An Analysis of Performance Variability in AWS Virtual Machines

Miguel de Lima[1], **Luan Teylo**[2], **Lúcia Drummond**[1]

[1]Instituto de Computação – Universidade Federal Fluminense (UFF)

[2]Centre Inria de l'université de Bordeaux - INRIA Bordeaux

Abstract. *Cloud computing platforms are essential for a wide range of applications, including High-Performance Computing (HPC) and artificial intelligence. However, the performance variability of virtual machines (VMs) in these shared environments presents significant challenges. This paper provides an extensive month-long analysis of the performance variability of C family VMs on Amazon Web Services (AWS) across two regions (`us-east-1` and `sa-east-1`), various instance generations, and market types. Our findings indicate that Graviton processors (c6g.12xlarge and c7g.12xlarge) exhibit minimal performance variability and high cost-effectiveness, with the c7g.12xlarge instance, in particular, offering significantly reduced execution times and lower costs. Intel and AMD instances, while showing performance improvements from generation c6 to c7, exhibited up to 20% variability.*

1. Introduction

Cloud computing platforms have become a viable option for numerous domains and applications, ranging from web hosting and e-commerce platforms to big data and analytics applications. Recently, cloud providers have intensified efforts to attract the High-Performance Computing (HPC) community, with major players like Amazon Web Services (AWS) demonstrating successful executions of large and complex simulations in the cloud. Concurrently, with the increasing popularity of AI applications, cloud platforms have become the top choice for executing these workloads, both during the training phase and model inference.

A critical distinction between the HPC and AI workloads compared to other applications is the necessity for guaranteed performance capabilities throughout execution. These applications demand substantial computational resources, including processing power and memory, and typically run for extended periods, ranging from hours to days. In the cloud model, physical resources are shared among Virtual Machines (VMs) from different clients, running various applications. This sharing can lead to potential interference due to host resource contention. Given the recent trend of adopting cloud resources for increasingly demanding applications, an important question arises: "Is the performance always consistent in the cloud?"

To address this question, we conducted a series of evaluations on Amazon Elastic Compute Cloud (EC2), a platform that provides scalable computing capacity in the form of VMs. In EC2, these VMs, or instances, are categorized into families, each suited for different usage purposes. For example, Compute Optimized instances (family C) are ideal for compute-bound applications, while instances from family M are Memory Optimized, making them perfect for workloads that process large datasets in memory. Each family

of instances includes various *types*[1], which determine the instance generation and CPU architecture. Besides the type, an instance also comes in different sizes, which define the computational resources such as the number of CPUs and the amount of memory. For instance, requesting a c7g.12xlarge instance means selecting the latest generation (version 7) of compute-optimized instances, powered by the AWS Graviton3 processor, which has 48 Virtual CPUs (VCPUs) and 96 GiB of memory.

In AWS, a VM can be allocated based on different markets, which determine not only their price but also their reliability. AWS has three main markets: the on-demand market, where users are guaranteed that the instance will be available for the entire requested duration; the spot market, where instances are offered at a steep discount (up to 90% cheaper compared to an on-demand instance of the same type) but can be terminated by the provider at any time; and the Reserved and Savings Plans, which offer lower prices in exchange for a commitment to a consistent amount of usage (measured in $/hour) for a one- or three-year term. In this work, one of our focuses is on the performance variability between the on-demand and spot markets. According to AWS, spot instances emerged as a solution to handle idle resources in the cloud. Instead of letting resources remain idle, they can be offered to users at a cheaper price. However, if demand in the on-demand market increases, these spot resources may be reclaimed by the provider.

While much of the related literature has evaluated the performance of EC2 VMs, to the best of our knowledge, no one has investigated performance variability considering the factors we present in this paper. Specifically, we conducted a study to verify and answer the following questions:

- **Q1:** Considering the same VM types, is the performance consistent across different regions?
- **Q2:** How does the evolution of VM generations impact performance? Is there a considerable gain in terms of performance and price with new generations compared to older ones?
- **Q3:** Is the performance of the Spot market consistent with that observed in the on-demand market?
- **Q4:** Is there significant performance variability throughout the days?

