CEN6086 - Benchmarking Amazon's EC2 and Google's GCP Cloud Platforms

– Aaron Goldstein –

College of Computing, Engineering and Construction

University of North Florida

Jacksonville, United States

– n01421643@unf.edu –

– Jonathan O'Berry –

College of Computing, Engineering and Construction

University of North Florida

Jacksonville, United States

– n01445775@unf.edu –

– Andrew Simon –

College of Computing, Engineering and Construction

University of North Florida

Jacksonville, United States

– n00695969@unf.edu –

Abstract

This project aimed to compare and analyze the performance of an 8-node cluster in Amazon's Elastic Compute Cloud (EC2) and Google's Google Cloud Platform (GCP). We accomplished this by analyzing the Memory bandwidth, IO performance, and Computational performance of both cloud providers using three different benchmarks. These benchmarks were STREAM, IOR, and NAS Parallel Benchmarks-Embarrassingly Parallel (NPB-EP), covering the metrics mentioned above, respectively. We ran each of these benchmarks three times a day, two days a week for two weeks, totaling 12 times on a single cloud provider, for 24 runs for a single benchmark across both Cloud Providers, for a final total of 72 tests at the end of our study. We use the data returned by these benchmarks to calculate the statistical difference between the Cloud Providers for various performance metrics. We plot those results on charts and graphs to allow for thorough analysis, the results of that analysis which we, in addition, share in the results and conclusion of this paper.

1.0 Introduction

This project compared Amazon’s Elastic Compute Cloud (EC2) and Google’s Google Cloud Platform (GCP). We compared these by analyzing a few key metrics on their capacity. In particular, this study investigates each service's memory bandwidth, IO performance, and computational performance. We accomplished this by running three different benchmarks on both cloud providers. These Benchmarks were STREAM, IOR, and NAS Parallel Benchmarks-Embarrassingly Parallel (NPB-EP). Each of these benchmarks was run three times a day for two days a week for two weeks, totaling each being run 12 times on a single cloud provider, 24 across both cloud providers, and 72 runs total across all benchmarks across all cloud providers. We then used the data returned by these benchmarks to calculate further metrics and plotted them on various charts and graphs for a more thorough analysis.

To understand this project, one must understand the EC2 and GCP platforms. First, EC2 is an Infrastructure as a Service (IaaS). IaaS means the cloud provider provides the hardware infrastructure, and everything else is highly customizable and configurable [1]. We additionally use the GCP Compute Engine to run the benchmarks with the GCP Cloud Provider. GCP Compute Engine is also an IaaS similar to EC2 [2]. GCP also offers other Software as a Service (SaaS) and Platform as a Service (PaaS) services [2]. While these additional services are not the main focus of the study, we will briefly clarify how they differ from IaaS. The critical difference between IaaS, PaaS, and SaaS is that IaaS allows you to manage from the operating system and up.

In contrast, PaaS will allow you to create, develop, and package software bundles [14]. SaaS will only enable you to manage surface-level settings like the application configuration or your data [1]. These different service types represent different levels of control and responsibility for you and the cloud provider. IaaS represents the most control and responsibility for you as a customer, followed by PaaS, and finally, SaaS, with you holding the least responsibility and control. In this study, we run the benchmarks on two clusters across the EC2 and GCP IaaS compute platforms, with each cluster node having different running variants of the Linux Operating System. The AWS cluster used alinux2, while GCP used Ubuntu 20.4. We had to upload the benchmarks to the different clusters manually, configure the nodes to communicate with each other through SSH, properly install and configure the requirements to run these benchmarks and the system set up to handle them, modify existing scripts provided on the class Canvas page to run the benchmarks in parallel across each node configuration properly, the different nodes being one node, two nodes, four nodes, and eight nodes. In summary, IaaS systems allow for far greater customization but also put a more significant burden of work and effort onto the user, requiring more expertise and effort for a customer to utilize them effectively than PaaS or SaaS. Now that we have introduced the study, we will soon demonstrate the results we received from our experience running all three benchmarks with the two cloud providers and the strengths and weaknesses of both.

2.0 Cloud Computing Platform

Amazon's EC2 and GCP are similar in many ways but also different. First, no cloud computing solution is worth anything if it is unreliable. For this reason, both Google and Amazon highly emphasize the reliability of their services. However, they ensure this reliability in different ways. With EC2, this reliability is shown through each region having a 99.99% availability and where replacement instances can be rapidly and predictably made [3]. Google has a similar standard for GCP, expecting a minimum of 99.99% availability [4].

Additionally, Cloud Providers can only ensure this reliability if the systems are secure. Amazon EC2 has recently improved its security using Intel AES New Instructions [5]. The data is then further encrypted with Amazon EBS encryption, ensuring data-at-rest and data-in-transit is kept secure [6]. These factors combined make AWS a top contender in terms of data security. While still very secure, GCP is not currently as secure as EC2. Google is now putting a process called the Infrastructure Processing Engine on GCP. The Infrastructure Processing Engine would offload the networking, providing an additional layer of security against DDOS attacks [7], [8]. Google Cloud Platform also has the option to include the protection of Shielded VMs and Confidential VMs as part of its services. These can be especially useful against root and boot kicks [9]. This brief comparison shows that EC2 and GCP have similar levels of security, with GCP having slightly less but still being highly competitive with EC2 in terms of security. The architecture of EC2 centers around elastic load balancing and the elastic cache; the key part of both services is their elastic nature [10]. This elastic idea, also known as auto-scaling, is also offered by GCP. However, they only do it in special instance groups called Managed Instance Groups (MIGs) [11]. In the case of GCP, the Cloud Provider future divides this architecture into four categories: general purpose, compute-optimized, memory-optimized, and accelerator-optimized; this shows GCP having customization be more readily available [12]. Regarding instance types, Amazon EC2 offers:

- General Purpose
- Compute-Optimized

- Memory-Optimized
- Storage-optimized
- Accelerated Computing
- High-Performance Computing (HPC) optimized.

For each of these instance types, EC2 offers a variety of subtypes. For general purpose instances EC2 offers 18 subtypes, such as, M7g, M7i, M7i-flex, M7a, Mac, M6g, M6i, M6in, M6a, M5, M5n, M5zn, M5a, M4, T4g, T3, T3a, and T2. These subtypes have different options for the number of vCPUs, ranging from 1 vCPU to 192 vCPUs. These subtypes share the aspect that the memory size is 4 GiB per vCPU. So, if an instance uses one vCPU, there will be 4 GiB; if it uses 192 vCPUs, it will have 768 GiB [5]. For the compute-optimized type, there are 12 subtypes. These are C7g, C7gn, C7i, C6g, C6gn, C6i, C6in, C6a, C5, C5n, C5a, and C4. The number of vCPUs used in these subtypes ranges from 1 vCPU to 192 vCPUs. All of these subtypes share the common factor that memory size is 2 GiB per vCPU, so if an instance uses 1vCPU, it will have 2 GiBs of memory, and if it uses 192 vCPUs, then it will have 384 GiBs of memory [5].

For Memory-optimized instances EC2 offers 20 subtypes, including, R7g, R7iz, R7a, R6g, R6i, R6in, R6a, R5, R5n, R5b, R5a, R4, X2gd, X2idn, X2iedn, X2iezn, X1, X1e, High Memory, and z1d. For most of these, the range of the number of vCPUs is 1 to 192, and the memory is 8 GiB per vCPU. For clarity, that means that if an instance uses one vCPU, the memory would be 8 GiB, and if an instance uses 192 vCPUs, the memory would be 1536 GiB [5]. The high memory subtype is an exception; however, it does not use vCPUs and has no obvious consistent assignment of memory per processor [5]. Accelerated Computing has 15 subtypes, P5, P4, P3, P2, DL1, Trn1, Inf2, Inf1, G5g, G5, G4dn, G4ad, G3, F1, and VT1. These use vCPUs in amounts ranging from 4 vCPUs to 192 vCPUs. The memory sizes to vCPU ratio for each change depends on how each subtype is being accelerated, with the highest instance memory being 2 TiB and the lowest being 16 GiB [5].

For Storage-optimized instances Amazon EC2 offers 10 subtypes, I4g, Im4gn, Is4gen, I4i, I3, I3en, D2, D3, D3en, and H1. These subtypes have vCPU numbers ranging from 1 vCPU to 128 vCPUs. I4g, I4i, I3en, and D3 have 8 GiB per vCPU, and the maximum memory amount reached is 1024 GiB. Im4gn, D3en, and H1all have 4GiB per vCPU, and the maximum memory found is 256 GiB. Is4gen has 6 GiB per vCPU; the largest number of vCPUs is 32, with 192 GiB of memory. I3 and D2 have 7.625 GiB per vCPU, and the maximum memory value reached is 512 GiB. In total, the largest memory reached is

1024 GiB [1]. HPC optimized has four subtypes, which are Hpc7g, Hpc7a, Hpc6id, and Hpc6a. HPC optimized uses physical cores rather than vCPUs, and each of the four subtypes has a set amount of memory, which are 128 GiB, 768 GiB, 1024 GiB, and 384 GiB, respectively [5].

A bit different from Amazon's offerings, as mentioned above, GCP offers four main instance types. For each of these, like Amazon EC2, GCP also offers a variety of subtypes.

- General Purpose
- Compute-optimized
- Memory-optimized
- Accelerator Optimized

For the general purpose type, GCP offers seven subtypes, which are C3, E2, N2, N2D, Tau T2D, N1, and Tau T2A. The number of vCPUs used by these subtypes ranges from 1 vCPU to 224 vCPUs. C3 is available in 2, 4, or 8 GB per vCPU, maxing out at 1408 GB. E2, N2, and N2D are available in 1, 4, or 8 GB per vCPU. However, in the case of E2, it is limited to only a tiny number of vCPUs maxing out at 128 GB. N2 is available in much greater numbers than E2 up to 128 vCPUs, so N2 maxes out at 864 GB. N2D is available in up to 224 vCPUs when running at 4 GB per vCPU, making it max out under these conditions rather than 8 GB per vCPU, reaching 896 GB. Tau T2D and Tau T2A have 4GB per vCPU, Tau T2D maxing out at 240 GB, and Tau2A maxing out at 192 GB. N1 has 3.75 GB per vCPU, maxing at 360 GB [13].

