A Performance Comparison of HPC Workloads on Traditional and Cloud-based HPC Clusters

Vanderlei Munhoz*, Antoine Bonfils†, Márcio Castro*, Odorico Mendizabal*
*Federal University of Santa Catarina, Florianópolis, Brazil
†Polytech Grenoble, Grenoble, France
vanderlei.filho@posgrad.ufsc.br, antoine.bonfils@etu.univ-grenoble-alpes.fr,
marcio.castro@ufsc.br, odorico.mendizabal@ufsc.br

***Abstract*—Cloud Computing allows users to access large computing infrastructures quickly. In the High Performance Computing (HPC) context, public cloud resources emerge as an economical alternative, allowing institutions and research groups to use highly parallel infrastructures in the cloud. However, parallel runtime systems and software optimizations proposed over the years to improve the performance and scalability of HPC applications targeted traditional on-premise HPC clusters, where developers have direct access to the underlying hardware without any kind of virtualization. In this paper, we analyze the performance and scalability of HPC applications from the NAS Parallel Benchmarks suite when running on a virtualized HPC cluster built on top of Amazon Web Services (AWS), contrasting them with the results obtained with the same applications running on a traditional on-premise HPC cluster from Grid'5000. Our results show that CPU-bound applications achieve similar results in both platforms, whereas communication-bound applications may be impacted by the limited network bandwidth in the cloud. Cloud infrastructure demonstrated better performance under workloads with moderate communication and medium-sized messages.**

***Index Terms*—High Performance Computing, Cloud Computing, NAS Parallel Benchmarks, Performance Evaluation**

I. Introduction

High Performance Computing (HPC) enables innovative scientific research in several areas of knowledge, such as Engineering, Medicine, Biology, Meteorology, among others. Often, research carried out in these areas is heavily based on using mathematical models, computational methods and/or numerical simulations to study existing problems and propose new solutions.

Traditionally, HPC clusters are deployed in large institutions or computing centers that carry out research that demands high computational power (a solution known as *on-premise*). In general, access to these computational resources is restricted to researchers from the institutions or computing centers themselves, thus limiting sharing of these computational resources with external researchers. Few institutions or research centers have such large-scale infrastructures, as the cost of acquisition and maintenance is extremely high.

This work was partially funded by the National Council for Scientific and Technological Development (CNPq) and Amazon Web Services (AWS) through the CNPq/AWS call Nº 64/2022 - Cloud Credits for Research. The authors would also like to thank Jean-François Méhaut for granting access to Grid'5000 infrastructure.

On the other hand, the Cloud Computing paradigm allowed greater democratization of access to large data processing and storage infrastructures to millions of organizations and individuals with few capital resources, applying massive economies of scale and reducing costs with Information Technology (IT) [1]. Public clouds rely on the same model for renewing and maintaining the physical infrastructure of large computing centers but differ in their business model. Cloud providers have their business focused on optimizing the use of physical infrastructure, regardless of the focus of the application or the user. Cloud resources are offered to clients based on a *pay-as-you-go* pricing model [2], reducing the entry barrier for small institutions and research groups as they can create and dispose of resources as needed [3], paying solely for what they use. With the ability to dynamically scale resources in real-time as per their application requirements, users can avoid over-provisioning or under-provisioning resources. According to Gartner,[1] Amazon Web Services (AWS), Microsoft Azure, Alibaba Cloud, Google Cloud Platform (GCP), and Huawei Cloud are the top public cloud providers today, collectively accounting for 80% of the market share.

Originally, Cloud Computing platforms were created and later optimized to support commercial web applications running on virtualized hardware over a shared physical infrastructure. In this way, the paradigm aims to allow the quick allocation and deallocation of virtualized resources in a dynamic way [4]. However, cloud providers have recently developed specialized products and services for the HPC domain. These products and services often include computing infrastructure options with high-speed interconnects, large instances with dozens of virtual CPUs (vCPUs), and instances equipped with Graphics Processing Units (GPUs) or Tensor Processing Units (TPUs) to accelerate parallel simulations and Machine Learning. Some providers also offer *bare metal* instances, giving users direct access to hardware without any virtualization. These recent advances in cloud infrastructures can allow the HPC community to leverage Cloud Computing resources on-demand, drastically reducing resource waste and maintenance costs of (traditional) on-premise HPC clusters [5], [6].