In this study, we considered eight distinct VM types from the C family, selected based on the criteria presented in Section 3. To measure performance, we used a benchmark application available in the NAS benchmark toolkit [Bailey et al. 1991]: the Embarrassingly Parallel (EP) application (more details in Section 3.2). All tests were executed in two distinct EC2 regions, `us-east-1` (USA) and `sa-east-1` (Brazil). In total, 16.117 executions were conducted over a period of one month (from June 23rd to July 23rd). The main contributions of the paper are:

- A study on the performance of a set of EC2 VMs from the Compute Optimized family, focusing on understanding their performance variability across two distinct regions, different processor architectures, and their evolution over generations.
- A set of observations, or lessons learned, extracted from the performance evaluation that can guide cloud users in the VM selection process.

[1]The complete list of instances offered by AWS can be seen at `https://aws.amazon.com/ec2/instance-types/`.

- A codebase that includes tools for collecting and analyzing the performance of VMs in AWS.

2. Related Works

Several works in the related literature have evaluated the performance variability of major cloud providers, focusing not only on VMs and CPU performance but also on other services, such as I/O and databases [Iosup et al. 2011, Jackson et al. 2010, Ericson et al. 2017]. However, when examining machine performance in general, the related literature often looks at VMs with low resources and does not focus on heavy workloads. For instance, in [Iosup et al. 2011], the authors presented a large study comprising two months of observations that considered several components of AWS, such as Amazon SimpleDB, S3, and EC2. For the VMs, only the deployment time, i.e., the time it takes to start an instance in EC2, was considered, and only for the instance *m1.small*, which has 1 vCPU and 1.7 GB of memory. While this metric is important for some categories of applications, such as auto-scaling systems, disaster recovery, and development/testing environments, it is less critical for long-running computational tasks.

In [Ericson et al. 2017], the authors compared the performance variability of CPU, RAM, and disk. They conducted a series of tests (the quantity and period of tests are not cited) considering five cloud providers, including AWS, and compared the variability to a baseline case executed in an on-premise cluster. One of their results shows that AWS had the highest performance variability in comparison to the other cloud providers. However, it is not clear in the paper what type of machine was used in AWS (or in the other cloud providers). Moreover, the authors focused on machines running Microsoft Server, which is usually not the operating system selected for heavy workload applications. Nonetheless, they show that when looking at the VMs' subsystems (memory, CPU, and disk), the CPU and memory account for a significant part of the variability seen in performance.

Although these works present an interesting comparison between cloud providers, they do not focus on heavy workloads. On the other hand, some studies focus exclusively on HPC workloads. This is the case in [Jackson et al. 2010], where the authors evaluated on-premise environments and EC2 using the NESC benchmarking framework [Oliker et al. 2004]. According to the authors, this benchmarking represents the workload of typical HPC applications. Their main focus in this evaluation was on MPI applications and the impact of virtualization on the communication patterns of the applications. Their results show that EC2 had the worst performance in terms of latency and execution time when compared to other on-premise environments (up to 20x slower). However, unlike the study presented in this paper, their main focus was not on the variability of VM performance but on the absolute performance when executing this set of applications compared to on-premise systems. Additionally, unlike our work, the authors studied an EC2 instance that was not optimized for computationally heavy applications: they used only m1.large VMs, which have 2 vCPUs and 7.5 GB of memory.

Applications from the NPB suite have been used to evaluate public cloud performance in several works [Munhoz et al. 2023, Ferrari et al. 2024, Dancheva et al. 2024]. In [Munhoz et al. 2023], the authors presented a study where three kernels of the NPB benchmark were used to compare the performance of a cluster of VMs running in EC2 with the performance of an on-premise cluster of Grid5000. Similar to our work, the

