	22.4867	20.1665	1.3387	18.2219	3.7810	31.8359	3.5784	32.1054
			size	bandwidth	29.6184	48.7118	1.3261	18.1428	3.7374	49.9181	4.4546	47.7415
				latency	31.3813	47.0324	1.3429	18.1756	3.8111	50.8057	4.1085	47.8391
	D	default	none	none	321.8664	212.7627	20.3128	208.4309	26.9994	257.7411	42.0286	505.3501
		random	none	none	337.3782	430.7862	20.3022	203.7957	23.5531	335.7089	40.4548	514.2740
		interleaved	none	none	322.9150	400.9783	20.3949	201.6204	23.1649	294.6353	38.9205	500.1284
		scotch	num	bandwidth	341.1078	405.3995	20.5828	205.9066	27.2643	331.7393	41.0833	498.3493
				latency	303.7208	200.1455	20.3968	204.7435	27.0185	247.3231	37.9792	459.5020
			size	bandwidth	329.8056	406.5129	20.4425	204.4810	27.5156	319.3964	40.1179	505.5926
				latency	336.6124	410.0144	20.3811	206.4881	27.3741	323.3661	39.2909	501.5793
D64-v3	C	default	none	none	27.5845	25.2198	1.2386	24.1031	5.1791	31.4335	3.2507	32.8213
		random	none	none	35.6062	56.7521	1.2495	19.8719	4.3384	58.6961	3.8031	56.3047
		interleaved	none	none	33.0472	45.1456	1.2362	20.5406	5.2690	48.1260	2.8302	50.2464
		scotch	num	bandwidth	29.7168	45.0079	1.3316	20.2451	4.2578	47.8185	3.4045	44.5116
				latency	21.2172	16.7012	1.3413	20.7085	4.4661	28.4378	2.6570	27.3097
			size	bandwidth	32.1365	42.5841	1.2225	20.2275	4.3909	46.3115	3.7898	44.3950
				latency	28.3604	46.2326	1.6542	20.5370	4.5940	46.0683	3.4692	43.9499
	D	default	none	none	314.6325	238.9455	19.3585	262.1599	33.2924	232.0928	40.8623	464.0513
		random	none	none	340.8709	551.5821	19.5452	249.6640	30.2030	321.0081	36.6701	465.8618
		interleaved	none	none	329.7481	423.7369	19.4710	242.6714	29.3039	296.3761	30.6474	451.6384
		scotch	num	bandwidth	337.7699	447.5160	18.8907	252.1807	31.8865	286.0136	33.7296	430.0693
				latency	267.6051	206.9754	19.3122	252.0348	31.9092	209.8749	30.2986	372.4798
			size	bandwidth	331.2000	446.2056	19.2685	251.7218	31.6197	280.6028	34.2415	417.6780
				latency	335.5737	446.1213	19.3483	251.7108	31.7285	299.3473	31.6934	417.9733

In the case of CG and LU, they only show improvements using lat-num (up to 16% faster). These two benchmarks have mostly point-to-point communication with a large number of messages, so improvements using latency make sense. CG is sensitive to long-distance communication; this fact explains why the best task mapping was achieved using latency plus the number of messages interchanged on its communication matrix. CG Depends on the availability of both high bandwidth and low latency [18]. However, using lat-vol does not generate significant improvements. This result is a clear indicator that using only a single metric provides marginal improvements in some scenarios.

BT, LU, and SP perform mostly point to point communication too. However, their results are slightly different. Their performance improves by lat-size too (up to 24% faster), but the other mappings do not produce a significant performance drop as in the case of CG and LU. Their communication pattern is less sensitive to bandwidth than CG, and for this reason, a fraction of the improvements come from intra-mapping optimizations.

FT and IS use mostly collective messages and obtain improvements up to 36% for IS on NAS size C. Most of the performance improvements come from better intra-instances task mapping. EP did not have any improvements; the fact that