However, the HPC ecosystem available in public clouds

[1] https://www.gartner.com/en/newsroom/press-releases/2022-06-02-gartner-says-worldwide-iaas-public-cloud-services-market-grew-41-percent-in-2021

979-8-3503-8160-3/23/$31.00 ©2023 IEEE
DOI 10.1109/SBAC-PADW60351.2023.00026

has not yet evolved enough to support widespread use, not only due to technological limitations but also the laborious process of migrating legacy HPC applications to a completely different environment than the one in which they were initially designed [7]. In addition, existing runtime systems and legacy HPC applications are optimized for on-premise HPC clusters, where users have direct access to the underlying infrastructure without any kind of virtualization. In this paper, we compare the performance and scalability of HPC applications extracted from the well-known NAS Parallel Benchmarks (NPB) suite on two platforms: a virtualized HPC cluster built on top of Amazon Web Services (AWS)[2] and a traditional on-premise HPC clusters from Grid'5000 (G5K).[3] Our results show that CPU-bound applications achieve similar results in both platforms, whereas communication-bound applications may be impacted by the limited network bandwidth in the cloud. Cloud infrastructure demonstrated better performance under workloads with moderate communication and medium-sized messages.

This paper is structured as follows. Section II discusses the fundamentals of the platforms and applications considered in this study. Section III presents the experimental methodology adopted in this work. Section IV provides details on the gathered experimental data and discusses the obtained results. Section V discusses related work in this area. Finally, Section VI concludes the paper by summarizing our findings and outlining future work.

II. Background

In this section, we first present an overview of the platforms used in our experimental evaluation. Then, we briefly discuss the HPC applications considered in this study.

A. Grid'5000 (G5K)

G5K is a renowned testbed that provides researchers with a large-scale and versatile platform for conducting experiment-driven research across various Computer Science domains [8]. It is designed to support Open Science and reproducible research, with full traceability of infrastructure and software changes on the testbed.

It offers a dedicated infrastructure that enables researchers to design and execute experiments in a controlled environment. It provides a wide range of resources, approximately 15,000 cores, and 800 compute nodes, including 9 computing cluster sites, storage systems, and networking infrastructure, allowing researchers to investigate and evaluate different aspects of computer science at a large scale. It is used in all areas of Computer Science, focusing on parallel and distributed computing, covering areas such as Cloud Computing, HPC, Big Data, and Artificial Intelligence (AI).

The platform is highly reconfigurable and controllable, so researchers can experiment with a fully customized software stack thanks to bare metal deployment features and isolate their experiment at the networking layer. It also provides advanced monitoring and measurement features for trace collection of networking and power consumption, providing a deep understanding of experiments.

B. Amazon Web Services (AWS)

Public clouds are designed to be accessible to anyone via the Internet without the need for long-term contracts or direct interaction with the provider. Three types of services traditionally describe their service model. *Infrastructure as a Service* (IaaS) refers to online services that provide APIs for users to spawn and manage compute infrastructure, including low-level details such as network, storage, and backups. The user often can choose the computing capacity of the infrastructure to be rented — typically in terms of virtual CPUs (vCPUs) — and can also configure other details such as hypervisor types, pre-installed OS, accelerators, and more. *Platform as a Service* (PaaS) refers to services that the user can use to create and deploy custom software applications using a configurable environment hosted by the cloud provider. Runtime, middleware, and software features are abstracted from the user and managed by the provider. Finally, *Software as a Service* (SaaS) are ready-to-use software applications with specific purposes that the provider typically offers through APIs, which users can use directly in their applications.

AWS is one of the most popular public cloud providers. It maintains physical resources in several geographically dispersed data centers, giving users access to computing resources in the form of *instances*. A Virtual Machine (VM) is a virtually allocated instance on a shared physical infrastructure maintained by AWS and is generally the most available and affordable option. AWS offers various types of instances with different characteristics such as type of *hypervisor*, CPU architecture, number of vCPUs, amount of memory, and much more, allowing users to choose the instance that best suits their needs.

Recently, AWS has launched numerous products and services for HPC. Its offerings include infrastructure options featuring (i) high-speed interconnections, which employ its proprietary Elastic Fabric Adapter (EFA) network interface that enables customers to run applications requiring high levels of inter-node communications at scale; (ii) large VM instances equipped with hundreds of vCPUs to cater to applications with substantial processing demands; (iii) instances equipped with accelerators such as GPUs or TPUs; (iv) fully managed shared storage named FSx built on popular high-performance file systems (e.g., Lustre and OpenZFS).

C. NAS Parallel Benchmarks

The NAS Parallel Benchmarks (NPB)[4] is a suite of parallel benchmark programs designed by the NASA Advanced Supercomputing (NAS) Division [9]. By measuring various aspects of performance, such as computational speed and efficiency, NPB provides a standardized way for researchers and developers to assess the capabilities of parallel computing

[2]http://aws.amazon.com/

[3]http://grid5000.fr

[4]https://www.nas.nasa.gov/software/npb.html

systems, including clusters, supercomputers, and distributed memory architectures. It focuses on various aspects of parallel computing, including computation, communication, and memory access patterns.