authors in [Munhoz et al. 2023] also used the EP kernel. They demonstrated that, in comparison to the on-premise cluster, this kernel showed 4% less performance, which the authors attributed to the multi-tenant usage in the cloud. However, they focused only on one type of instance (c5n.4xlarge), and their comparison mainly addressed the performance impact when running MPI applications using different numbers of processors. In [Ferrari et al. 2024], the NPB kernel was used to compare the performance of burstable and non-burstable VMs. Their tests covered a period of 24 hours and focused on small-size VMs from the T and M families. Finally, in [Dancheva et al. 2024], the authors used the OSU benchmark and several kernels of the NAS to compare the performance of instances in EC2 to an on-premise cluster called Beskow. To manage the EC2 instances, they used Parallel Cluster, and their tests focused on the performance of collective MPI functions and the scaling of distributed applications when running in the cloud. As in our work, the authors also focused on C family instances; specifically, they conducted their tests using 40 VMs of c5n.18xlarge and concluded that the performance was very similar to the on-premise cluster.

To the best of our knowledge, in the HPC context, no previous work has focused on evaluating the performance of EC2 instances considering the factors we are studying. Notably, the variability of performance across different regions and markets. Moreover, we also evaluate the performance of the VMs in relation to their generation.

3. Methodology

In this section, we describe the experimental methodology we applied for this study and present the main components related to this study: the platform (Section 3.1), the benchmark tools and used parameters (Section 3.2), and the execution protocol (Section 3.3). Note that all scripts used for our tests, as well as the results, are available and documented at `https://github.com/migueluff/awsbench`.

3.1. Experimental Environment

For the performance evaluation in this paper, we established specific criteria to select instances, ensuring their similarity in computational characteristics, particularly in terms of the number of cores and memory. The focus was on instances suitable for medium-scale CPU-bound applications, with an aim to include the latest three generations of instances (v5, v6, and v7) featuring Intel, AMD, and Graviton CPUs. Based on these criteria, eight instance types were selected, and their main characteristics are summarized in Table 1.

As shown in Table 1, the VMs with Graviton processors are not available in generation v5 (only v6 and v7). This is because Amazon only introduced this processor in 2018, while the v5 instances date back to 2017. Another important consideration related to the selection of instances was their availability in different markets and regions. Specifically, it was considered those VMs available in the two regions regarded in this study (`us-east-1` and `sa-east-1`), which also be allocated either in the spot or on-demand markets. However, we faced some difficulties meeting this last criterion since instance *c7a.12xlarge* were not available in the `sa-east-1` region at the time we conducted our tests. Nonetheless, we decided to keep them on the list as it would allow us to evaluate the evolution in terms of performance between the AMD generations.

Finally, it is important to note that while Intel and AMD instances use hyper-threading, Graviton instances do not, meaning each vCPU corresponds to a physical core.

Therefore, for Intel and AMD instances, 24 physical cores are presented, while Graviton instances have 48 physical cores. As it will be detail in Section 3.3, to make a fair comparison between these instances, 24 threads are used in all our tests.

Tabela 1. Set of instances used in this study. All instances have 48 vCPUs and 96 GiB of memory.

Instance Type	CPU Model	On-demand Price USD	
		us-east-1	sa-east-1
c5.12xlarge	2nd Intel Xeon (Cascade Lake 8275CL)	2.04	3.14
c6i.12xlarge	3rd Intel Xeon (Ice Lake 8375C)	2.04	3.14
c7i.12xlarge	4th Intel Xeon (Sapphire Rapids 8488C)	2.14	3.30
c5a.12xlarge	2nd AMD EPYC (7R32)	1.84	2.83
c6a.12xlarge	3rd AMD EPYC (7R13)	1.83	2.82
c7a.12xlarge	4th AMD EPYC (9R14)	2.46	–
c6g.12xlarge	AWS Graviton2 (Arm)	1.63	2.51
c7g.12xlarge	AWS Graviton3 (Arm)	1.74	2.66

3.2. Benchmark Tools

For the evaluation, we used the NAS Parallel Benchmarks toolkit. NAS is a well-known set of programs that have been used in several works to evaluate the performance of parallel supercomputers [Hosseini et al. 2024, Subhlok et al. 2002, Bakhtiarifard et al. 2024]. The toolkit includes five kernels that mimic real applications derived from the computational fluid dynamics domain. Each application is associated with different problem classes that determine how computationally heavy the application is, with the smallest tests being from class S (small for quick test purposes) and the largest tests being from classes D, E, and F. For this work, we selected the Embarrassingly Parallel (EP) kernel (class D).