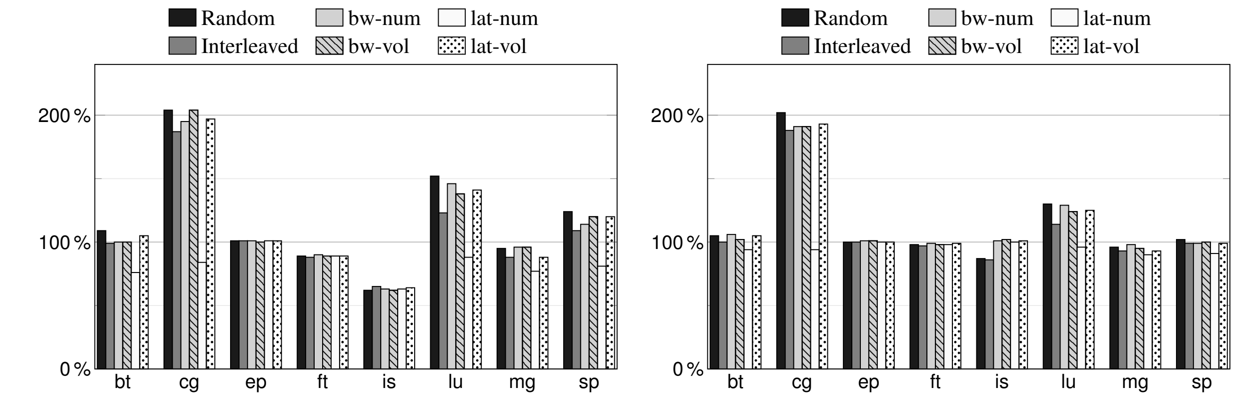
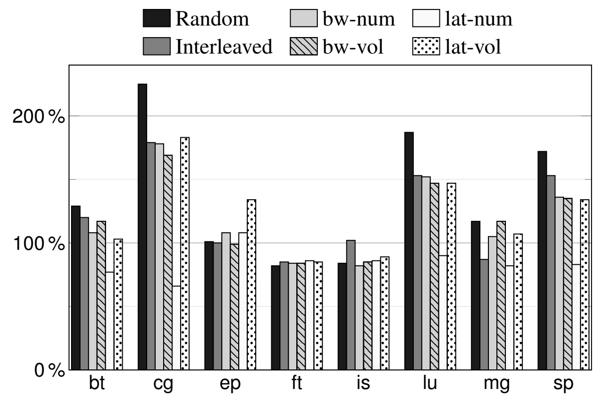


Fig. 4: Normalized execution time of D32v3 using NAS input size C (left) and using NAS input size D (right). Values lower than 100% mean faster execution time than baseline.



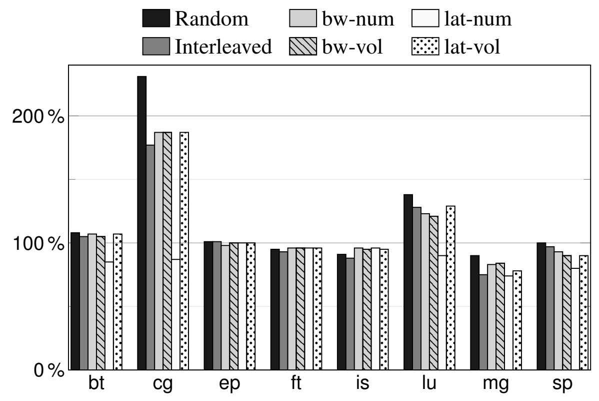


Fig. 5: Normalized execution time of D64v3 using NAS input size C (left) and using NAS input size D (right). Values lower than 100% mean faster execution time than baseline

EP performs almost no communication during its execution explains the result obtained.

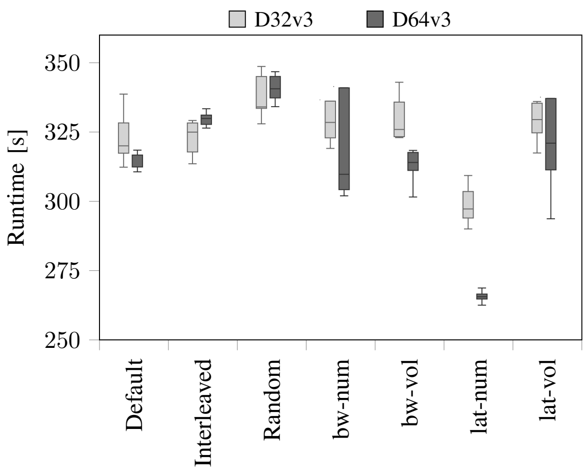
Figure 5 shows the results for D64v3. The first thing to notice is that results from Random are worse on this VM instance size. Results are consistent with those from D32v3, with performance improvements only in some benchmarks. The important takeaway from these results is validating lat-num as a blend metric that allows improving performance in most of the cases for the NAS benchmarks. In the cases that lat-num is not the best metric, the results obtained are not distant from the better metric. The results showed that we could use lat-num as a default metric for the task mapping mechanism to improve performance on NAS applications. Another important aspect is to analyze if the performance was stable or there were fluctuations during the experiments. The network variability, as showed previously in Figure 1, may impact some experiments more than others depending on whether they are more sensitive to latency, bandwidth or both.