For compute-optimized machines, GCP has three subtypes, which are H3, C2D, and C2. These are available with vCPUs ranging from 2 vCPUs to 112 vCPUs. H3 is available only in 88 vCPUs at 352 GB. C2D is available in 2, 4, or 8 GB per vCPU. It is available from 2 to 112 vCPUs, meaning it maxes out at 896 GB. C2 is available at 4 GB per vCPU and from 4 to 60 vCPUs, maxing out at 240 GB. Therefore, compute-optimized maxes out at 896 GB for GCP [13]. GCP offers three subtypes for memory-optimized types: M1, M2, and M3. M1 offers between 14.9 and 24GB per vCPU. Because of this, the M1 reaches its maximum memory capacity at 160 vCPUs and 3844 GB. M2 is available in 208 or 416 vCPUs. On 208 vCPUs, it runs at 28.3 GB per vCPU, reaching 5888 GB, whereas 416 vCPUs can run at 28.3, 21.2, or 14.2 GB per vCPU. M3 is available in the range of 32 to 128 vCPUs. M3 runs at various rates, maxing out at 3904 GB when using 128 vCPUs [13]. The accelerator-optimized type has two subtypes, A2 and G2. A2 has a standard and ultra version. The standard version runs mostly at 7.08 GB per vCPU, maxing at 680 GB. However, it also has one 96 vCPU instance that runs at 1360 GB or 14.17 GB per vCPU. The ultra version runs 14.17 GB per vCPU and maxes at 1360 GB. G2 is more straightforward than A2 in that it exists in the range of 4 to 96 vCPUs and runs at a rate of 4GB per vCPU, maxing out at 384 GB and 96 vCPUs. However, G2 does add the complexity of additional custom memory that allows for a potentially substantial increase in memory, such as a user being able to increase it from 384 GB to a value up to 432 GB [13]. This thorough comparison of the different instance options for the cloud providers' respective computing platforms shows that EC2 and GCP are highly competitive, but EC2 has a greater variety of options.

3.0 Related Work

E. F. Noviani et al. conducted a study analyzing the performance of AWS and GCP cloud providers, comparing the performance of CPU processing, latency, and throughput [15]. The authors utilized the Golang Framework (Gorilla Mux) and a SQLite database, employing JMeter for load testing the APIs of both cloud providers. Their findings reveal that AWS exhibits a higher success rate than GCP, with GCP utilizing 19.68% more CPU processing power. AWS produced an average response time of 80,931.69 ms, while GPC only produced an average response time of 15,623.77 ms. Although this seems to favor GCP, the authors suggest this discrepancy was due to the lack of errors in GCP's calculations. They conclude that while GCP may perform with less response time, this is only due to their system's lack of deep data checking. AWS outperformed GCP in all other metrics.

S. Sithiyopasakul et al. evaluate the performance of Infrastructure as a Service (IaaS) across GCP, Microsoft Azure, and AWS. IaaS primarily concerns stability, reliability, and scalability [16]. Their research emphasizes the Auto Scaling features of these platforms, measuring CPU utilization metrics. Results show GCP demonstrating the quickest scale-out times at 140 seconds, with Microsoft Azure exhibiting the slowest at 540 seconds. However, AWS excelled in recovery testing, boasting a recovery time of 89 seconds, whereas Microsoft Azure measured in at 154 and GCP at 407 seconds. Load testing results found Microsoft Azure to have the highest average response times, with AWS having longer initial response times that subsequently decreased to competitive times.

A. P. Kurniawan et al. compared the computing performance of AWS, Azure, and GCP using a Docker Container environment for a web application [18]. The authors utilized JMeter (stress testing, measuring average speed and CPU throughput), Sysbench (CPU, file IO, memory, and database performance), and Apache Benchmark (load testing). Results demonstrated that AWS performed with the highest average speed and throughput, processing the highest number of requests per second (166.5/sec compared to GCP at 117.9/s and Azure at 100.1/sec). The Sysbench results found advantages to each of the cloud providers. GCP boasts the highest file IO read/write speeds, Azure gives the best memory performance, and AWS has the most efficient CPU usage. GCP and Azure demonstrated near-equal results for the final load balance testing, whereas AWS measured at just over double their time.

A. S. Suwardi Ansyah et al. studied a similar comparison of the AWS, GCP, and Azure platforms with a focus on IoT devices leveraging MQTT Brokers [23]. Their methodology involves VMs set up across these platforms on a Linux system (8GB RAM and 2 CPU cores), measuring load testing with MQTTLoader, with messages, throughput, latency, number of subscribers, and QoS levels as performance metrics. The overall results are broken into three scenarios: Azure exhibits the highest consistency and lowest latency for scenario one, Azure outperforms in both publisher and subscriber throughput, GCP has the most messages in scenario two, and similar findings in scenario three as the previous.

4.0 Result and Comparisons

4.1 Test Bed

To manage the AWS Cluster, PCluster was used to automate the process of setting up the nodes, establishing communication, and sharing storage between them. The specific operating system used is alinux2 (Amazon Linux 2), which AWS describes as providing a "security-focused, stable, and high-performance execution environment to develop and run cloud applications" [17]. The instance type is t2 micro, starting at $0.0052 per hour. AWS describes T2 instances as a "low-cost, general-purpose instance type that provides a baseline level of CPU performance that can burst above the baseline when needed" [18]. We configured the cluster to create eight nodes and for PCluster to use Slurm (Simple Linux Utility for Resource Management) as the job scheduler associated with the cluster. Slurm is a free and open-source job scheduler for Linux, commonly used for supercomputers and computer clusters. It provides a framework for starting, executing, and monitoring our MPI (Message Passing Interface) parallel jobs [19]. The guide posted by our classmate Guna Sekaran Jaganathan was the primary reference for establishing the initial AWS environment and running the stream benchmark [20]. A combination of the different guides on Canvas was used as a reference to set up the IOR and NPB benchmarks. We have created a concise guide linked in the reference section, outlining the steps to complete this process [21].

This project used Ubuntu 16.04 LTS Pro as the operating system for the STREAM benchmark run on GCP, and the machine type was g2-small. In setting up, GCP must generate the SSH key on the root node and copy and paste it into the settings of all other nodes. This process allowed the STREAM benchmark to run on 1, 2, 4, 6, and 8 nodes. However, during this project, we discovered that using Ubuntu Pro and potentially using a Machine Type of g2 small causes issues with the IOR and NPB benchmarks when we run them over three nodes. For these reasons, we later changed to the operating system Ubuntu 20.04 and the machine type e2-small; this allowed the NPB and IOR benchmarks to run on all node configurations.

4.2 Benchmarks and Metrics

Before we analyze the results or get into the methodology, we must understand the results returned by the benchmarks and their analysis; we must know what they are doing and evaluating on the computer systems. The stream benchmark is the standard for measuring memory bandwidth. It measures the sustainable memory bandwidth and the corresponding computation rate for simple vector kernels [22]. A vector kernel involves applying basic arithmetic or mathematical operations to each vector element (a one-dimensional array of numbers). See for example [0, 1, 2, 3,4]. A vector kernel is not to be confused with an operating system kernel but rather a core computation or routine meant to be used for testing and measuring the performance of certain aspects of a computer system. The critical vector kernels of the stream benchmark are the following:

- Stream Metrics
  - Copy: where data is copied from one vector to another.
  - Scale: Multiply each element in the vector by a single value (a scalar).
  - Add: Add two vectors element-wise
  - Triad: A combination of multiplication and addition, each element is multiplied by a scalar, then added to a corresponding element of another vector.

The next benchmark is IOR, a parallel IO designed to assess the performance of parallel file systems [23]. This test is crucial as it tells us how fast our parallel computing system can write and read data to and from the storage system. We focused on the key metrics:

- IOR Metrics
  - Max Write: The maximum speed at which data was written to the storage system during the test.
  - Max Read: The maximum speed at which data was read from the storage system during the test.

The final benchmark we included in this study was the NPB 3.3.1, EP (Embarrassingly Parallel) Benchmark. This benchmark is part of NAS Parallel Benchmarks (NPB), a set of benchmarks developed by NASA to evaluate the performance of highly parallel supercomputers [25]. An Embarrassingly Parallel workload is one where little or no effort is required to separate the problem into several parallel tasks, in other words, when there is little or no need for communication between the parallel tasks or the results created by them [26]. The key metrics measured by NPB are:

- NPB Metrics
  - Mop/s total: The Computing System performed X many millions of operations per second across all processes (nodes).
  - Time in Seconds: The total time taken to run the benchmark.

Overall, these three benchmarks combined give us an excellent holistic view of the performance of different parallel computing systems and the reliability and performance of these systems in real-world scenarios, enabling us to make informed decisions. Now that we have a foundational understanding of the three benchmarks let's discuss the methodology for running these benchmarks and some of the issues we faced.

4.3 Research Methodology

As discussed previously, we carried out the benchmarks on AWS using PCluster and the previously mentioned configuration settings. Some essential considerations between the differences in the platform are that for AWS, we used a cluster management tool that automatically created the nodes for us after configuring it. In contrast, we manually created the GCP nodes based on a template we created. We decided to use AWS PCluster because StarCluster, the initially recommended tool for this class to complete this process on AWS, is no longer supported or maintained and, consequently, is rendered unusable for our study.

When creating the AWS PCluster, we made a key pair and used this to SSH into the head node for the cluster to install and run the different benchmarks for the cluster. To create and delete the nodes for the AWS PCluster, we simply start and stop the cluster every time we need to run the benchmarks and when we are done running them for the day. We can also efficiently SSH between the head and compute nodes for AWS without issue and need for further SSH configuration as was necessary for GCP, where we created a public key for the root node and manually copied that key to every compute node. We had a total of 8 nodes per cluster. We ran the benchmark on configurations of 1 node, two nodes, four nodes, and eight nodes, three times each per day, for two days a week, for two weeks, for each benchmark, for both cloud providers, leading to a total of 288 outputs.

Due to the AWS PCluster's shared storage, we installed the benchmarks on a single node for AWS, the head node of the cluster, which makes them available to all the other compute nodes on the cluster. On the other hand, we managed the individual nodes manually for GCP, installing the benchmarks on them one at a time as they did not have shared storage. The machine types for the compute nodes were T2 Micro for AWS. The AWS shared storage volume is of type gp3, with a size of 40 GiB, a base throughput of 125 MiB/s, and a max of 1000 MiB/s, according to the user console and AWS documentation [35].

Type	Volume status
gp3	✓ Okay
IOPS	Throughput
3000	125

On GCP, we used g2-small to run the stream benchmark and e2-small for the NPB and IOR benchmarks. The operating systems used differed across both cloud providers, with the AWS cluster using Amazon Linux 2 and the GCP nodes using Ubuntu 20.04 LTS. We decided to use the latter operating system based on a suggestion from our classmate, Guna Sekaran Jaganathan, after having issues running the NPB benchmark in parallel on more than four nodes in GCP with a previous operating system, Ubuntu 16.04 Pro. The error was a bottleneck regarding the ongoing connections between the nodes as an SSH authentication error when a benchmark run tried to use more than three nodes for NPB. The SSH error led us to believe that we did not configure SSH properly or some nodes were trying to connect and couldn't. We thought this was odd, as we were still able to successfully SSH to nodes individually, and they worked when executing the benchmark on a smaller number of nodes, only breaking when we hit trying to run four nodes at once. We proceeded to try adding all SSH public keys for each other node in every node to ensure all nodes could connect, but this still did not fix the issue.