According to [Bailey et al. 1991], the EP Kernel "[...] provides an estimate of the upper achievable limits for floating point performance without significant inter-processor communication". The benchmark is a CPU-bound application and uses OpenMP for parallelism. For our tests, this application (considering class D) presents a good trade-off between execution time and problem complexity. Moreover, it has been used in several works on the evaluation of cloud instances [Ferrari et al. 2024, Dancheva et al. 2024].

For the execution, we used an EC2 image with GNU Linux (Ubuntu 24.04 distribution). In this image, all applications were compiled using GFORTRAN version 13.2.0. For the EP Kernel, we used OpenMP version 4.5 and bound the threads to the cores using the following environment variables: `OMP_PLACES=cores` and `OMP_PROC_BIND=spread`. Thread binding is important because it ensures that threads are pinned to specific cores, reducing context switching and cache misses, thus improving performance consistency and minimizing variability. As mentioned before, we used 24 threads in all EP executions. This, combined with thread binding, guarantees a fair comparison among the distinct VMs since the selected instances have between 24 and 48 physical cores. To achieve this, we set up the environment variable `OMP_NUM_THREADS=24` before each EP execution.

3.3. Execution Protocol

To collect data consistently, we defined an execution protocol where tests were dispatched at fixed time intervals. Each test was executed sequentially, meaning there were no parallel executions. Once an execution finished, the benchmark metrics, execution time and Millions of Operations Per Second (MOPs), were saved in a CSV file alongside the VM type, region, market, and the timestamps when the test started and finished. To avoid bias from repeating the same test sequences, the order of tests in a sequence was randomly defined.

To execute this protocol, we developed a tool called AWSBENCH. The tool receives a list of VMs to be evaluated (described in Table 1), the benchmark tool, and the time interval, and then creates a list of all the tests that need to be executed. In this context, a test is defined as a tuple comprising the region, instance type, market, and Benchmark Kernel. The tool then starts to execute the tests at each time interval, one by one in sequence and non-concurrently. For each test case, AWSBENCH allocates the VM in the specified region and market, executes the benchmark n consecutive times (on the same VM), turns off the machine, and then moves to the next randomly selected test case in the list. This process is repeated for all tests in the list, which consist of all possible combinations for both markets and regions. In total, 1.400 test cases are executed in each time interval. For the results presented in this paper, we selected an interval of 2 hours, with five consecutive executions per test ($n = 5$). In this way, our test analysis covers a wide range of time periods throughout the day, across different days and weeks.

4. Results

In this section, we present the results and describe the data analyses conducted with the collected performance metrics to answer the research questions of this study. All the analyses are available in the repository[2], as well as the dataset collected throughout this work.

4.1. Performance Across the Different Regions

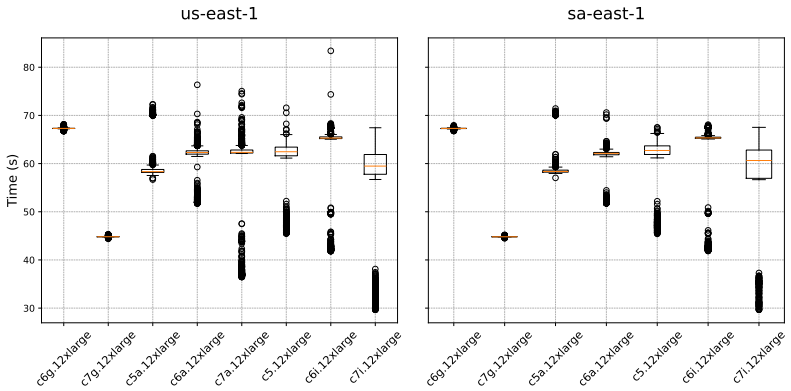

Figura 1. Boxplot of execution times for the evaluated instances across `us-east-1` and `sa-east-1`. Note that the *c7a.12xlarge* instance is not available in `sa-east-1`.