NPB kernels are derived from Computational Fluid Dynamics (CFD) applications, representing real-world aerospace and engineering applications. Reference implementations of the benchmarks are provided by NASA using different parallel programming libraries such as OpenMP and Message Passing Interface (MPI). In this paper, we used the most recent reference MPI implementation (version 3.4.2), which includes five kernels (two implemented in C and three in Fortran).

Input problem sizes in NPB are predefined and indicated as different classes. The standard sizes consist of six classes (S, W, A, B, C, and D): (i) S and W are designed for quick test purposes; (ii) A, B, and C are the standard inputs, with each class experiencing approximately a four-fold increase in size compared to the previous one; and (iii) D, E and F are the large inputs, with each class experiencing approximately a sixteen-fold increase in size compared to the previous one. A brief description of the kernels considered in this paper are given below:

- **Integer Sort (IS)**: It performs an integer sorting (bucket sort) among a sparse set of numbers, which simulates an important computation for particle-in-cell applications. This kernel simulates and measures integer computation and data communication capabilities.
- **Embarrassingly Parallel (EP)**: It is an embarrassingly parallel benchmark that generates pairs of Gaussian random deviates according to a specific scheme. This kernel features negligible communication overhead.
- **Conjugate Gradient (CG)**: It employs a Conjugate Gradient method to compute an approximation to the smallest eigenvalue of a large, sparse, unstructured matrix. This kernel tests unstructured grid computations and communications by using a matrix with randomly generated locations of entries. It features irregular memory access and communication patterns.
- **Multi-Grid (MG)**: It uses a V-cycle MultiGrid method to compute the solution of the 3D scalar Poisson equation. The algorithm works continuously on a set of grids that are made between coarse and fine. It tests both short and long-distance data movements.
- **Discrete 3D Fast Fourier Transform (FT)**: It contains the computational kernel of a 3D Fast Fourier Transform (FFT)-based spectral method. FT performs three one-dimensional (1D) FFTs, one for each dimension, and features long-distance communications.

III. Experimental Methodology

G5K and AWS feature a large diversity of hardware configurations. To make a fair comparison, we first analyzed the available configurations of processors and interconnection networks in both infrastructures. We compared the available processors using the following criteria: processor generation, number of cores, processor launch date, and Last-level Cache (LLC) size. Based on the available clusters in G5K and VM instances in AWS, we selected the *Dahu* cluster from G5K and *c5n.4xlarge* VM spot instances from AWS Elastic Compute Cloud (EC2). It is worth mentioning that although it would be possible to use bare metal instances in AWS, they are substantially more expensive than the virtualized ones (6 times more expensive during our experiments). Because of that, we did not select these instances in our study. AWS also offers HPC-optimized instances, such as the *hpc6a.48xlarge*, but we did not find similar hardware configurations in G5K, so we did not use those instances. More details about the processor configurations selected for this study are shown in Table I.

TABLE I: Selected processor configurations from G5K and AWS.

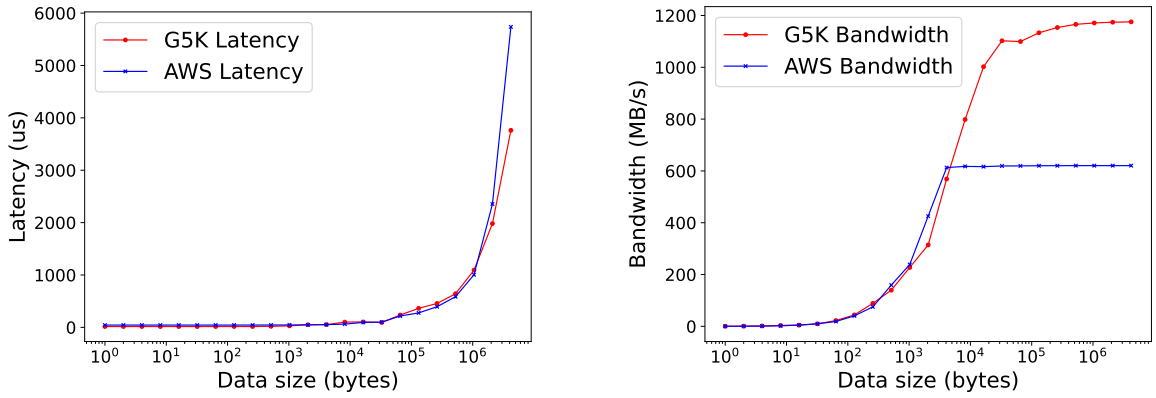
	G5K	**AWS**
Code Name	Skylake	Skylake
Model	Intel Xeon Gold 6130	Intel Xeon Platinum 8124M
Lithography	14 nm	14 nm
Launch Date	2017	2017
Cores/vCPUs	16	16
Clock frequency	2.1 GHz	3.0 GHz
Cache size (LLC)	22 MB	24 MB