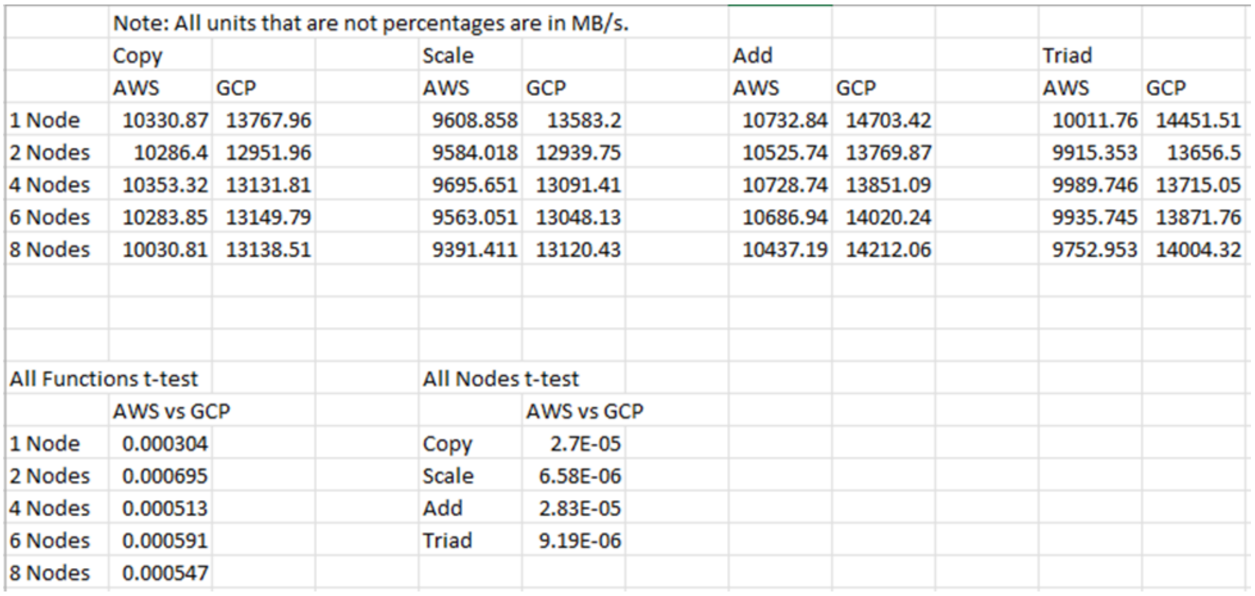
Additionally, we tried changing the machine type from g2-small to e2-small, but this still resulted in the error. The key factor resulting in the fix was a complete change to another operating system on GCP. When running the NPB, we compiled the benchmark with Class A, configuring it to use the standard benchmarking size. Ubuntu 16.04 Pro did work for the stream benchmark. Therefore, the Stream benchmark uses an operating system different from NPB and IOR for GCP, as we conducted that benchmark before making the change. We used Ubuntu 16.04 Pro initially, as Ubuntu 16.04 was the operating system recommended in the guide on Canvas for Project 1 with GCP. However, the 16.04 standard version was no longer available. In addition, for our study, when following the guide posted by Guna for AWS, we noticed after we had completed the benchmarks for Stream, we noticed Guna had posted a new command for compiling the Stream benchmark `mpicc -DPARALLEL_MPI -O3 -o stream_mpi stream_mpi.c`, which automatically aggregates the results of the benchmark. By this point, we had already aggregated the results using a custom Python script, which we will upload alongside this document. Upon confirmation with the professor after the first project demo, he requested that we re-generate this data. We re-generated the Stream benchmark outputs on AWS after using the new compile command; however, after inspecting the different outputs, we concluded that the results were not significantly different from those retrieved using the previously written Python script and that AWS was still not scaling well with parallelization for Stream, hence why we decided to proceed with our data which we collected over two weeks. We analyzed the results for Stream by manually entering them in the Excel template provided to us by the professor. From here, we generated bar plots showcasing the differences in both cloud providers across the different metrics and box plots showing the distribution and variability of the cloud providers.

Additionally, for Stream, we measured the statistical significance of the difference between the cloud providers using the provided t-test functions in the template. We conducted a t-test (including all functions simultaneously) to compare the two cloud providers across all node configurations. In addition, we conducted a different t-test with the node configuration set to eight nodes. Then, we measured the statistical differences across the different vector kernels for both cloud providers. For the analysis of IOR and NPB, we conducted a similar analysis and generated similar graphs. We measured the statistical significance of the difference between cloud providers using a related samples t-test. However, we analyzed these two benchmarks primarily through Python, Pandas, Matplotlib, Seaborn, and Scipy.

Nevertheless, we also filled out the Excel template provided by the professor to verify the results of the script. We generated descriptive statistics for each benchmark, such as metric averages, standard deviations, and coefficient of variations. In the next section, we will explain the results of our analysis.

4.4 Results

For the STREAM benchmark, as seen in the graphs below, both GCP and EC2 have a decent response rate of 10,000 and 15,000, respectively. However, while these graphs show GCP having a higher amount than EC2, the box plots show GCP covers a broader range; even the lows of that range are typically higher than those of EC2 across all node configurations. We can support this observation by the generally higher SDs and COVs of GCP vs AWS. Interestingly, both providers showed variation across the two weeks, with AWS improving in performance in the second week versus the first and GCP degrading in performance in the second week over the first. We may observe this in the box plots displayed below. AWS and GCP both fail to display noticeable benefits due to parallelization for the Stream benchmark, with higher node count configurations generally performing worse than even a single node configuration; as seen in the different bar charts, the difference between the eight-node configuration and the single node configuration is particularly noticeable for GCP. Additionally, the p values returned by the t-tests for each metric are minimal, less than 0.05, implying a statistically significant difference between the means of each vector kernel operation.



Note: All units that are not percentages are in MB/s.

	Copy		Scale		Add		Triad	
	AWS	GCP	AWS	GCP	AWS	GCP	AWS	GCP
1 Node	10330.87	13767.96	9608.858	13583.2	10732.84	14703.42	10011.76	14451.51
2 Nodes	10286.4	12951.96	9584.018	12939.75	10525.74	13769.87	9915.353	13656.5
4 Nodes	10353.32	13131.81	9695.651	13091.41	10728.74	13851.09	9989.746	13715.05
6 Nodes	10283.85	13149.79	9563.051	13048.13	10686.94	14020.24	9935.745	13871.76
8 Nodes	10030.81	13138.51	9391.411	13120.43	10437.19	14212.06	9752.953	14004.32

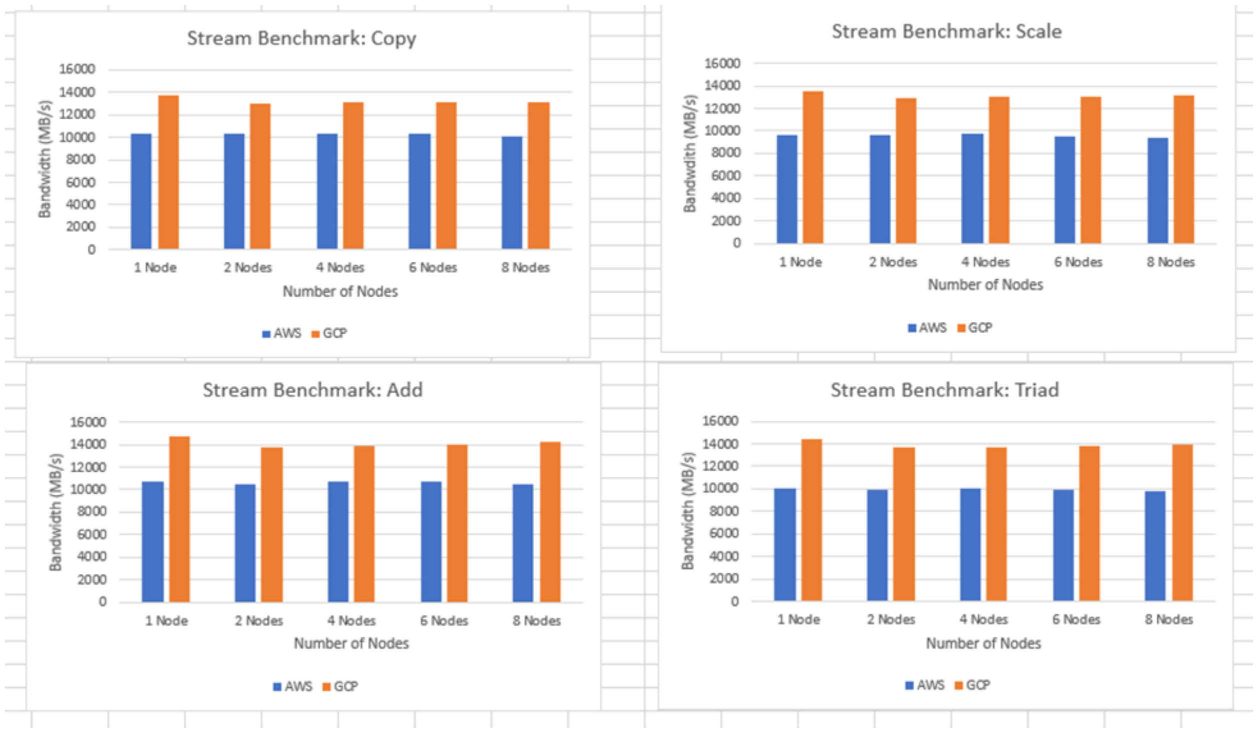
All Functions t-test

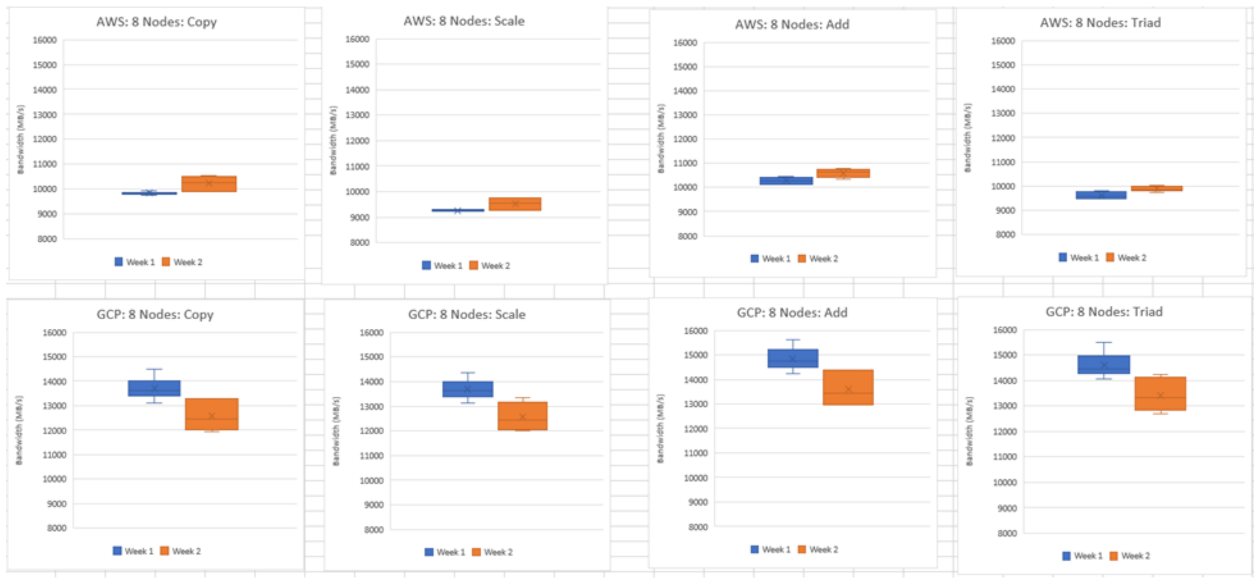
	AWS vs GCP
1 Node	0.000304
2 Nodes	0.000695
4 Nodes	0.000513
6 Nodes	0.000591
8 Nodes	0.000547