[2]https://github.com/migueluff/awsbench

To address question Q1, we first analyzed the distribution of the execution time of the tests. This distribution is presented as a boxplot in Figure 1, where we can observe some interesting performance trends. Firstly, the boxplot reveals that the overall performance of the evaluated VMs is consistent across both regions. In other words, the median, and interquartile range are nearly identical, which indicates that there is no significant difference in performance between these regions.

Secondly, we see that the Graviton processors (*c6g.12xlarge* and *c7g.12xlarge*) show no outliers in the boxplot, while Intel and AMD machines have several outliers throughout the evaluation. Moreover, the instance *c7g.12xlarge* demonstrated better performance, with an average execution time 22.4% and 19.55% shorter compared to the AMD and Intel machines of the same generation, respectively. This performance gain becomes even more interesting when we add the cost dimension to our analysis: as presented in Table 1, the *c7g.12xlarge* has the lowest monetary cost compared to the VMs of its generation in both regions.

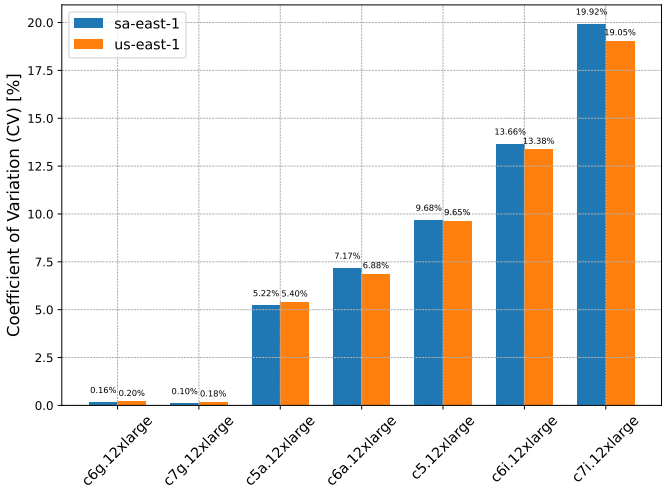

Figura 2. Coefficient of Variation (CV) for each VM instance type in `sa-east-1` and `us-east-1` regions.

These results indicate that the performance variability is directly related to the architecture of the processor used in the VMs and that the region has no impact on it. To assess this, we calculated the Coefficient of Variation (CV) for both performance metrics (Execution Time and MOPs). The CV is a statistical measure that quantifies the relative variability of data, defined as the ratio of the standard deviation to the mean and expressed as a percentage. Lower CV values indicate more consistent performance, while higher values suggest greater variability.

For this analysis, we computed the mean and standard deviation by grouping the executions per region and instance type, without distinguishing between markets. Figure 2 presents the CV values for the evaluated VMs in both regions, considering the Execution Time metric. The results for the MOPs are similar to those presented in Figure 2. Thus, due to space limitations, these results are supplemented and can be viewed in the Git repository.

As we can see, Graviton processors in fact demonstrate almost no performance

variability (less than 0.3% variability for both regions), while Intel machines had the highest CV values (up to 20%). When comparing the variation between regions, we observe that the performance variability across the studied instance types is nearly the same in the `us-east-1` and `sa-east-1` regions. Therefore, for these two regions, we saw no impact on the degree of variability of the instances for the conducted tests.

> The performance variability of VMs appears to be primarily influenced by their processor architecture. Graviton processors (Arm) show almost no variability, while Intel processors exhibit the highest variability. No significant impact was observed from regional differences.

4.2. Performance Across Instance Generations

Next, we evaluate the variability of VM performance across different generations (Q2). The objective of this analysis is to determine if there are always gains in terms of performance and cost when a new generation of VM is released. For this analysis, we focus on the data collected in the `us-east-1` region, as it contains data for all eight instance types we considered. To study the relationship between price and performance, we used the prices of the On-demand market, which did not change throughout our tests, and for the performance metrics, we used the average Mops.