Figure 6 shows concisely the variability of the results using a box plot. In most of the cases, the variability is higher on bandwidth dependent tests. Is also noticeable that D64v3 is more stable over the several variations of the task mapping experiments. Notice that results became more stable for bigger NAS input sizes.

B. Application Phases

After identifying *lat-num* as the best of the proposed approaches for task mapping we need to look into the following scenario. What happens whenever an application present changes its communication pattern (i.e., when it presents multiple phases) and how can our technique react to those changes using only the available information as a tenant. Such changes can be annotated inside the application code, signaling our mapping technique to measure the current state and perform a new task mapping allocation.

Phases can represent parts in scientific applications that have iterative processing stages where its communication



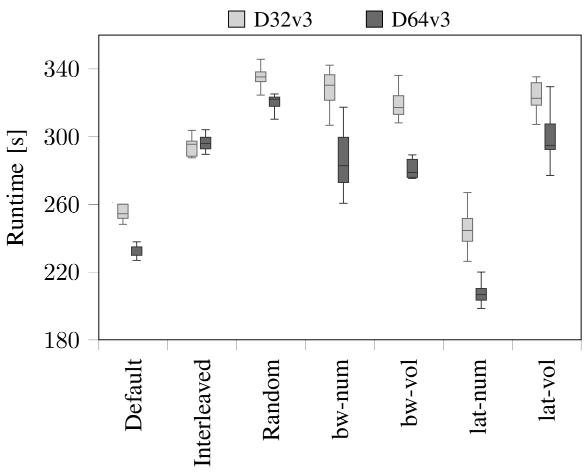


Fig. 6: Performance comparison of using different task mapping mechanisms on the BT benchmark (left) and LU benchmark (right) with input size D, on a cluster of D32v3 or D64v3 instances

pattern changes when it starts. When such a situation happens, the application may synchronize using a barrier to load or communicate new data to continue processing; this resembles multiple executions of several algorithms inside an application. Joining several phases together form a *bundle*. To simulate such behavior we analyzed the same bundle of applications in three scenarios:

- Without using any task mapping
- Task mapping only for the first execution phase
- Perform task mapping before every phase in the bundle

The main idea of task mapping at every phase execution is to recalculate the state of the network and improve upon that information by generating a new rank file specifying in which node allocates each task should at that moment. A scientific application consists of several phases, and by using the mechanism evaluated until now, we had no way to react to such changes.

C. Performance Impact

An incorrect task mapping generated by using network characterization only at the start of the application and not changing the mapping even when the application communication pattern changed can lead to degraded performance. In order to understand the performance impact of incorrect task mapping, we ran every benchmark using the communication pattern from all the other benchmarks. By doing so, we obtained an impact matrix that helps to evaluate the performance loss due to wrong mapping on multi-phased applications.

Table III shows the normalized results of running the benchmarks with the incorrect phase. In some cases, such as when an LU phase occurs after an initial CG phase, the impact causes a performance hit that slows down the application up to 2.95x. On average the improvements are near 1% on average, those can be due to fluctuations caused by the variable nature of the cloud at the data center network, and not actual improvements achieved by using the wrong communication matrix. By using the impact matrix, we can estimate the execution time wasted by an application as a penalty for using the wrong communication task mapping.

TABLE III: Normalized results of the impact of using the communication pattern from another benchmark, D64v3 - Size D.

bench	Mapping Source (latency + number of messages)							
	bt	cg	ep	ft	is	lu	mg	sp
bt	1.0000	1.1143	0.9815	0.9899	0.9879	1.0412	1.2156	1.0492
cg	2.0841	1.0000	0.9977	1.2080	1.2043	2.2560	1.7589	2.1601
ep	1.0058	1.0050	1.0000	1.0085	0.9920	0.9894	0.9966	0.9977
ft	1.0173	1.0269	1.0007	1.0000	0.9982	1.0276	1.0203	1.0252
is	0.9824	1.0299	1.0003	0.9949	1.0000	0.9922	0.9960	1.0084
lu	1.0106	1.1920	0.9396	0.9425	0.9503	1.0000	1.4486	1.0254
mg	1.2017	1.0564	1.0397	1.0220	0.9853	1.0855	1.0000	1.1387
sp	0.9981	1.0800	0.9885	0.9864	0.9841	0.9987	1.1950	1.0000

D. Performance gap

The performance gap is the time difference of executing a task mapping allocation on a multi-phased application using only a measurement of the current network performance at the application start versus measuring every time the application's communication pattern changes.