All Nodes t-test

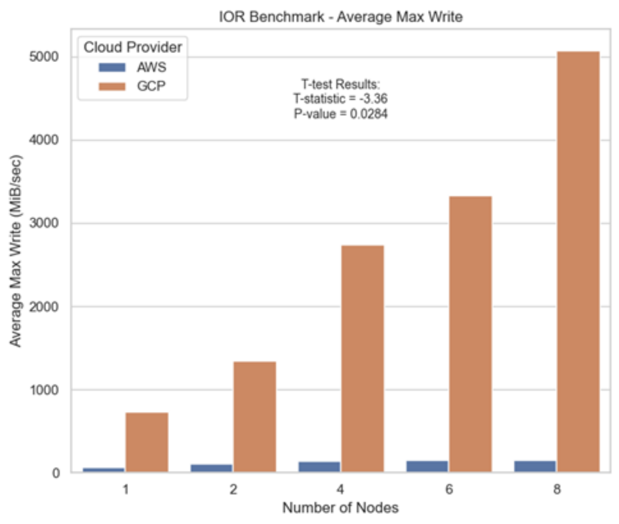
	AWS vs GCP
Copy	2.7E-05
Scale	6.58E-06
Add	2.83E-05
Triad	9.19E-06

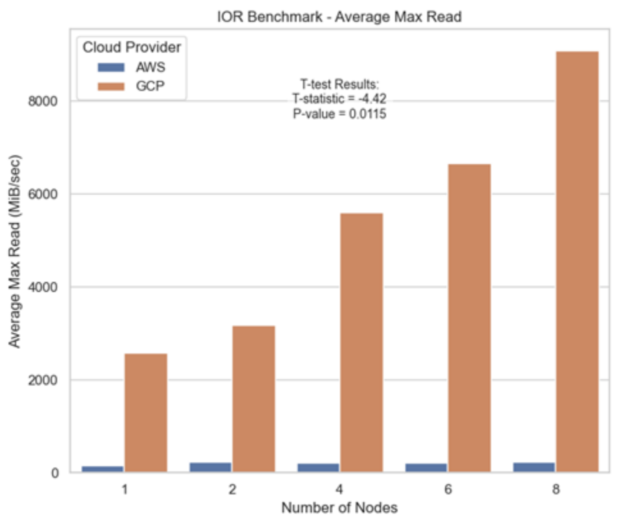
Overall, for all vector kernels, GCP reaches higher performance in terms of memory bandwidth for similar instance types, leading us to conclude that GCP is superior regarding sustainable memory bandwidth.



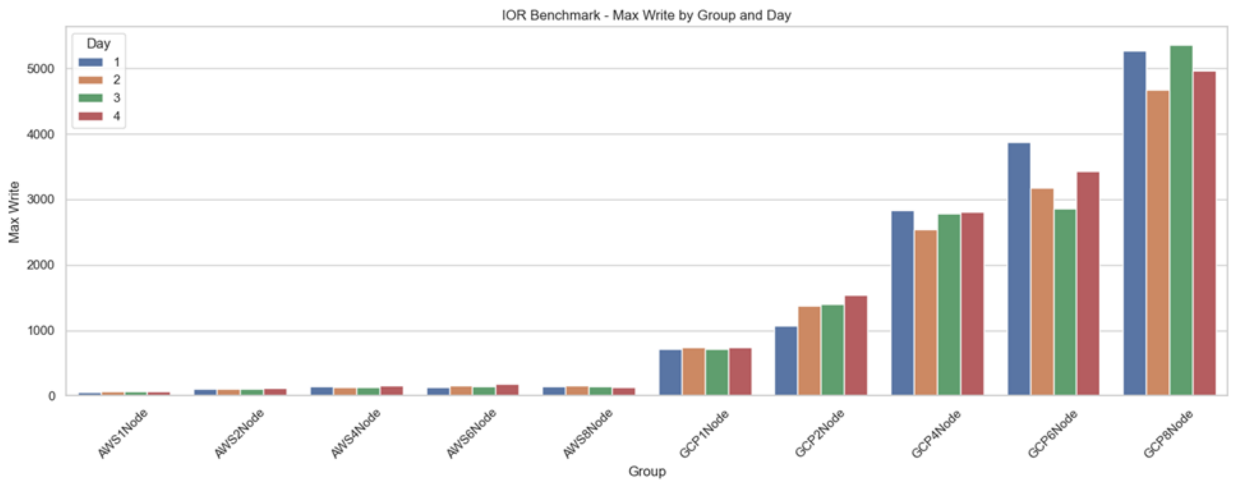


Next, we will analyze the IOR benchmark. As discussed previously, this will help us better understand the output performance of the two cloud providers. We will review the graphs individually to discover what they tell us about the benchmark results and what conclusions we can derive.

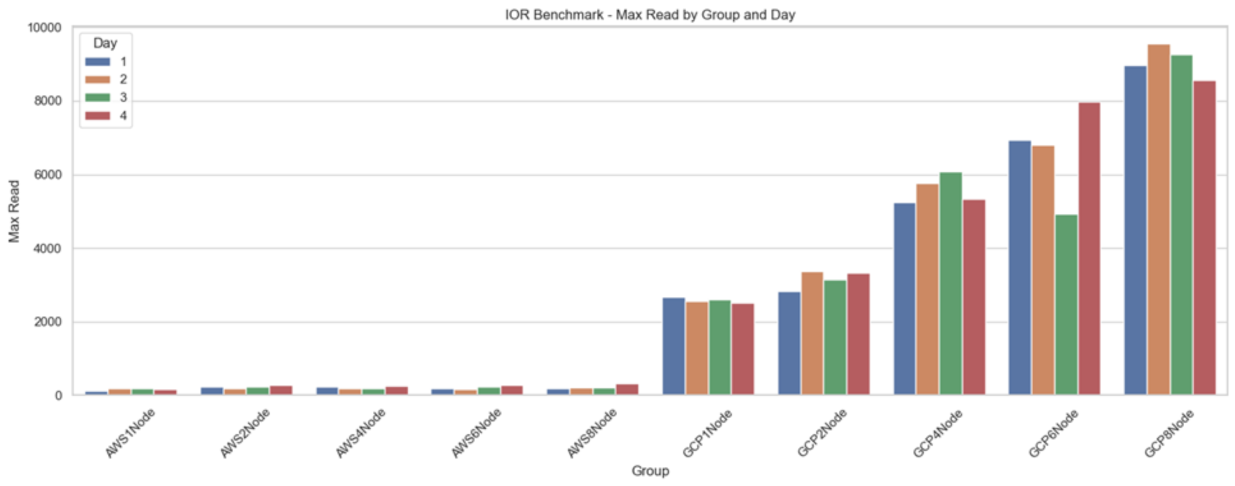




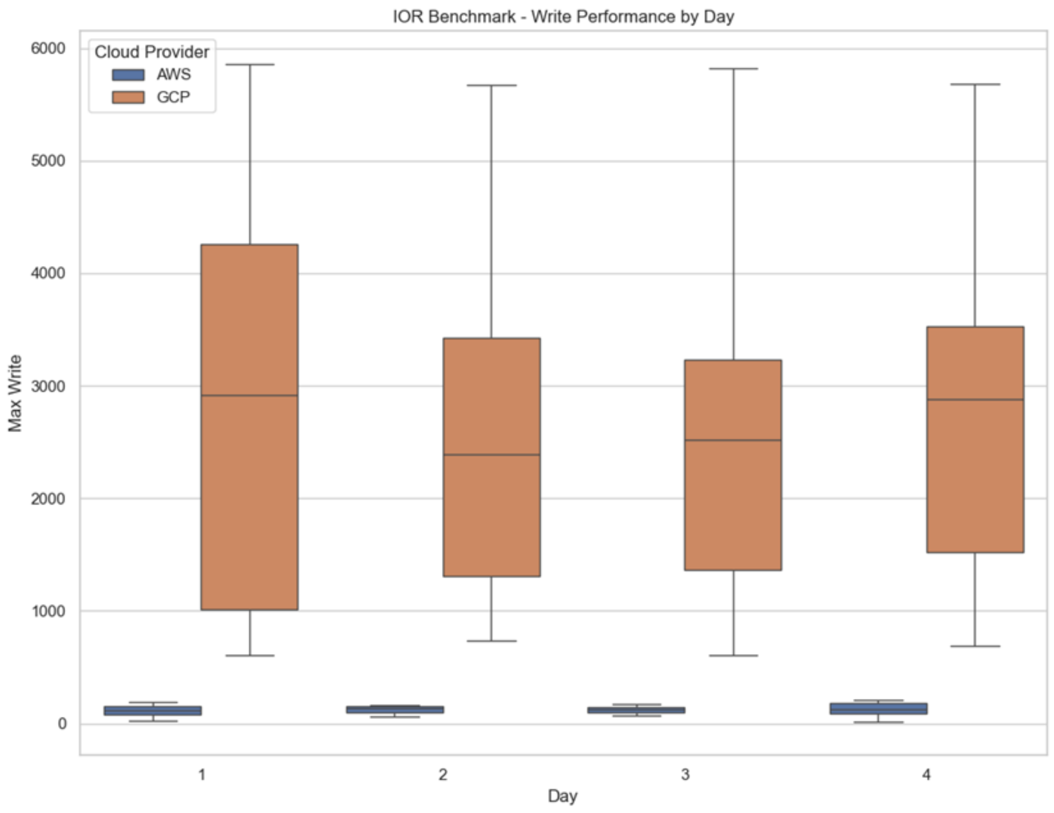
Starting with the bar chart comparing the average read and write across all the different node configurations for AWS and GCP, we can see that GCP vastly outperforms AWS in both metrics regarding MiB/sec. Suppose the visualization is not sufficiently convincing. In that case, the low p-values, less than our alpha of 0.05, indicate a statistically significant difference between the two groups for both metrics, and the negative t-values tell us AWS is performing worse than GCP. We use a shared gp3 volume of 40 GiB, 3000 IOPS, a base throughput of 125 MiB, and a max of 1000 MiB for AWS. At the same time, each GCP compute node has its own separate storage volume for each compute node. Given this information, various factors could have influenced the results we received, such as the underlying infrastructure between providers, AWS bottlenecking, or the AWS PCluster using a shared storage volume instead of separate volumes for each compute node. While adding convenience, this may have caused AWS to set a lower max on the total IO speed while GCP continued to scale based on the number of volumes, eight for each node. In addition, we see GCP benefiting from parallelization from this benchmark, beating AWS across the board, extending to when we use no parallelization at the one-node mark for both metrics. AWS uses parallelization to an extent, showing a visible progression for the write metric and the first few of the read metric before it begins to flatline.



