OpenStack and Google Cloud performance comparison in Infrastructure as a Service model

Michał Łątkowski* and Robert Nowak†‡

October 19, 2022

Abstract

Cloud computing is becoming common, and the choice of proper infrastructure is essential. One of main issues is choosing between private and public cloud, between commercial and non-commercial solutions. This paper aims to compare the parameters of OpenStack and Google Cloud systems. Both systems deliver a computing cloud service, enabling the user to use the infrastructure as a service (IaaS) model. We developed the pipeline using the Python programming language and its libraries, which enable communication with the aforementioned clouds. We measured various parameters of instances and task execution: instance launch and deletion times, and their dependence on the number of launched instances. Moreover, we used benchmark algorithms to check the instance performance. We analysed the results and the factors that contributed to them and provided conclusions, recommendations, and suggestions for further research based on the gathered data.

Keywords: OpenStack, Google Cloud, Infrastructure as a Service, performance comparison

1 Introduction

Cloud computing is a model of universal, convenient, network-based access at the user’s request to a shared pool of configurable computing resources (networks, servers, applications, services) that can be quickly made available with minimal interaction by the service provider [15]. The constantly growing popularity of all clouds suggests the need for extensive, objective research, analysis, and comparisons of individual cloud solutions.

Infrastructure as a Service (IaaS) is a model in which the user has access to networks, disk space, processing nodes, etc. The user can run any operating systems and applications and, to a limited extent, manage network aspects of the resources offered. However, the user still does not physically control the infrastructure [15].

In our work, we consider two cloud technologies - Google Cloud Platform (GCP) [5] [4] and OpenStack [20] [6]. The reason for choosing these clouds was,
respectively, OpenStack’s unflagging position as the leading open-source cloud computing software and the growing popularity of Google Cloud Platform. In the [1] report, Google Cloud Platform, compared to Microsoft Azure and Amazon Web Services, achieved the greatest increase in its use by large companies. An additional motivation is that while OpenStack is a tool widely researched in the literature, few scientific articles analyse Google Cloud in the context of IaaS systems (compared to, for example, articles about Amazon Web Services).

OpenStack’s position as one of the leading technologies in the private cloud market has been reinforced by numerous studies. Much of the research (for example [23]) compared the features offered by individual private clouds as well as their architectures. However, we would like to focus more on the performance aspect.

In this article, we compare the parameters of OpenStack and GCP. The motivation of this paper is:

• An explicit comparison between those 2 technologies has not been carried out before.

• The available literature depicts the performance before 2017, and both technologies have been changed multiple times throughout 5 years.

• The conclusions are very significant for research teams at universities. Our findings support those researchers trying to decide whether to buy physical infrastructure and maintain a private cloud or use a public cloud, paying for its’ use.

• Some characteristics we analyzed have not been studied in other papers. What is more, we tried to eliminate the influence of other factors while studying each element.

We developed the pipeline using the Python programming language as depicted in Sec. 3. The measured parameters of instances and task execution are depicted in Sec. 4. We provided a wide range of parameters and used the same tools and benchmarks for both cloud systems. We conclude our research in Sec. 5. Sec. 5 depicts comments and further works.

Presented research also verifies the extent to which other papers’ conclusions have become outdated since each research we found on the aforementioned clouds was published at least four years ago.

2 Literature review

The article [13] addresses the issue of instance startup time. The authors examine AWS, Azure, and Rackspace clouds. OpenStack is not directly mentioned in this work, but Rackspace was considered, and due to the fact that Rackspace was built on OpenStack, the results might apply to OpenStack. The authors share many interesting conclusions.

• Particular attention should be paid to the information provided by cloud providers. For example, the fact that an instance is in the running state does not necessarily mean it is ready to be used.

• Instance startup time does not depend on the time of day.
• Instances running MS Windows in Rackspace take significantly longer to start up (9 times longer in the study) than those running Linux.