We considered clusters composed of 1 (single node), 2, 4, and 8 nodes in both platforms and we evaluated the results with a variable number of MPI processes per node (1, 2, 4, 8 and 16). Nodes were interconnected by either a 10 Gbps Ethernet network (G5K) or a 25 Gbps Ethernet network (AWS). Considering the availability of VM instances in AWS and their monetary costs, we selected the appropriate *C class* for all experiments. The execution time of each application/configuration using the C class ranged from a few seconds to a hundred seconds, depending on the application and number of MPI processes.

Nodes from the *Dahu* cluster (G5K) were reserved using the OAR[5] batch scheduler, which gave us exclusive access to the reserved nodes. We leveraged the HPC@Cloud toolkit [10], [11] to build the cluster on top of AWS/EC2. HPC@Cloud is an open-source toolkit that offers a suite of tools that enable users to configure cloud infrastructure, execute jobs, monitor performance, predict costs, and interact with the provisioned resources in an automated and provider-agnostic manner. The same software stack was installed on both platforms, including the Operating System (CentOS v7.8), GCC (v7.3), and OpenMPI (v5.1).

We relied on the overall execution times of the NPB kernels' output to assess the scalability of NPB kernels on both platforms. All results in this paper are based on the average execution times taken from 10 repetitions of each experiment. Considering all possible scenarios (5 NPB kernels $\times$ 4 numbers of nodes $\times$ 5 numbers of MPI processes per node $\times$ 2 platforms $\times$ 10 repetitions of each experiment), the average and maximum standard deviation observed in

[5]http://oar.imag.fr



(a) Latency results

(b) Bandwidth results

Fig. 1: Network performance comparison using OSU microbenchmarks (OMB).

G5K were 0.14 and 0.82 seconds, respectively, and in AWS were 0.16 and 0.85 seconds, respectively. Speedups were calculated by dividing the average execution times of NPB kernels running with a single MPI process (single node) by their average execution times when running in parallel (one or more nodes). The scalability results shown in Section IV-B with 2, 4, 8, 16, 32, 64, and 128 MPI processes were obtained from the combinations of number of nodes and number of MPI processes per node that gave the best results in each platform.

IV. Results

A. Latency and Bandwidth

To get a better overview of the network performance of both platforms, we first executed two microbenchmarks to measure the latency and bandwidth while stressing inter-node communication. We chose the OSU Micro Benchmarks (OMB) suite[6] for these experiments, which is a well-known microbenchmark for measuring and evaluating the performance of communication operations. OMB offers a set of host-based microbenchmarks that can evaluate the sustained message passing bandwidth and latency between two compute nodes. Although OMB also implements benchmarks for testing OpenSHMEM, UCP, and CUDA, we focused on the MPI benchmark implementation (OMB v 7.2). A description of the microbenchmarks evaluated in this paper is given next:

- **Latency**: We chose the `osu_latency` test to measure the ping-pong latency between two nodes. A process sends a message with a certain data size to the receiver and waits for the reply. Upon receiving a message from the sender, the receiver replies with the same data size. This ping-pong test executes many successive iterations, and average one-way latency numbers are obtained. The blocking version of MPI functions (*MPI_Send* and *MPI_Recv*) are used in the tests.
- **Bandwidth**: For bandwidth evaluation, we used the `osu_bw` test. In this test, the sender sends out a fixed number (equal to the window size) of back-to-back messages to the receiver and waits for a reply. The receiver replies only after receiving all these messages. This process is repeated for several iterations, and the bandwidth is calculated based on the elapsed time from the sending of the first message until the time the sender receives the reply from the receiver and the number of bytes sent by the sender. This bandwidth test aims to determine the maximum sustained data rate that can be achieved at the network level. Thus, non-blocking versions of MPI functions (*MPI_Isend* and *MPI_Irecv*) are used in the test.

[6]http://mvapich.cse.ohio-state.edu/benchmarks/

Figure 1 exhibits the latency and bandwidth achieved with the point-to-point microbenchmarks from OMB running on G5K (red line) and AWS (blue line). The x-axis corresponds to the data size in bytes, and y-axis shows the latency (Figure 1a) and bandwidth (Figure 1b). For small-size messages, the latency remains consistent and very similar on both platforms. When dealing with messages ranging from 2 to 4 MB or potentially larger, both platforms experience an increase in latency. Regarding bandwidth, a similar pattern emerges when transmitting small messages (up to 4 kB), with a bandwidth of approximately 600 MB/s. As the message size increases, the G5K bandwidth keeps increasing, reaching twice the bandwidth capacity compared to AWS.