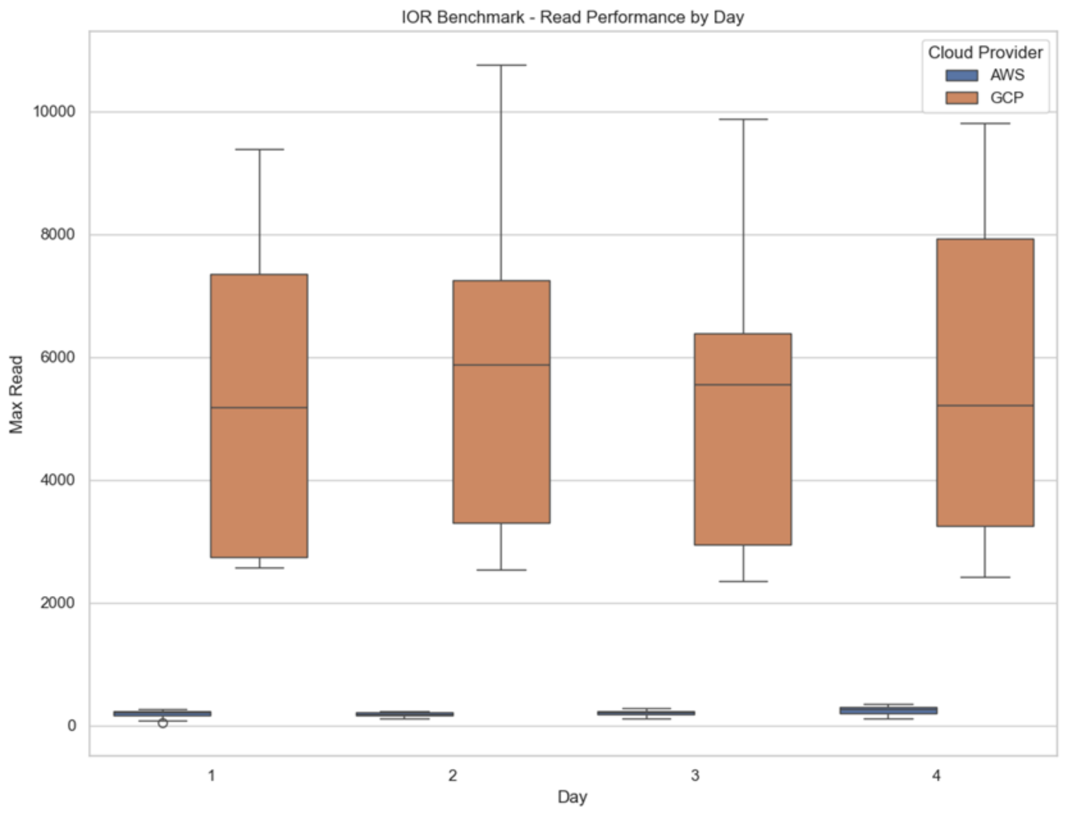
The above graph shows the overall performance for the write metric over the two weeks (days 1 and 2 vs days 3 and 4, respectively) for each cloud provider and node configuration combination. It shows that GCP is vastly outperforming AWS. However, it does not show any vast difference in performance across the two weeks, with no consistent improvement in performance in one week over the other across all node configurations.



Similar to the graph for the read metric, the above graph shows the overall performance for the read metric over the two weeks (days 1 and 2 vs. days 3 and 4, respectively) for each cloud provider and node configuration combination. While once again showing that GCP is vastly outperforming AWS, it does not show any vast difference in performance across the two weeks, save for the 3rd day for GCP on the 6th node configuration. However, even this is higher than the performance achieved with six or even eight nodes on AWS AWS.



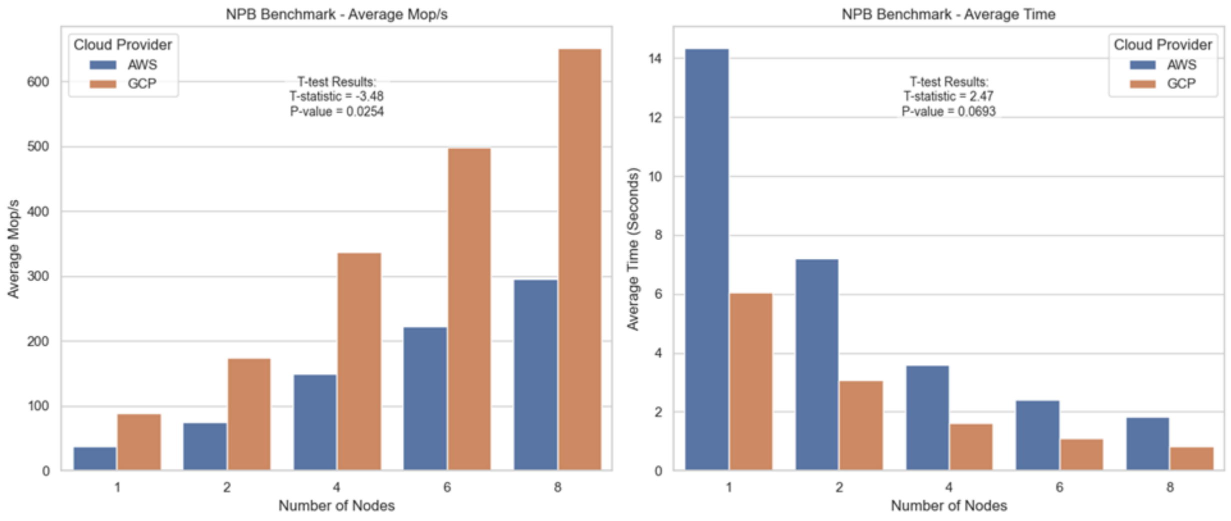
```
AWS              GCP              AWS              GCP              AWS              GCP              AWS              GCP
Min: 26.79       Min: 610.26      Min: 58.80       Min: 732.39      Min: 67.10       Min: 610.64      Min: 11.33       Min: 687.63
1Q: 76.21        1Q: 1015.82      1Q: 97.76        1Q: 1304.80      1Q: 96.13        1Q: 1365.85      1Q: 91.35        1Q: 1520.01
Median: 120.13   Median: 2914.02  Median: 137.12   Median: 2386.37  Median: 121.99   Median: 2521.05  Median: 126.63   Median: 2885.12
3Q: 156.51       3Q: 4256.87      3Q: 154.53       3Q: 3425.31      3Q: 146.62       3Q: 3232.39      3Q: 178.61       3Q: 3526.96
Max: 191.91      Max: 5861.46     Max: 166.41      Max: 5671.57     Max: 173.54      Max: 5820.10     Max: 213.11      Max: 5678.99
IQR: 80.30       IQR: 3241.05     IQR: 56.77       IQR: 2120.50     IQR: 50.48       IQR: 1866.55     IQR: 87.26       IQR: 2006.95
```



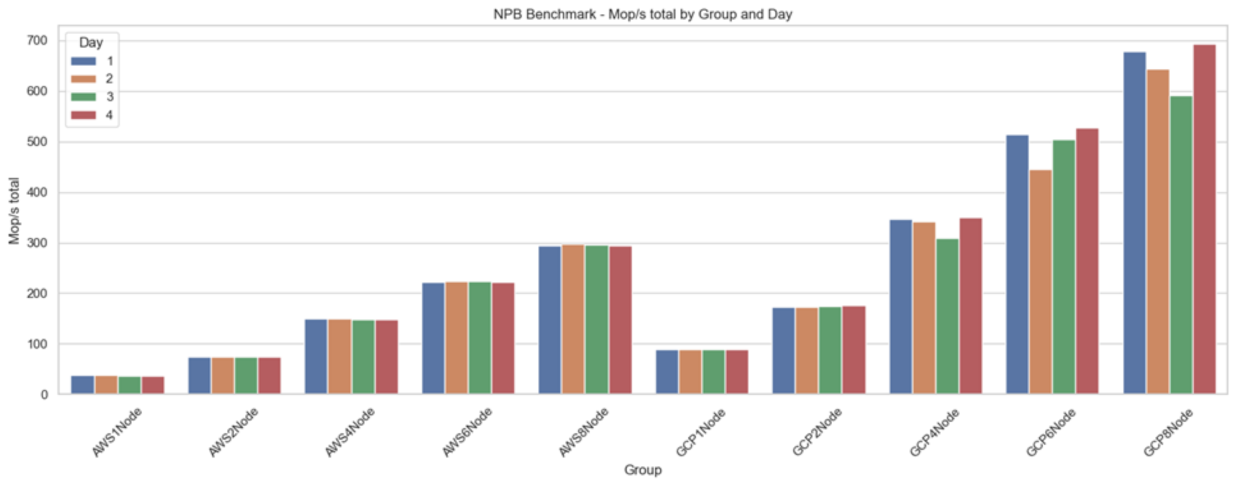
```
AWS            GCP              AWS            GCP               AWS            GCP              AWS            GCP
Min: 41.89     Min: 2572.37     Min: 117.14    Min: 2531.27      Min: 109.49    Min: 2354.21     Min: 109.24    Min: 2419.82
1Q: 167.83     1Q: 2748.02      1Q: 170.00     1Q: 3305.73       1Q: 186.06     1Q: 2942.81      1Q: 204.80     1Q: 3245.03
Median: 213.19 Median: 5177.46  Median: 186.09 Median: 5870.13   Median: 211.98 Median: 5552.02  Median: 271.32 Median: 5216.99
3Q: 236.92     3Q: 7356.24      3Q: 211.83     3Q: 7245.14       3Q: 240.46     3Q: 6377.79      3Q: 294.00     3Q: 7932.35
Max: 276.70    Max: 9383.34     Max: 234.26    Max: 10762.52     Max: 279.53    Max: 9881.46     Max: 352.30    Max: 9809.43
IQR: 69.09     IQR: 4608.22     IQR: 41.84     IQR: 3939.40      IQR: 54.40     IQR: 3434.98     IQR: 89.20     IQR: 4687.32
```

Both boxplot graphs for the IOR benchmark graphs show similar results. The distribution for GCP is much higher overall for both read and write operations while having a much larger IQR range across the board. GCP had a much higher standard deviation than AWS across both metrics for the IOR benchmark. However, GCP also generally had a lower COV than AWS, implying that AWS varies more relative to its mean than GCP.