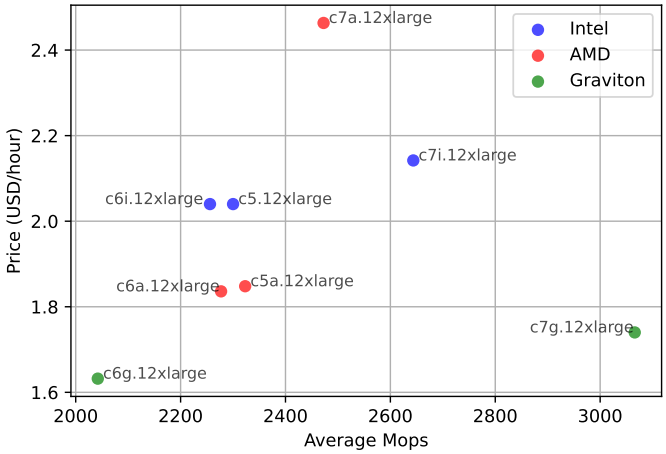

Figura 3. Comparison of VM prices (y-axis) and their average performance (x-axis) across different generations. On the x-axis, higher values indicate better performance.

Figure 3 shows the relationship between the price of the VMs (y-axis) and their average performance (x-axis). As we can see, from generation c5 to c6 of the Intel and AMD machines, there is no increase in performance and almost no difference in price. However, from generation c6 to c7, we observe an increase of 12.83% and 8.30% in the average performance of these instances, respectively. Nevertheless, for both architectures, there is an increase in the price of the latest generation machines: an 24.66% increment from *c6a.12xlarge* to *c7a.12xlarge* and an 7.89% increment from *c6i.12xlarge* to *c7i.12xlarge*.

The Graviton machines show the best performance improvement from one generation to another: from *c6g.12xlarge* to *c7g.12xlarge*, there is an increment of 33.39%

in average Mops, and an increase of only 7.16% in monetary cost. Moreover, the *c7g.12xlarge* presents the best tradeoff between cost and performance when compared to the other VMs. For instance, in comparison to the *c7i.12xlarge*, this machine presented 15.12% more Mops (3065.92 Mops against 2602.39), at a cost 30.08% lower (1.26$ against 1.64$).

For applications with similar characteristics to the EP Kernel, the latest generation Graviton machines offer the best performance-to-cost ratio, while the latest generation AMD machines provide the worst.

4.3. The Variability Across Markets

In this analysis, we investigate whether there is a performance difference between VMs running in the Spot market and those running in the On-demand market (Q3). We focus the analysis on the data collected from `us-east-1`, which contains all VMs we study, and we divide the dataset into Spot and On-demand VMs. The bar chart in Figure 4 presents the CV, considering the Mops metric, for each type of VM.

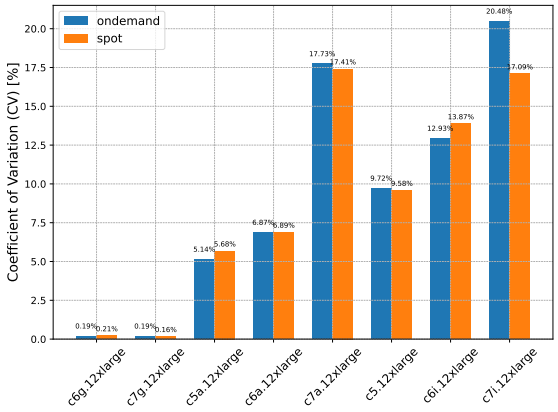

Figura 4. Coefficient of Variation (CV) of performance (measured in Mops) for each type of VM in the Spot and On-demand markets in the `us-east-1` region.

As seen, in general, the variability of the performance of the machines is similar regardless of the market and follows the same trends presented in Section 4.1. The biggest difference between markets was reported for instance *c7i.12xlarge*, where we saw 3.39% more variability in the On-demand market compared to the Spot one. However, that difference can be explained by the number of tests executed successfully in the On-demand market for this VM, 4971 tests in total, compared to only 4359 tests executed successfully in the Spot market. This happens because while in the On-demand market the VM *c7i.12xlarge* was always available, in the Spot market that was not the case: a total of 11.47% of requests to VM spots were denied during our tests due to the lack of availability in the Spot market.