To illustrate several cases we used four NAS benchmarks selected at random without repetition from the benchmarks that are point-to-point communication-based. The worst case may include repeated NAS benchmarks/kernels. We did not consider EP and IS, both collective communication based, as their impact was negligible to the other benchmarks as shown in III. Figure 7c shows a 6.87% increase on total execution time caused by performing task mapping only at the execution start and a reduction of 23.03% when the mapping is calculated during runtime at every code annotation. A similar result is achieved in the other two scenarios proposed in Figure 8. The negative impact on Figure 7b is 17.17%, and the improvement achieves 28.32%. On Figure 7a, total execution time takes 8.20% more with *Only First* mapping and an improvement of 26.05% using *At Every* communication pattern change. The performance gap allows us to estimate the percentage of time available to perform algorithm improvements in the future, i.e., migrating tasks between VMs in the same deployment.

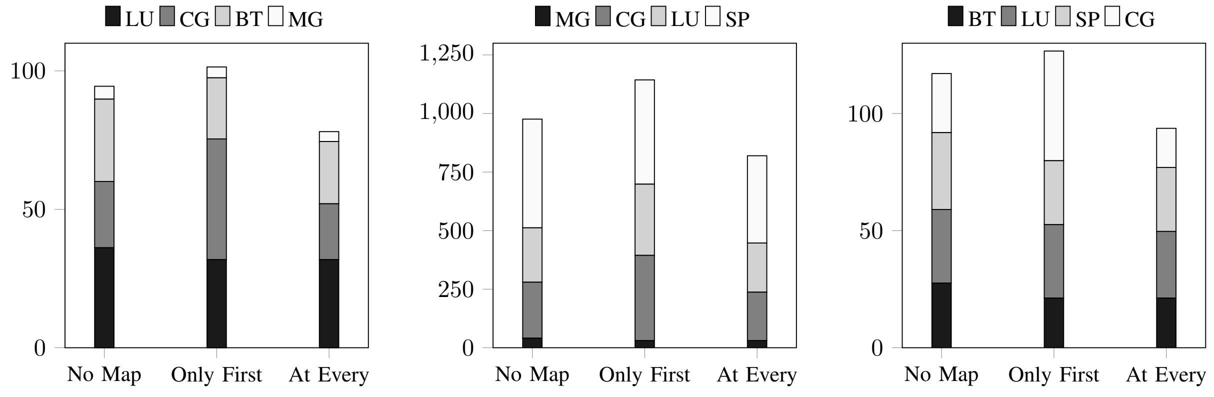


Fig. 7: Performance gap on running three bundle of four NAS benchmarks comparing their execution time when using the three different scenarios. The vertical axes show execution time in seconds.

IV. Related Work

Some authors have researched some of the topics mentioned in this paper, to compare our work with theirs we split their contribution into three parts. The first describes previous work on task mapping using clusters or cloud infrastructures. The second part shows work on the study of Network variability in the cloud. The third shows how cloud infrastructures have been used or evaluated by their suitability to run HPC applications.

A. Task Mapping:

The work of Rak et al. [19] presented the cMe (cloud MPI enabler) which builds cloud-based cluster on cloud providers. Their work relies over the mOSAIC platform, that builds a PaaS over cloud providers. However, the authors do not provide experimentation's and their work present only the definition of the cMe.Chanchio and Thaenkaew [3] proposed the Time-bound thread-based Live Migration (TLM) mechanism for VM migration. They aim to minimize the downtime and the impact to other VM during the migration process; They develop a prototype of TLM over KVM. They demonstrate that TLM can migrate both OpenMP and MPI versions of half of the NAS benchmarks Suite.

Vu and Hwang [4] proposed a VM algorithm that minimizes the communication cost and reduces the energy consumption. Their algorithm could reduces the network traffic congestion in a simulated data center, however the energy consumption was unmodified. The work of Saad and El-Mahdy [2] proposes an analytic model that aims to identify the network characteristics between VM, by performing a set of point-to-point automated tests.

B. Network Variability:

The work of Persico et al. [20] measured the network performance of Inter Datacenter communication, for AWS and Azure providers. They deployed VM instances on five different geographic locations and performed throughput and latency tests. They conclude that the latency is comparable on both providers and the throughput is around 50% better on Azure.