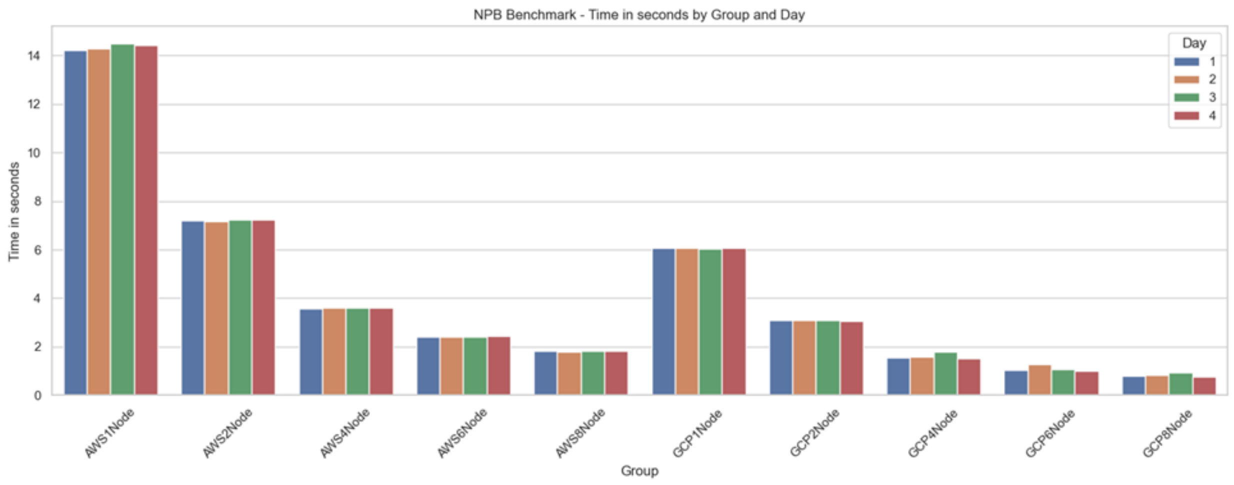
The final benchmark we ran in the study is NPB, where we focused on the metrics total millions of operations per second (the total number of operations per second across all nodes in millions) and the time in seconds for the benchmark to complete.



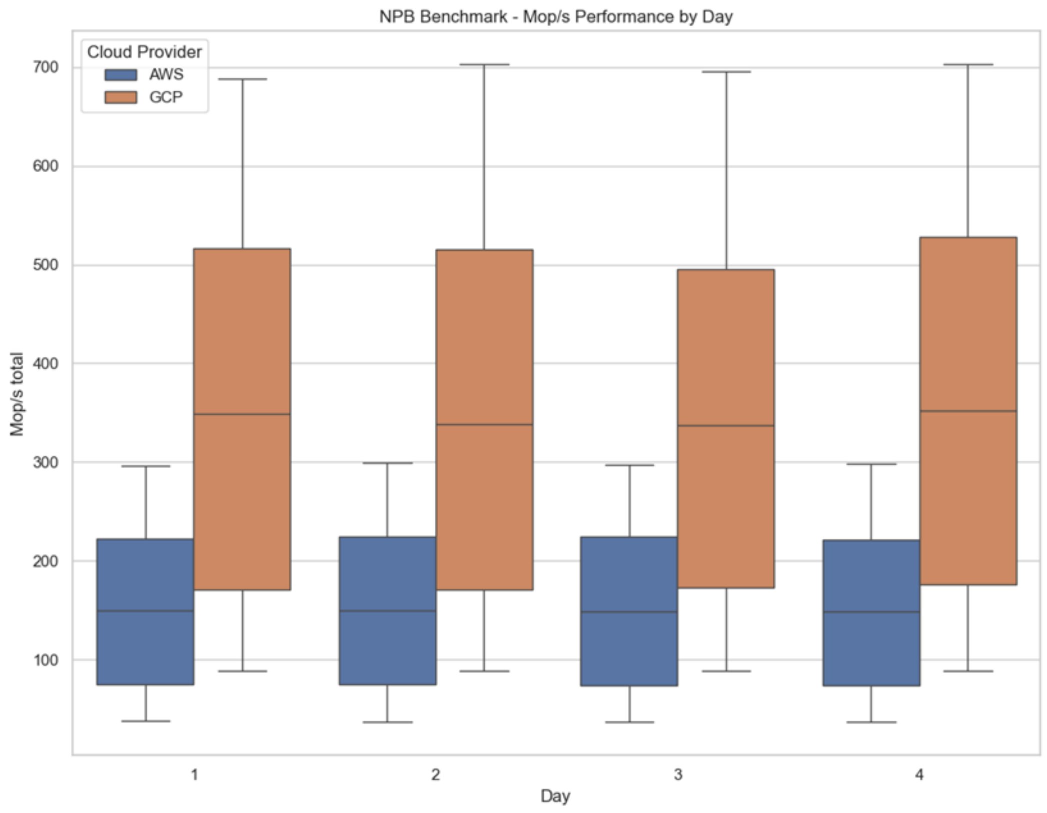
Cloud providers show signs of parallelization for both metrics; however, once again, GCP seems to provide superior performance, with AWS completing fewer operations per second and taking more than twice the time across all node configurations. The p-value value for average Mop/s is 0.0254 less than our alpha level of 0.05. Therefore, the difference is statistically significant, with a negative t value meaning AWS performed worse than GCP. The p-value for average time is 0.0693, which might imply a non-statistically significant difference if we apply a hard and fast rule to follow the standard threshold of significance of 0.05; however, simply upon inspecting the data, we can see that a reduction of more than half in terms of time by using GCP has practical significance. For the time metric, the t value is 2.47, meaning AWS receives higher numbers than GCP; however, this is the one metric where we do not want higher numbers, meaning AWS is still performing poorly compared to GCP.



There is no immediate obvious variation for Mop/s total between weeks and signs of parallelization from both cloud providers; however, GCP is still performing better across the board when the number of nodes is equal.

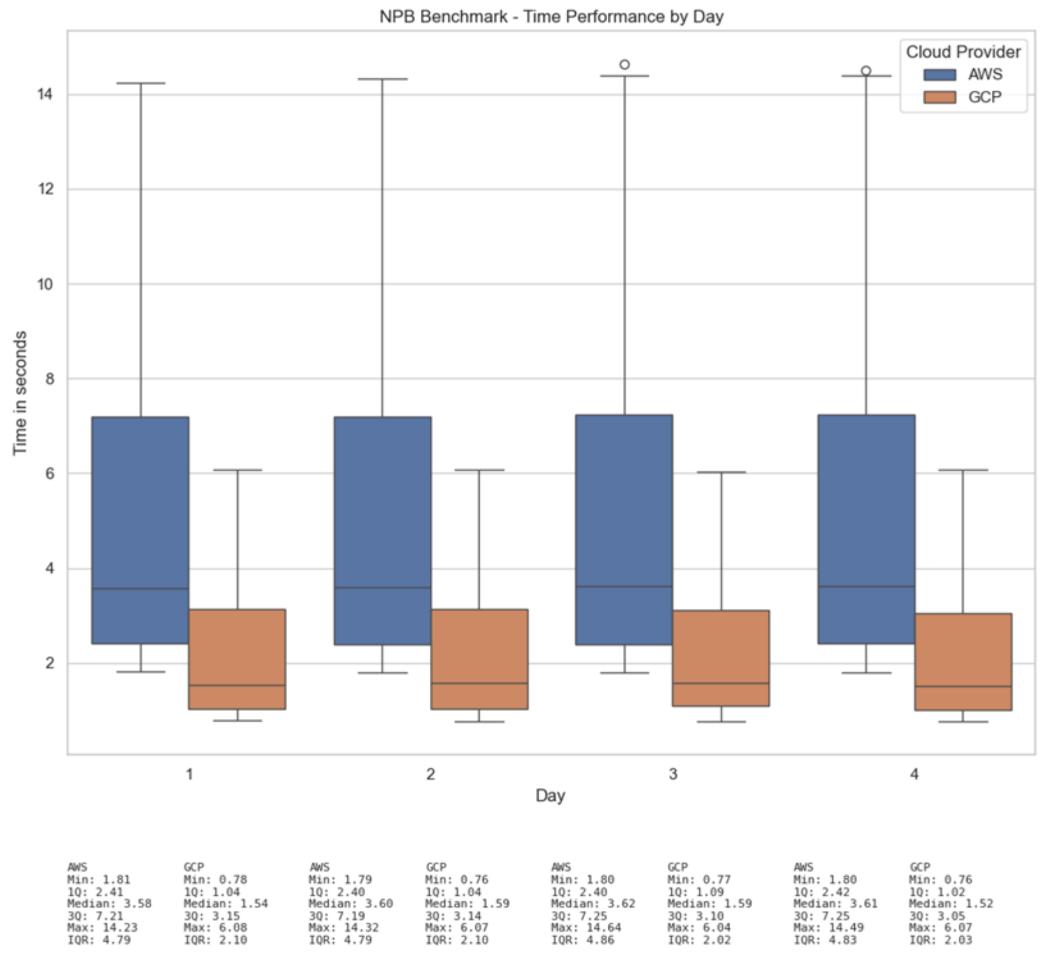


For the time in seconds metrics, there is no clear variation in weeks, clear parallelization across both providers, and superior performance for GCP for all node configurations. GCP typically takes less than half the time of AWS when there are an equal number of nodes.



```
AWS            GCP             AWS            GCP             AWS            GCP             AWS            GCP
Min: 37.73     Min: 88.27      Min: 37.50     Min: 88.52      Min: 36.67     Min: 88.84      Min: 37.06     Min: 88.40
1Q: 74.52      1Q: 170.87      1Q: 74.68      1Q: 171.14      1Q: 74.05      1Q: 173.04      1Q: 74.12      1Q: 176.15
Median: 149.90 Median: 348.46  Median: 149.29 Median: 337.93  Median: 148.43 Median: 336.92  Median: 148.59 Median: 352.56
3Q: 222.86     3Q: 516.51      3Q: 224.46     3Q: 515.53      3Q: 224.17     3Q: 495.34      3Q: 221.94     3Q: 528.17
Max: 296.20    Max: 688.23     Max: 299.19    Max: 702.73     Max: 297.75    Max: 695.25     Max: 298.65    Max: 703.25
IQR: 148.34    IQR: 345.64     IQR: 149.78    IQR: 344.39     IQR: 150.12    IQR: 322.30     IQR: 147.81    IQR: 352.02
```