Apart from the variability, Figure 4 also demonstrates that the evaluated VMs have similar performance in both markets. These results confirm the guarantees offered by AWS for the Spot market. According to the company, the only differences between

the markets are the availability of the Spot VMs and the price, but not the performance of the machines. Note that, Figure 4 also includes VM *c7a.12xlarge*, as it is available in the `us-east-1` but not in `sa-east-1`. Once again, it confirms that the latest generations of Intel and AMD VMs presented high variability in performance throughout our tests.

In terms of price, we see in Figure 5 that the VMs offered in the Spot market are significantly cheaper than those in the On-demand market: the VM *c5.12xlarge* shows the biggest difference (60.80% cheaper compared to the On-demand one), while the VM *c7i.12xlarge* shows the smallest difference (53.60%).

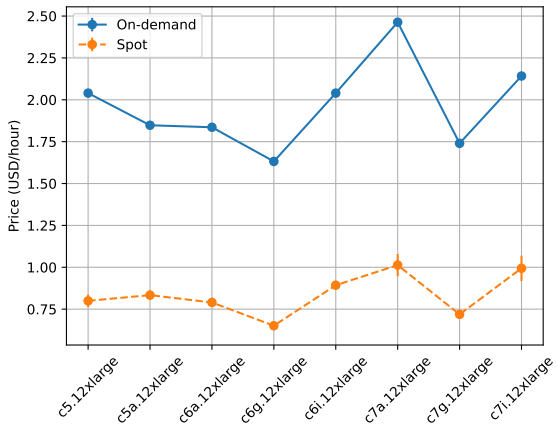

Figura 5. Average price comparison between On-demand and Spot VMs in the `us-east-1` region.

For applications that can tolerate faults, using Spot VMs offers significant cost savings without compromising performance. However, it is necessary to consider the availability of VMs in the Spot market, as some instances may experience higher denial rates than others.

4.4. The Variability Throughout the Day

Next, we analyzed the performance variability of the VMs throughout the days (Q4). Figure 6 presents a heatmap showing the performance of the VMs over the evaluated days in `us-east-1`. To generate this heatmap, the average Mops per day of each instance type was computed and normalized. Thus, the color shift to 1.0 represents the peak performance of the instance throughout the tests. The same graph for `sa-east-1` can be seen in the supplementary material.

As can be seen, there is no performance trend per day. In other words, we do not see a column in the heatmap where all VMs presented either low or high performance. On the other hand, we see trends per instance type. For instance, we can see that the Graviton instances (*c6a.12xlarge* and *c7a.12xlarge*) showed no variation in performance over the days during our tests. Once again, the most variation in performance throughout the days can be seen in instances *c6i.12xlarge*, *c7a.12xlarge*, and *c7i.12xlarge*.

These results suggest that, during our tests, the resources in AWS did not face any event that would globally affect the performance of the VMs in a region. Moreover, since the AMD and Intel instances presented the highest variation in performance in

all our previous analyses, the behavior seen in Figure 6 can once again be attributed to characteristics of the VMs themselves (such as their CPU architecture).

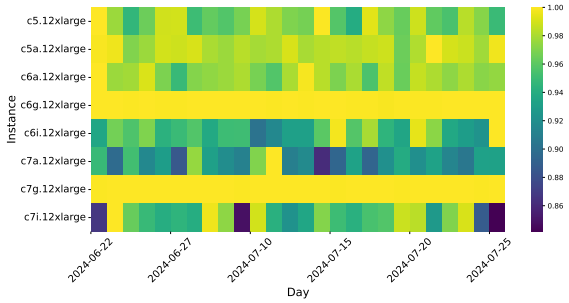

Figura 6. Heatmap showing the average Mops per day for each instance type in `us-east-1`

The absence of a daily performance trend suggests stable resource availability in AWS. However, specific instance types, particularly AMD and Intel instances, showed notable performance variation throughout the days, likely due to inherent characteristics of their CPU architectures rather than external factors affecting the entire region.