B. Results with NPB

As explained in Section III, we carried out experiments with NPB applications using different numbers of nodes ($N = \{1, 2, 4, 8\}$) and varying the numbers of MPI processes per node ($P = \{1, 2, 4, 8, 16\}$). We analyzed the scalability of the applications on both platforms, ranging the total number of MPI processes from 2 to 128 ($np = \{2, 4, 8, 16, 32, 64, 128\}$). Since some combinations of $N \times P$ can result in the same number of MPI processes (np), we selected the combination that achieved the best result for each np value in each platform. This strategy guarantees that the performance comparison

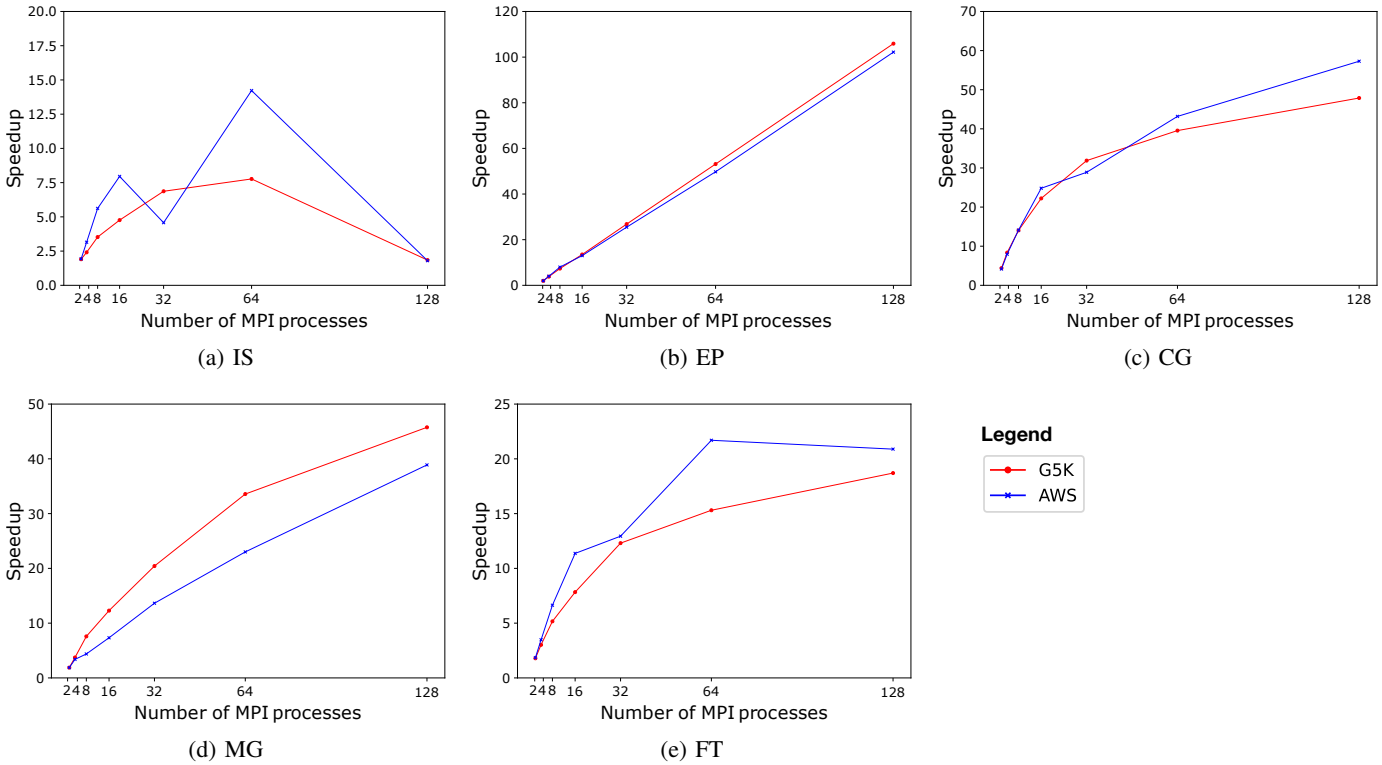


Fig. 2: Performance scalability (speedups) obtained with NPB kernels (class C).

considers each platform's best results. Next, we discuss the performance results observed by running the NPB kernel.