• Instance startup times are positively correlated with the system image size.

We explored all these aspects for both clouds (OpenStack and GCP) in this paper.

The paper [22] strictly compares three private clouds: OpenNebula, CloudStack and OpenStack. Although the authors emphasize that clouds can be compared in different ways, the most important conclusions in this work are related to efficiency. Of the three, OpenStack is the most stable. The results are surprising as all three private clouds showed similar performance results to those obtained when running tests in a native environment, i.e. without virtualization. The lack of differences stems from choosing peculiar criteria, more distant than those used in the other studies and those used in this study.

Interesting correlations can be found in the [19] study comparing OpenStack and CloudStack, such as a positive correlation between instance startup and deletion times and disk size. The conclusion is as follows – OpenStack is significantly better in terms of startup and deletion times of instances and the performance presented in benchmarks. These results are confirmed in another comparison of the same platforms [15]. According to the authors, OpenStack rewards the user for significant difficulties in the initial configuration with better benchmark results and overall stability and performance. The authors used the UnixBench tool [12]. OpenStack achieves a significant advantage in almost every test that is part of this tool.

The [21] demonstrates the advantage of OpenStack over Eucalyptus in performance tests (also carried out using UnixBench). An important conclusion is the lack of a negative correlation between the number of simultaneously launched instances and the average time of launching one instance. The conclusion suggests that OpenStack was not running instances concurrently at the time of the study. We analyse this aspect in more depth in the following sections.

The article [14] presents the analysis of the instance startup times on GCP. The startup time is from issuing the command to create an instance until the instance is ready to go into standby (equivalent to the fact that the user can connect to the instance, for example, via SSH). The author distinguishes three phases in this process: query processing, resource allocation, and instance launch. The consequence of the analysis presented in [14] is the following conclusion - among the three mentioned phases, the first two are relatively constant - that is, the user has no practical influence on these phases. The greatest fluctuations occur at the stage of starting the instance. The number of CPU cores and RAM does not matter in terms of the startup time of the instance. The boot time depends on the system image type - with the emphasis that Linux images behave similarly. The most significant differences appear when we compare Windows images with Linux images.

In the article [11], the authors formulate 15 different hypotheses regarding the performance of IaaS systems and the reasons for the differences in performance between them. There are also two important concepts not previously mentioned in this paper: hardware heterogeneity and noisy neighbour effect. Hardware heterogeneity means using different types of resources, for example, processors with different architectures, to support the same type of instance.
Some studies, e.g. [18], indicate that it may influence the observed results. The noisy neighbour effect is a deterioration in the performance of the instance in use due to sharing the physical infrastructure with some other instances currently taking up the resources. Usually, the processor cores are not shared, but there are differences in reading and writing from disk or network bandwidth.

The geographic location of the instance does matter, while temporal factors (such as time of day, day of the week) do not have a significant influence. However, it should be emphasized here that the differences between regions do not transfer between different clouds, i.e., a region that offers better performance than the other for one technology may be worse when using another cloud. In the studied clouds, the European region is usually preferable in terms of performance to the American region.

The article [3] compares GCP and AWS in terms of various factors, such as compute, memory read/write, disk I/O and other factors not analyzed in this paper (such as pricing, elasticity or network bandwidth). The authors, by running series of distributed benchmarks on Amazon Web Services and Google Cloud Platform. The conclusions were: AWS excelled in memory bandwidth, but Google Compute Engine surpassed Amazon’s EC2 in terms of read and write throughput. The CPU performance did not differ significantly.

We also studied reports from external companies that compare the performance of instances offered by individual vendors. The Cockroach Labs report from 2021 [10] compares AWS, Azure, and GCP. Using a series of benchmarks, the authors of this report reach the following conclusions: (1) if we are measuring the performance of single-core processors, GCP wins over others; (2) AWS scales better than GCP in the case of multi-core processors, as well as in the case of database transactions; (3) Azure is the best at I/O operations. The final winner of the above comparison in 2021 was GCP, which suggests there is a need for further research on this technology.