Regarding intra-cloud performance, the work of Persico et al [9] aims to create a methodology to identify the network throughput and applied it to Azure. They performed several tests over time and were able to conclude that the network is stable over time when the instances were deployed and not reallocated. Instances with bigger sizes performed better than the smallest ones as well.

The work of Filer et al. [21] provides an in-depth analysis of the Elastic Optical Networking (EON). Their focus is on how Microsoft's Azure could leverage the recent advances in Optical Network to improve the overall communication of the cloud service.

C. Cloud for HPC:

The work of Prabhakaran and Lakshmi [22] evaluate the cost-benefit of using Amazon, Google and Microsoft cloud instances to execute HPC jobs instead of the SahasraT supercomputing from Indian Institute of Science, the cost-benefit of the supercomputer were better than the cloud. They calculated a cost of US$ 0.0126 per core per hour using 100% of the supercomputer uninterruptedly for five years and compared to the cost of the cloud instances. However, since the cloud instances are available in an on-demand base, the authors do not provide production data of the utilization rate of the SahasraT, the cost comparison could change.The work of Kotas et al. [23] compared the AWS and Azure as a platform for HPC, they evaluate one instance of each provider and build several cluster configurations, varying from 1 to 32 nodes. The authors executed the HPCC and HPCG benchmarks suite. They found that the AWS instance offers a better performance rate per dollar and Azure offers better bandwidth. And the user needs to identify which provider is best suitable for a given application.

Aljamal et al. [24] compared Azure, AWS, Google Cloud and Oracle Cloud as a platform for HPC. The authors made a comparative analysis of the provider's offers. However, they did not perform a simulation. They concluded that there is not a cloud provider that fits all user requirements.

Different from the related works, where the authors have focused on task mapping or network performance, our work provides a model that combines the task mapping with the network performance of public cloud to provide better performance for HPC applications.

V. Conclusions

Despite improvements on several performance aspects on cloud data center infrastructures over the years, network variability on latency and bandwidth remains as an issue.

In this paper, we evaluated several metrics that allowed improving performance on scientific parallel applications over a public cloud infrastructure. We evaluated our proposal using a set of parallel benchmarks running on an actual public cloud. In our analysis, we identified that using the instantaneous latency on the cloud network and the volume of messages interchanged by an application provides the best results for the evaluated benchmarks. Average results from the two VM sizes achieved 13.57% using latency and number of messages as the best metric for CloudMap. Regarding the impact of incorrect task mapping allocation, results showed degraded performance up to 2.96x on a specific benchmark and up to 19.98% when an application has several distinct phases. In contrast, using a correct allocation for every phase using our proposal, we can achieve improvements of up to 28.32%. From these results we can conclude that in most cases there is enough time to perform a task migration while still improving total execution time and reducing costs. Our experiments showed that there is room for improvement on the task mapping algorithm used, and also explained how the different application phases may benefit from those improvements to obtain more performance gains. The only downside of the proposed approach is that benchmarks and applications that are inherently point to point communication based can benefit more from the mechanism. Our results also showed that our proposal is a solution to reduce in part the cloud network bottlenecks caused by the shared resources model that the cloud infrastructures offer.

Acknowledgment

This work was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001.

References

[1] B. Varghese and R. Buyya, "Next generation cloud computing," *Future Gener. Comput. Syst.*, vol. 79, no. P3, pp. 849–861, Feb. 2018.

[2] A. Saad and A. El-Mahdy, "Network Topology Identification for Cloud Instances," in *International Conference on Cloud and Green Computing*, 2013, pp. 92–98.

[3] K. Chanchio and P. Thaenkaew, "Time-bound, thread-based live migration of virtual machines," *Int. Symp. on Cluster, Cloud, and Grid Computing*, pp. 364–373, 2014.

[4] H. T. Vu and S. Hwang, "A Traffic and Power-aware Algorithm for Virtual Machine Placement in Cloud Data Center," *International Journal of Grid and Distributed Computing*, vol. 7, no. 1, pp. 350–361, 2014.

[5] E. D. Carreno, M. Diener, E. H. Cruz, and P. O. Navaux, "Automatic Communication Optimization of Parallel Applications in Public Clouds," *Proceedings - 2016 16th IEEE/ACM International Symposium on Cluster, Cloud, and Grid Computing, CCGrid 2016*, pp. 1–10, 2016.

[6] L. Zuo, S. Dong, L. Shu, C. Zhu, and G. Han, "A multiqueue interlacing peak scheduling method based on tasks' classification in cloud computing," *IEEE Systems Journal*, vol. 12, no. 2, pp. 1518–1530, June 2018.