The IS kernel tests integer computation and communication performance. This benchmark combines point-to-point and collective communication operators. Figure 2a shows the speedup for AWS (blue line) and G5K (red line). Excepted by the configuration with 32 processes, speedup was superior in AWS. For instance, with 64 processors, the speedup measured with AWS reached 14.2 compared to 7.7 in the G5K – almost a twofold increase in performance. In both platforms, the speedup decays to less than 2 with 128 processors, demonstrating that the parallel execution does not scale well for such a large amount of processes. The point in the graph corresponding to 32 processes running on AWS coincides with a change in the number of nodes used in the experiments, *i.e.*, the addition of an extra node. Variations on the speedup curve with adding extra nodes were not observed in the Grid'5000 test results. As the IS benchmark did not reach the AWS bandwidth limit, AWS could still present a better performance for this kernel.

The EP kernel estimates the upper achievable limits for floating-point performance, representing CPU-intensive applications with minor or negligible communication between processes. For communication, the EP application does not use point-to-point communication operators like *MPI_Send()* and *MPI_Irecv()*, but just a few collective communication operators such as *MPI_BCast()* and reduction operators. Figure 2b shows the speedup for AWS and G5K running the EP kernel. As can be noticed, for both AWS and G5K, the speedup increases linearly with the number of processes. Since the network is not a bottleneck and has little influence on the application performance, extra processes can accelerate the test execution. G5K demonstrated a slightly superior performance compared with AWS as we increased the number of processes, but the difference is small (less than 4%). Although modern virtualization technologies add minimal overhead to CPU-bound applications, this difference could be caused by multi-tenant usage, *i.e.*, sharing the same physical server with other VMs. Even though the memory and the processing cores are segmented, the processor caches are still shared among multiple VMs. Moreover, VMs may also interfere with each other's performance due to network interface sharing [12].

CG is used to compute an approximation to the smallest eigenvalue of a large, sparse, symmetric positive definite matrix and incur irregular communication patterns using the *MPI_Send()* and *MPI_Irecv()* point-to-point communication operators. As demonstrated in Figure 2c, this kernel allowed good improvements in performance with the parallel versions, as indicated by the high speedup values. We still observed a similarity between the results obtained on G5K and AWS. For the scenarios with a higher number of processes, e.g., 64 and 128, AWS surpasses G5K. We monitored network traffic with the wireshark tool to investigate this phenomenon further. We could observe the higher network traffic on AWS and that MTU (Maximum Transmission Unit) was set to 1000 bytes on G5K and 9000 on AWS. The higher value in MTU

configuration aligns efficiently with the data size exchanged during internode communication in AWS. It contributes to the better performance even running on top of virtualized infrastructure.

MG provokes intensive data communication using many *MPI_send()* and *MPI_BCast()* calls. Figure 2d shows that MG speedup using AWS is inferior to the G5K speedup. Unlike IS and CG kernels, where AWS performed better than G5K, the communication pattern comprises large messages with the *MPI_BCast()* operation in MG. Thus, even with an MTU set to 9000, the data transferring cost through a network with lower bandwidth becomes noticeable in this test scenario. The better performance observed with G5K does not result only from its bare-metal infrastructure; it is a consequence of the higher network bandwidth.

FT performs 3D Fast Fourier Transform (FFT) methods, executing three one-dimensional FFTs, one for each dimension, and features long-distance communications. It represents a computationally intensive application that uses collective communication operators such as *MPI_BCast()* and *MPI_Alltoall()*. Speedup curves depicted in Figure 2e, obtained from the experimental results, resemble the observed behavior with the IS application but with better scalability with the increasing number of processes and higher speedup. IS and FT execute the same collective communication operators, but FT does not use point-to-point communication. In addition, the message size with FT is smaller than in MG. Consequently, the superior performance of AWS against G5K is due to the difference between the MTUs and the fact that the network bandwidth did not become a bottleneck neither for G5K nor for AWS.

V. Related Work

Recent advances in virtualization technology and investments in compute-optimized hardware done by public cloud providers caught the attention of the HPC community. Because of that, the performance evaluation of HPC applications in public cloud environments has been a recurring theme in the literature.

Application kernels from NPB have been used to evaluate public clouds' performance and cost efficiency. *Okada et al.* [13] compared the performance achieved by LU and SP kernels when running on GCP and on a private cloud implemented with OpenStack. The results showed that the performance of HPC applications can be affected by VM allocation in physical hardware and by the use of hyper-threading technology. *Maliszewski et al.* [14] evaluated the performance of all NPB kernels when running on three different instances/network interconnections of Microsoft Azure. They demonstrated that the interconnection plays a crucial role in speeding up the kernels. *Roloff et al.* [15] performed a more complete evaluation, comparing the performance and cost efficiency of HPC applications running on three public clouds against the same applications running on a traditional on-premise cluster. They concluded that the cloud can provide a viable platform for running HPC applications despite some disadvantages in the deployment. Moreover, results obtained with kernels from NPB showed that HPC applications can run efficiently on the cloud, but care must be taken when choosing the provider, as the differences between them are significant. Big Data benchmarks have also been used to evaluate the feasibility of clouds for HPC. *Salaria et al.* [16] presented a comparative study of the performance of representative big data benchmarks, and HPC benchmarks running on supercomputer and cloud. Experiments using C4 compute-optimized Amazon EC2 instances revealed that these instances are feasible for scientific computing and its applications in simulations, modeling, and analysis.