The above boxplots for the Total Mop/s performance by day show GCP typically outperforming AWS. GCP tends to have a much larger IQR, typically more than twice the range. This observation from the boxplots is supported by GCP having a much higher standard deviation and COV for Average Total Mop/s than AWS for most of the node configurations except for the configuration with one node. These results suggest that while GCP typically receives better results for computationally heavy operations, AWS may have better consistency.



```
AWS            GCP            AWS            GCP            AWS            GCP            AWS            GCP
Min: 1.81      Min: 0.78      Min: 1.79      Min: 0.76      Min: 1.80      Min: 0.77      Min: 1.80      Min: 0.76
1Q: 2.41       1Q: 1.04       1Q: 2.40       1Q: 1.04       1Q: 2.40       1Q: 1.09       1Q: 2.42       1Q: 1.02
Median: 3.58   Median: 1.54   Median: 3.60   Median: 1.59   Median: 3.62   Median: 1.59   Median: 3.61   Median: 1.52
3Q: 7.21       3Q: 3.15       3Q: 7.19       3Q: 3.14       3Q: 7.25       3Q: 3.10       3Q: 7.25       3Q: 3.05
Max: 14.23     Max: 6.08      Max: 14.32     Max: 6.07      Max: 14.64     Max: 6.04      Max: 14.49     Max: 6.07
IQR: 4.79      IQR: 2.10      IQR: 4.79      IQR: 2.10      IQR: 4.86      IQR: 2.02      IQR: 4.83      IQR: 2.03
```

The above boxplot is for the time in seconds to run the NPB benchmark by day; it shows GCP with an overall lower distribution, typically outperforming AWS. GCP tends to have a much lower IQR, typically less than twice the IQR of AWS. For this metric, AWS has a lower STD across the board, except for the first node. This observation implies that AWS's average distance from the mean is generally smaller. The COV followed the same pattern, with AWS having a smaller COV except on the one-node configuration, showing less relative variability than GCP. These results show that AWS performed more consistently for the NPB benchmark, albeit receiving lesser performance.

4.5 Services Offered

Service Type	AWS EC2	GCP	Microsoft Azure
Pricing Models	On-Demand, Reserved Instances, Spot Instances, Free Tier	On-Demand, Committed Use Discounts, Sustained Use Discounts	On-Demand, Reserve Instances, Spot Instances, Azure Hybrid Benefit
Instance Types	General Purpose, Compute/Memory/Storage Optimized, Accelerated Computing	General Purpose, Compute/Memory Optimized, Accelerator Focussed	General Purpose, Compute/Memory/Storage Optimized, GPU, High -Performance
Elasticity	Amazon EC2 Auto Scaling	Compute-Engine (Virtual Machine Service)	Azure Autoscale
Containers	Amazon ECS, EKS (Kubernetes), Fargate	Google Kubernetes Engine (GKE), GCE Container-Optimized OS	Azure Kubernetes Service (AKS), Azure Container Instances (ACI)
Serverless Computing	AWS Lambda, AWS Fargate	Cloud Functions	Azure Functions, Azure Logic Apps
Object Storage	Amazon S3	Cloud Storage	Azure Blob Storage

Block Storage	Amazon EBS, Instance Store	Persistent Disks	Azure Managed Disks
File Storage	Amazon EFS	Cloud Filestore	Azure Files
Databases	Amazon RDS, DynamoDB	Cloud SQL. Cloud Spanner	Azure SQL Database, Azure Cosmos DB
Data Analytics	Amazon EMR, Athena, Redshift	BigQuery, Dataflow, Dataproc	Azure HDInsight, Azure Databricks
Networking	Amazon VPC, Route 53, Elastic Load Balancer (ELB)	Virtual Private Cloud (VPC), Load Balancing	Azure Virtual Network, Azure Load Balancer
Access Manageme nt	AWS IAM, Shared Responsibility Model	Cloud IAM, Shared Responsibility Model	Azure Active Directory, Shared Responsibility Model
Content Delivery	Amazon CloudFront	Cloud Content Delivery Network (CDN)	Azure Content Deliver Network (CDN)
Machine Learning	Amazon SageMaker	AI Platform, TensorFlow, AutoML	Azure Machine Learning, Azure Cognitive Services
Internet of Things	AWS IoT Core	Cloud IoT Core	Azure IoT Hub, Azure IoT Central

Source: [27], [28], [29].

4.6 Economic Model

AWS, GCP, and Azure all offer similar consumption-based, pay-as-you-go models that offer high flexibility, allowing users to scale their resources up or down as needed. This model can appeal to customers who wish to utilize these services at varying levels of investment, providing the service without the need for significant upfront costs or long-term commitments. While this high flexibility can be appealing in many scenarios, the dynamic nature of the model can lead to unpredictable costs as workloads fluctuate.

Both AWS and Azure offer Reserved Instances as a cost-saving option for scenarios where users can provide a pre-known, specific amount of computing capacity ahead of time. This fact makes the model more suitable for applications using a predictable workload and steady amount of resource demand over extended periods. The downsides to this model are the typical long-term lock-in for payment and the need to limit options to a specific region [30]. GCP offers a similar model called Committed Usage Discounts, allowing users to commit a specified amount of resources in exchange for a price saving [31]. The difference here is that CUDs require the user to specify an amount of resources, while Reserved Instances require specific instances, making GCP's CUD model a bit less granular. GCP's CUD offers higher project flexibility, allowing committed resources to be partitioned easily across multiple projects. Another benefit to GCP's CUD is the lack of an up-front payment requirement, making the option have less of a barrier to entry.

All three service providers also provide a unique price modeling strategy to appeal to different types of users and scenarios. AWS offers a free tier designed to allow users to explore select available services with usage limits at no cost for a limited 12-month period [32]. While this offering can be helpful for a foundational introduction to the suite of AWS services, the fixed usage and limitations to specific services hold it back from being a way to explore the platform thoroughly. GCP offers a Sustained Use Discount (SUD) deal, providing users with automatic discounts on the pay-as-you-go pricing for virtual machines that run continuously in a specific region for a given month. Discounts start at 25% usage in the month and can progress to higher tiers after additional continuous usage [33]. While this model can cause users to feel pressured to keep purchasing resources continuously to maintain the discount, it can appeal to users who will utilize these workloads as a hands-free discount on top of the usual price point. Azure offers a Hybrid Benefit model that allows users to leverage existing on-premise Windows Server licenses and SQL Server instances for VMs in Azure at a cost saving

[34]. This option is more niche as it only applies to these server types. However, it can be an appealing option for organizations with these existing licenses who wish to operate in a hybrid cloud environment.

In combination with provided services and pricing models, AWS has an edge when one wants an extensive global network and a comprehensive suite of services, appealing to a wide range of scenarios that require simple startup or a more robust array of tools. GCP excels in data analytics and machine learning, favoring scenarios and organizations that handle significant amounts of data. Azure leads as an ease-of-use option for organizations currently invested in the Microsoft ecosystem looking for seamless integration with their existing Microsoft products.

5.0 Conclusions

Our comprehensive examination of Amazon's EC2 and GCP using various benchmarks metrics has revealed significant insights into how these services perform in Memory Intensive, IO Intensive, and CPU intensive tasks.

5.1 Conclusions derived from statistics

Memory-Intensive Tasks: The Stream benchmark results indicated GCP achieves better performance in memory bandwidth, consistently scoring higher than AWS across all node configurations. This statement suggests GCP is a highly feasible option for memory-intensive applications, such as machine learning or processing large amounts of data.

IO-Intensive Tasks: The IOR benchmark results showed GCP performed better regarding read and write options in the file system. GCP's structure of separate storage volumes for each node likely contributed to this enhanced performance, potentially at an increased cost for the storage volumes and effort required to set up the benchmarks on the different nodes. These observations suggest GCP is a great choice for applications with many IO operations.

CPU-Intensive Tasks: The NPB-EP benchmark shows that GCP exhibited superior performance for CPU-bound applications, likely due to its optimized infrastructure to handle high computational loads. It could complete more operations and finish in less than half the time for all node configurations. The above observations suggest that GCP is more efficient for CPU-bound applications.

5.2 Scalability Analysis

Across the board, GCP showed better potential to scale using parallelization for the different benchmarks, except for the stream benchmark, where both benchmarks seemed to show flat performance across all node configurations. Even when both providers failed to parallelize the benchmark efficiently, GCP performed better with a statistically significant difference, as confirmed by the Stream t-tests. This observation shows that GCP has a robust infrastructure capable of efficiently scaling to manage increased workloads. EC2, on the other hand, while scalable, showed great limitations, particularly in the IOR benchmark, only showing the ability to scale successfully in the NPB benchmark and even then performing worse than GCP.

5.3 Learning Curve and User Experience

AWS and GCP presented unique challenges and learning curves. While PCluster provided a more automated and efficient environment for running the benchmarks, it required a deep understanding of knowledge specific to AWS, such as how to use AWS PCluster, configure a cluster, start a cluster, stop a cluster after running the benchmarks for the day to save money, accessing the private IP addresses of the different compute nodes using the AWS CLI and modifying the scripts provided for GCP on Canvas to better function for our AWS environment. We also had to use the private IP addresses retrieved using the AWS CLI in our scripts to cause the nodes to share the workload properly when running the benchmark. A key benefit in terms of efficiency in setting up the environment is the shared storage across the cluster, reducing the number of nodes on which a benchmark needs to be installed and reducing the possibility of user error. The PCluster setup is useful for those looking to understand better how to automate their clusters' management, perhaps after already having experience working with individual VMs, as configuring the cluster and troubleshooting can be difficult if you don’t know what you’re doing.

GCP required far more manual setup, which resulted in increased transparency and control of the environment despite potentially being more time-consuming in certain aspects, such as setting up the SSH key pairs and creating the initial compute nodes. This manual setup may be useful for those looking for a better understanding of how to manage individual nodes and how to set up connections between them. Despite providing better performance across the board, GCP gave us great issues for running the later two benchmarks, NPB and IOR, at first. Eventually, we solved this issue by rebuilding our environment from scratch using a completely different operating system. However, this, and many other fixes we attempted, took a lot of work.