[7] D. Grove and P. Coddington, "Precise MPI performance measurement using MPIBench," in *Proceedings of HPC Asia*, 2001, pp. 24–28.

[8] G. Wang and T. S. E. Ng, "The impact of virtualization on network performance of amazon ec2 data center," in *2010 Proceedings IEEE INFOCOM*, March 2010, pp. 1–9.

[9] V. Persico, P. Marchetta, A. Botta, and A. Pescape, "On network throughput variability in microsoft azure cloud," *Global Communications Conference*, no. ii, 2015.

[10] J. Sonnek, J. Greensky, R. Reutiman, and A. Chandra, "Starling: Minimizing Communication Overhead in Virtualized Computing Platforms Using Decentralized Affinity-Aware Migration," in *Int. Conf. on Parallel Processing*, sep 2010, pp. 228–237.

[11] J. Slawinski, U. Villa, T. Passerini, A. Veneziani, and V. Sunderam, "Issues in Communication Heterogeneity for Message-Passing Concurrent Computing," in *Int. Symp. on Parallel & Distributed Processing, Workshops and Phd Forum*, 2013, pp. 93–102.

[12] A. Gupta, L. V. Kalé, D. Milojicic, P. Faraboschi, and S. M. Balle, "HPC-aware VM placement in infrastructure clouds," in *Int. Conf. on Cloud Engineering*, 2013, pp. 11–20.

[13] L. Yin, J. Sun, L. Zhao, C. Cui, J. Xiao, and C. Yu, "Joint Scheduling of Data and Computation in Geo-Distributed Cloud Systems," in *Int. Symp. on Cluster, Cloud and Grid Computing*, 2015, pp. 657–666.

[14] D. H. Bailey, E. Barszcz, J. T. Barton, D. S. Browning, R. L. Carter, L. Dagum, R. A. Fatoohi, P. O. Frederickson, T. A. Lasinski, R. S. Schreiber *et al.*, "The nas parallel benchmarks," *The International Journal of Supercomputing Applications*, vol. 5, no. 3, pp. 63–73, 1991.

[15] S. White, A. Alund, and V. S. Sunderam, "Performance of the nas parallel benchmarks on pvm-based networks," *Journal of Parallel and Distributed Computing*, vol. 26, no. 1, pp. 61–71, 1995.

[16] F. Trahay, F. Rue, M. Faverge, Y. Ishikawa, R. Namyst, and J. Dongarra, "EZTrace: a generic framework for performance analysis," in *International Symposium on Cluster, Cloud and Grid Computing (CCGrid)*, 2011, pp. 618–619.

[17] M. Azure, "Compute benchmark scores for linux vms," https://docs.microsoft.com/en-us/azure/virtual-machines/linux/compute-benchmark-scores, Apr. 2019.

[18] J. Dongarra, M. A. Heroux, and P. Luszczek, "Hpcg benchmark: a new metric for ranking high performance computing systems," *Knoxville, Tennessee*, 2015.

[19] M. Rak, M. Turtur, U. Villano, and L. Pino, "A portable tool for running MPI applications in the cloud," *Int. Conf. on Intelligent Networking and Collaborative Systems*, pp. 10–17, 2014.

[20] V. Persico, A. Botta, A. Montieri, and A. Pescapé, "A first look at public-cloud inter-datacenter network performance," *Global Communications Conference*, 2016.

[21] M. Filer, J. Gaudette, M. Ghobadi, R. Mahajan, T. Issenhuth, B. Klinkers, and J. Cox, "Elastic Optical Networking in the Microsoft Cloud," *Journal of Optical Communications and Networking*, vol. 8, no. 7, p. A45, 2016.

[22] A. Prabhakaran and J. Lakshmi, "Cost-benefit Analysis of Public Clouds for offloading in-house HPC Jobs," *Int. Conf. on Cloud Computing*, pp. 57–64, 2018.

[23] C. Kotas, T. Naughton, and N. Imam, "A comparison of Amazon Web Services and Microsoft Azure cloud platforms for high performance computing," *Int. Conf. on Consumer Electronics, ICCE 2018*, vol. 2018-Janua, pp. 1–4, 2018.

[24] R. Aljamal, A. El-Mousa, and F. Jubair, "A comparative review of high-performance computing major cloud service providers," *Int. Confe. on Information and Communication Systems*, vol. 2018-Janua, pp. 181–186, 2018.