Other researchers have focused on the use of private cloud solutions for HPC. *Tomić et al.* [17] evaluated the HPL and NAMD benchmarks on the HPE OpenStack testbed and NAMD benchmarks on a supercomputer located at Rijeka University Supercomputing Center. Their results revealed two major bottlenecks: the throughput of the interconnect and cloud orchestration layer, among others responsible for managing the communication between Cloud instances. *Lit et al.* [18] compared the performance of Open Source Clouds platforms such as Nimbus, Open Nebula and OpenStack for HPC according to the HPC Challenge (HPCC) benchmark suite. They concluded that OpenStack achieved the best performance. *Gupta et al.* [19] evaluated the performance-cost tradeoffs of running a HPC application using NPB and two real-world applications with up to 256 cores. They considered two on-premise HPC clusters and one private cloud that uses KVM for virtualization. They found that clouds are more cost-effective for low communication-intensive applications such as embarrassingly parallel and tree-structured computations, and HPC-optimized clusters are better for the rest.

Our work is complementary to the aforementioned works. We concentrated our analysis on a well-known cloud provider (AWS) that offers instances optimized for HPC. To the best of our knowledge, our study is the only one that evaluates the performance of MPI-based NPB kernels running on the most recent compute-optimized VM instances from AWS (c5n), contrasting them against the results obtained with the same kernels running on G5K.

VI. Conclusion

Access to HPC clusters is limited to the well-established research and computing centers. Emerging research groups, startups, and practitioners not well provided for financially have found a barrier in advancing their HPC-related research. The substantial costs and complexities associated with deploying and maintaining on-premise HPC infrastructures have created barriers for these entities. A low-cost alternative is to adopt on-demand pay-per-use infrastructures such as those provided by cloud computing for running HPC applications. However, a fundamental question arises: Can these shared and virtualized infrastructures deliver performance comparable to that of conventional HPC clusters? To answer this question, we presented a comparative study involving the Grid'5000 cluster (on-premise) and AWS public cloud (on-demand) as platforms

to execute realistic MPI applications implemented by the NAS Parallel Benchmark.

Our investigations were framed by extensive experiments, detailed in Section III, encompassing varying numbers of nodes and MPI processes per node. We carefully selected similar infrastructure and hardware components to AWS and G5K for a fair comparison. The results show that CPU-bound applications achieve similar results on both platforms. The virtualized infrastructure delivered a slightly small performance, but the differences observed in speedup with AWS and G5K were less than 4%.

Interestingly, AWS performed better in scenarios characterized by workloads with moderate communication and medium-sized messages. This performance gain in the virtualized infrastructure can be attributed to AWS's MTU configuration, which is set at a higher value, thereby aligning efficiently with the size of data exchanged between nodes. However, an inherent limitation of cloud-based infrastructure is its bandwidth capacity. As the workload and message sizes increase, the G5K bandwidth reaches twice the capacity of AWS. In scenarios with communication-bound applications, AWS may be impacted by the limited network bandwidth compared to G5K.

Our comparative study has depicted nuanced performance behaviors between the G5K cluster and AWS infrastructure, showcasing areas of superiority in specific scenarios for each platform. However, it is worth mentioning that the performance results with the virtualized infrastructure are similar in most cases. There are no significant performance penalties in CPU-bound applications. The main differences appear with the communication-bound applications.

REFERENCES

[1] R. Buyya and B. Varghese, "Next Generation Cloud Computing: New Trends and Research Directions," *Future Generation Computer Systems*, vol. 79, no. 3, pp. 849–861, sep 2017. [Online]. Available: https://doi.org/10.1016/j.future.2017.09.020

[2] "The nist definition of cloud computing," Gaithersburg, USA, Tech. Rep., 2011.