5. Conclusion and Future Work

This study provides a comprehensive analysis of one month of performance variability of virtual machines in the C family on AWS across two regions (`us-east-1` and `sa-east-1`), different instance generations, and markets. Our data reveal that VM performance is consistent between `us-east-1` and `sa-east-1`, with Graviton processors (*c6g.12xlarge* and *c7g.12xlarge*) demonstrating almost no variability in their performance compared to Intel and AMD machines. The *c7g.12xlarge* instance, in particular, showed significantly shorter execution times and lower costs, making it the most cost-effective option for applications with characteristics similar to the NAS EP kernel. While Intel and AMD machines showed performance improvements from generation c6 to c7, these gains were accompanied by increased costs and higher performance variability. Additionally, performance in Spot and On-demand markets was found to be similar, with Spot instances offering substantial cost savings, making them an attractive option for applications that can tolerate failures.

In addition to our findings, we developed AWSBENCH, a tool that automatically manages the execution of performance probes in AWS. All data presented in this paper, as well as supplementary material with all graphs that were suppressed due to space limitations, are made publicly available. In the near future, we will extend this analysis to other instance families and regions, further exploring the impact of different workload types, such as distributed and I/O-bound applications, on performance variability. Moreover, we aim to extend our investigation to examine the impact of communication overhead in cloud environments by using MPI kernels available in the NAS benchmark. Additionally, we will extend this study to instances that are optimized for network-intensive workloads, such as the c6gn and c7gn VMs.

Acknowledgments

This research is supported by project CNPq/AWS 421828/2022-6, Brazil.

Referências

Bailey, D. H., Barszcz, E., Barton, J. T., Browning, D. S., Carter, R. L., Dagum, L., Fatoohi, R. A., Frederickson, P. O., Lasinski, T. A., Schreiber, R. S., et al. (1991). The nas parallel benchmarks. *The International Journal of Supercomputing Applications*, 5(3):63–73.

Bakhtiarifard, P., Igel, C., and Selvan, R. (2024). Ec-nas: Energy consumption aware tabular benchmarks for neural architecture search. In *ICASSP 2024-2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5660–5664. IEEE.

Dancheva, T., Alonso, U., and Barton, M. (2024). Cloud benchmarking and performance analysis of an hpc application in amazon ec2. *Cluster Computing*, 27(2):2273–2290.

Ericson, J., Mohammadian, M., and Santana, F. (2017). Analysis of performance variability in public cloud computing. In *2017 IEEE International Conference on Information Reuse and Integration (IRI)*, pages 308–314. IEEE.

Ferrari, G. C. F., Castro, M., et al. (2024). Comparing burstable and on-demand aws ec2 instances using nas parallel benchmarks. In *Anais da XXIV Escola Regional de Alto Desempenho da Região Sul*, pages 61–64. SBC.

Hosseini, S. S., Chuen, A. M., and Chan, W. M. (2024). Computational aerodynamics study of the lift+ cruise vtol concept vehicle components. In *Transformative Vertical Flight (TVF) Meeting*.

Iosup, A., Yigitbasi, N., and Epema, D. (2011). On the performance variability of production cloud services. In *2011 11th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*, pages 104–113. IEEE.

Jackson, K. R., Ramakrishnan, L., Muriki, K., Canon, S., Cholia, S., Shalf, J., Wasserman, H. J., and Wright, N. J. (2010). Performance analysis of high performance computing applications on the amazon web services cloud. In *2010 IEEE second international conference on cloud computing technology and science*, pages 159–168. IEEE.

Munhoz, V., Bonfils, A., Castro, M., and Mendizabal, O. (2023). A performance comparison of hpc workloads on traditional and cloud-based hpc clusters. In *2023 International Symposium on Computer Architecture and High Performance Computing Workshops (SBAC-PADW)*, pages 108–114. IEEE.

Oliker, L., Canning, A., Carter, J., Shalf, J., and Ethier, S. (2004). Scientific computations on modern parallel vector systems. In *SC'04: Proceedings of the 2004 ACM/IEEE Conference on Supercomputing*, pages 10–10. IEEE.

Subhlok, J., Venkataramaiah, S., and Singh, A. (2002). Characterizing nas benchmark performance on shared heterogeneous networks. In *Proceedings 16th International Parallel and Distributed Processing Symposium*, pages 9–pp. IEEE.