5.4 Pricing Models

Concerning pricing, all platforms offer a pay-as-you-go, on-demand style pricing model, providing users with a highly flexible solution that allows them to scale their resource usage depending on their current needs. This option eliminates the need for upfront costs or time commitments, but the dynamic nature of the model can lead to unpredictable costs. AWS and Azure offer Reserved Instances for scenarios where workloads are known, and both parties can make commitments in advance, while GCP offers CUDs. These options allow users to achieve a pricing discount at the cost of a time commitment, typically an upfront payment, and some limitations to resource usage. The primary difference in appeal between Reserved Instances and CUDs is that Reserved Instances require the user to identify specific instances, while CUDs require identifying a specified number of resources. CUDs offer higher flexibility with the option to spread resources to multiple projects and can also be more appealing to organizations due to the lack of upfront fees.

5.5 Overall Assessment of Platform Suitability

AWS EC2, while we have found its performance to be worse than that of GCP in this study, is still a major player in the cloud industry due to its wide variety of instance types and broad suite of services. It's suitable for diverse use cases and provides users with flexibility. This makes it the ideal cloud provider for those individuals and businesses needing a cloud provider to meet their highly varied computational needs for highly competitive prices, often undercutting GCP on a per-house basis. [36] Being the pioneer in the field, AWS currently boasts the most extensive global reach and depth of service catalog. These offerings might appeal to well-established organizations that wish to expand current functionality in many directions, leveraging all the various specialized services AWS offers.

GCP is highly optimized in terms of computing power, making it highly effective for analyzing large amounts of data and machine learning applications. This claim can be observed by its superior performance in CPU and memory-intensive tasks, as shown by the Stream and NPB benchmarks. This observation makes it a go-to cloud provider for organizations with a strong need for data-driven insights and the need to perform complex computational tasks like machine learning. GCP stands out with its focus on data analytics, appealing to organizations operating with significant workloads seeking out machine learning capabilities.

Azure was not part of our study, but to provide a brief context, it holds a key strength in its ability to integrate with Microsoft products, making it a prime option for organizations with a strong investment in the Microsoft ecosystem.

Final Thoughts

This study highlights the importance of choosing the right cloud platform for specific use cases. AWS EC2, while remaining a colossal figure in the cloud industry, and having the largest amount of available services, GCP continues to prove itself a robust competitor for high-performance computation tasks. GCP, while potentially providing a smaller array of instance types and services, may better meet a user's computational requirements if their existing instance types or services align with their use case. Any potential cloud user should carefully consider all the factors in choosing a cloud provider, keeping in mind technical performance and price, how the solution will integrate with their existing infrastructure, and whether or not their organization has the expertise to maintain the solution, as each cloud provider has its learning curve.

References

[1] Amazon, "What is IaaS (Infrastructure as a Service)?," Amazon, https://aws.amazon.com/what-is/iaas/.

[2] Google, "What is Paas?&nbsp; |&nbsp; google cloud," Google, https://cloud.google.com/learn/what-is-paas.

[3] Amazon, "Run your workloads, not the infrastructure," Amazon, https://aws.amazon.com/ec2/ec2-get-started/#:~:text=Amazon%20EC2%20offers%20a%20highly,for%20each%20Amazon%20EC2%20Region. (accessed Dec. 4, 2023).

[4] Ancoris, "Exactly how reliable is the google cloud platform?," Ancoris, https://www.ancoris.com/blog/how-reliable-google-cloud-platform.

[5] Amazon Web Services, "Amazon EC2," Amazon, https://aws.amazon.com/ec2/instance-types/ (accessed Dec. 4, 2023).

[6] Amazon Web Services, "Amazon Ebs Encryption - Amazon elastic compute cloud," Amazon, https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/EBSEncryption.html (accessed Dec. 4, 2023).

[7] A. Conole, "Should I offload my networking to hardware? A look at hardware offloading," Red Hat - We make open source technologies for the enterprise, https://www.redhat.com/en/blog/should-i-offload-my-networking-hardware-look-hardware-offloading (accessed Dec. 4, 2023).

[8] J. Morra, "Google launches IPU co-developed with Intel Into Cloud," Electronic Design, https://www.electronicdesign.com/technologies/embedded/article/21252731/el

ectronic-design-google-launches-ipu-codeveloped-with-intel-into-cloud (accessed Dec. 4, 2023).

[9] Google, “About shielded VMS | Compute Engine documentation | google cloud,” Google, https://cloud.google.com/compute/docs/about-shielded-vm (accessed Sep. 26, 2023).

[10] Soujanya, “AWS Architecture &amp; Framework explained with diagrams,” Mindmajix, https://mindmajix.com/aws-architecture.

[11] Google, “Autoscaling groups of instances | Compute Engine documentation | google cloud,” Google, https://cloud.google.com/compute/docs/autoscaler (accessed Sep. 26, 2023).

[12] Intellipaat, “GCP Architecture,” Intellipaat, https://intellipaat.com/blog/gcp-architecture/#no3.

[13] Google, “General-purpose machine family for compute engine | Compute Engine documentation | google cloud,” Google, https://cloud.google.com/compute/docs/general-purpose-machines (accessed Dec. 4, 2023).

[14] Wikipedia contributors, “Platform as a service,” Wikipedia, The Free Encyclopedia, 02-Dec-2023, Online. Available: https://en.wikipedia.org/wiki/Platform_as_a_service (accessed Dec. 05, 2023).

[15] E. F. Noviani, B. Kembara, B. A. Yudha Pratama, D. A. Permata Sari, A. M. Shiddiqi and B. J. Santoso, "Performance Analysis of AWS and GCP Cloud Providers," 2022 IEEE International Conference on Cybernetics and Computational Intelligence (CyberneticsCom), Malang, Indonesia, 2022, pp. 236-241, doi: 10.1109/CyberneticsCom55287.2022.9865484.

[16] S. Sithiyopasakul, T. Archevapanich, B. Purahong, P. Sithiyopasakul, A. Lasakul and C. Benjangkaprasert, "Performance Evaluation of Infrastructure as a Service across Cloud Service Providers," 2023 International Electrical Engineering Congress (iEECON), Krabi, Thailand, 2023, pp. 58-63, doi: 10.1109/iEECON56657.2023.10127100.

[17] “Amazon Linux 2,” Amazon, https://aws.amazon.com/amazon-linux-2/?amazon-linux-whats-new.sort-by=item.additionalFields.postDateTime&amp;amazon-linux-whats-new.sort-order=desc (accessed Dec. 6, 2023).

[18] A. P. Kurniawan et al., "Performance Evaluation for Deploying Dockerized Web Application on AWS, GCP, and Azure," 2023 IEEE International Conference on Control, Electronics and Computer Technology (ICCECT), Jilin, China, 2023, pp. 346-350, doi: 10.1109/ICCECT57938.2023.10140775.

[19] https://en.wikipedia.org/wiki/Slurm_Workload_Manager

[20] G. S. Jaganathan and S. Ahuja, “Project 1 AWS help - Getting STREAM to run on AWS by setting up a cluster with t2.micro instance type using ParallelCluster,” Canvas, https://canvas.unf.edu/courses/95451/discussion_topics/1174869.
[21] S. Ahuja, “Project 2 - AWS Benchmark Guide - New IOR and NPB Scripts,” Canvas, https://canvas.unf.edu/courses/95451/discussion_topics/1199766.
[22] [1] Nvidia Corporation, “Nvidia Grace CPU benchmarking guide,” STREAM - NVIDIA Grace CPU Benchmarking Guide, https://nvidia.github.io/grace-cpu-benchmarking-guide/foundations/STREAM/index.html#:~:text=STREAM%20is%20the%20standard%20for%20measuring%20memory%20bandwidth.,the%20corresponding%20computation%20rate%20for%20simple%20vector%20kernels (accessed Dec. 6, 2023).
[23] A. S. Suwardi Ansyah et al., "MQTT Broker Performance Comparison between AWS, Microsoft Azure and Google Cloud Platform," 2023 International Conference on Recent Trends in Electronics and Communication (ICRTEC), Mysore, India, 2023, pp. 1-6, doi: 10.1109/ICRTEC56977.2023.10111870.
[24] G. Lockwood, “Introduction: IOR Documentation,” Introduction - ior 4.1.0+dev documentation, https://ior.readthedocs.io/en/latest/intro.html (accessed Dec. 6, 2023).
[25] Wikipedia Contributors, “NAS parallel benchmarks,” Wikipedia, https://en.wikipedia.org/wiki/NAS_Parallel_Benchmarks (accessed Dec. 6, 2023).
[26] Wikipedia Contributors, “Embarrassingly parallel,” Wikipedia, https://en.wikipedia.org/wiki/Embarrassingly_parallel (accessed Dec. 6, 2023).
[27] DigitalOcean, "Comparing AWS, Azure, and GCP," DigitalOcean Resources, https://www.digitalocean.com/resources/article/comparing-aws-azure-gcp (Accessed: December 5, 2023)
[28] Veritis, "AWS vs Azure vs GCP: The Cloud Platform of Your Choice," Veritis Blog, Available: https://www.veritis.com/blog/aws-vs-azure-vs-gcp-the-cloud-platform-of-your-choice/ (Accessed: December 5, 2023)
[29] Goldstein, Aaron. "Cloud Computing Homework." CEN6086, University of North Florida, September 2023.
[30] Veritis. "AWS vs Azure vs GCP Cloud Cost Comparison." Veritis, 2023, https://www.veritis.com/blog/aws-vs-azure-vs-gcp-cloud-cost-comparison/ (Accessed December 5, 2023)
[31] Google. "Signing Up for Committed Use Discounts." Google Cloud Documentation, 2023. Available: https://cloud.google.com/compute/docs/instances/signing-up-committed-use-discounts (Accessed December 5, 2023)

[32] Amazon. “AWS Free Tier.” Amazon Web Services Documentation, 2023. Available: https://aws.amazon.com/free/?all-free-tier.sort-by=item.additionalFields.SortRank&all-free-tier.sort-order=asc&awsf.Free%20Tier%20Types=*all&awsf.Free%20Tier%20Categories=*all (Accessed December 5, 2023)

[33] Google. “Sustained Use Discounts” Google Cloud Documentation, 2023. Available: https://cloud.google.com/compute/docs/sustained-use-discounts (Accessed December 5, 2023)

[34] Microsoft. “Azure Hybrid Benefit” Microsoft Learn, 2023. Available: https://learn.microsoft.com/en-us/windows-server/get-started/azure-hybrid-benefit (Accessed December 5, 2023)

[35] Amazon, “Amazon EBS volume types - amazon elastic compute cloud,” Amazon, https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ebs-volume-types.html (accessed Dec. 6, 2023).

[36] “AWS vs google cloud price comparison: Cost-effective solutions,” Cloudvisor, https://cloudvisor.co/blog/aws-vs-google-cloud-price/#:~:text=Larger%20Instances%3A%20For%20larger%20instances%2C%20AWS%E2%80%99s%20pricing%20remains,often%20undercutting%20Google%20Cloud%20on%20a%20per-hour%20basis.