[3] R. Buyya, S. N. Srirama, G. Casale, R. Calheiros, Y. Simmhan, B. Varghese, E. Gelenbe, B. Javadi, L. M. Vaquero, M. A. S. Netto, A. N. Toosi, M. A. Rodriguez, I. M. Llorente, S. D. C. D. Vimercati, P. Samarati, D. Milojicic, C. Varela, R. Bahsoon, M. D. D. Assuncao, O. Rana, W. Zhou, H. Jin, W. Gentzsch, A. Y. Zomaya, and H. Shen, "A Manifesto for Future Generation Cloud Computing: Research Directions for the Next Decade," *ACM Computing Surveys*, vol. 51, no. 5, nov 2019. [Online]. Available: https://doi.org/10.1145/3241737

[4] R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg, and I. Brandic, "Cloud Computing and Emerging IT Platforms: Vision, Hype, and Reality for Delivering Computing as the 5th Utility," *Future Generation Computer Systems*, vol. 25, no. 6, jun 2009. [Online]. Available: https://doi.org/10.1016/j.future.2008.12.001

[5] P. Browne, X. Lu, J. Mills, and D. Panda, "OpenStack and Network Fabrics," in *The Crossroads of Cloud and HPC: Exploring OpenStack Cloud Computing for Scientific Workloads*. CreateSpace Independent Publishing Platform, oct 2017, pp. 14–22.

[6] B. Bethwaite and X. Lu, "OpenStack and Virtualised HPC," in *The Crossroads of Cloud and HPC: Exploring OpenStack Cloud Computing for Scientific Workloads*. CreateSpace Independent Publishing Platform, oct 2017, pp. 2–12.

[7] M. A. S. Netto, R. N. Calheiros, E. R. Rodrigues, R. L. F. Cunha, and R. Buyya, "HPC Cloud for Scientific and Business Applications: Taxonomy, Vision, and Research Challenges," *ACM Computing Surveys*, vol. 51, no. 8, jan 2018. [Online]. Available: https://doi.org/10.1145/3150224

[8] D. Balouek, A. Carpen Amarie, G. Charrier, F. Desprez, E. Jeannot, E. Jeanvoine, A. Lèbre, D. Margery, N. Niclausse, L. Nussbaum, O. Richard, C. Pérez, F. Quesnel, C. Rohr, and L. Sarzyniec, "Adding virtualization capabilities to the Grid'5000 testbed," in *Cloud Computing and Services Science*, ser. Communications in Computer and Information Science. Springer International Publishing, 2013, vol. 367, pp. 3–20.

[9] 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.

[10] V. Munhoz and M. Castro, "Hpc@cloud: A provider-agnostic software framework for enabling hpc in public cloud platforms," in *Simpósio em Sistemas Computacionais de Alto Desempenho (WSCAD)*. Sociedade Brasileira de Computação (SBC), 10 2022, pp. 157–168.

[11] J. F. Uller, J. V. Souto, P. H. Penna, M. Castro, H. Freitas, and J.-F. Méhaut, "Enhancing programmability in noc-based lightweight manycore processors with a portable mpi library," in *Simpósio em Sistemas Computacionais de Alto Desempenho (WSCAD)*. Sociedade Brasileira de Computação (SBC), 10 2020, pp. 155–166. [Online]. Available: https://sol.sbc.org.br/index.php/wscad/article/view/14066

[12] J. R. Brunetta and E. Borin, "Selecting efficient cloud resources for hpc workloads," in *Proceedings of the 12th IEEE/ACM International Conference on Utility and Cloud Computing*, ser. UCC'19. Association for Computing Machinery, 2019, p. 155–164.

[13] T. K. Okada, A. Goldman, and G. G. H. Cavalheiro, "Using nas parallel benchmarks to evaluate hpc performance in clouds," in *2016 IEEE 15th International Symposium on Network Computing and Applications (NCA)*, 2016, pp. 27–30.

[14] A. M. Maliszewski, E. Roloff, E. D. Carreño, D. Griebler, L. P. Gaspary, and P. O. A. Navaux, "Performance and cost-aware hpc in clouds: A network interconnection assessment," in *2020 IEEE Symposium on Computers and Communications (ISCC)*, 2020, pp. 1–6.

[15] E. Roloff, M. Diener, A. Carissimi, and P. O. A. Navaux, "High performance computing in the cloud: Deployment, performance and cost efficiency," in *4th IEEE International Conference on Cloud Computing Technology and Science Proceedings*, 2012, pp. 371–378.

[16] S. Salaria, K. Brown, H. Jitsumoto, and S. Matsuoka, "Evaluation of hpc-big data applications using cloud platforms," in *2017 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID)*, 2017, pp. 1053–1061.

[17] D. Tomić, Z. Car, and D. Ogrizović, "Running hpc applications on many million cores cloud," in *2017 40th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, 2017, pp. 209–214.

[18] C. Li, J. Xie, and X. Zhang, "Performance evaluation based on open source cloud platforms for high performance computing," in *2013 6th International Conference on Intelligent Networks and Intelligent Systems (ICINIS)*, 2013, pp. 90–94.

[19] A. Gupta and D. Milojicic, "Evaluation of hpc applications on cloud," in *2011 Sixth Open Cirrus Summit*, 2011, pp. 22–26.