3 Materials and Methods

The pipeline we created to carry out the experiments consists of 3 tools: application for GCP measurements, application for OpenStack, and application for results analysis. All applications were made in Python. The GCP part uses the googleapiclient.discovery library to create an object that connects to the GCP cloud. We developed modules to create and delete instances, perform various measurements, and store results in the database. The OpenStack application uses openstacksdk library. The results analysis application uses a Jupyter Notebook [2] and its visualization part with our extensions.

Interactions between the user and the pipeline are presented in the Fig. 1. Our applications are used in a similar way as many Unix systems tools, where the modification of parameters is performed by setting appropriate flags. By modifying the parameters, the user can replicate all the measurements described in Section 4 using the appropriate shell commands. The detailed user instruction including a list of flags that were used to carry out the experiment is in Appendix A. The performed measurements are saved to CSV files. Each record includes startup time, deletion time, and timestamp.

GCP measurements application and OpenStack measurements application algorithm is similar and is shown in Fig. 2.
4 Results

Our experiments reduce random factors influencing the measurements by repetition. What is more, we always try to keep the other parameters (apart from the measured one) exactly the same. Each experiment, apart from the one that examines the influence of the zone (and region), is performed in the us-central1-a zone in the case of Google Cloud Platform. Each of the instance startup time measurements was repeated 100 times. The measurements of benchmarks were repeated 10 times. Such an approach reduces most of the factors, such as the influence of the state of the provider’s services or noisy neighbour problem.

For our experiments, we defined several instances depicted in Tab. 1.

4.1 Dependence of instance launch times on the time of day

Both in GCP and OpenStack, the time of the day has no effect on instance startup times, as depicted in Fig. 3. The difference between days of the week is also not important. The reason for this could be, in both cases, infrastructure redundancy. Both clouds had enough resources at any point in time during the analysis period. However, in the case of private clouds, which often do not have service level agreements, one should take into account the hardware capabilities of the infrastructure, as well as whether intensive calculations are performed during measurements.
Our results confirm the conclusions of the article [11]. Considering the above experiment results, in further experiments, the times of the day and days of the week are treated as negligible factors in OpenStack and GCP.

### 4.2 Dependence of startup times on the region in GCP

We compared the times of launching the instance when it is placed in `us-central1-a` zone to `europe-central2-a`. In both cases, an instance of type `n1-standard-1` was used.

The Fig. 2 shows the results. Not only do instances in the `europe-central2-a` zone take less time to start up on average than those in `us-central1-a` (12.7 versus 13.5), but they also have different variances (9.7 in the US and 5.0 in Europe). Our results confirmed conclusions from [11].

A single measurement in the European zone, which differs from the average, was made at 23:07:51, on December 17, 2021. Most likely, it was due to network problems or a temporary delay in service delivery from GCP, as other measurements made that day are close to average.

In our opinion, the difference in launching time is the data centre load in a
name | vCPU | RAM | disk | comments
--- | --- | --- | --- | ---
f1-micro | 1 × | 614 MB | 10 GB | a shared-core instance, GCP instance
n1-standard-1 | 1 × | 3.75 GB | 10 GB | contrary to OpenStack, GCP instances are not tied with any particular disk size. However, in this paper, GCP instances use 10 GB disks
n1-standard-8 | 8 × | 30 GB | 10 GB | GCP instance
custom | 8 × | 16 GB | 16 GB | GCP instance
os1 | 1 × | 1 GB | 5GB | OpenStack instance
os2 | 1 × | 1 GB | 10 GB | OpenStack instance
os3 | 4 × | 4 GB | 10 GB | OpenStack instance
os4 | 4 × | 4 GB | 32 GB | OpenStack instance
os5 | 8 × | 16 GB | 10 GB | OpenStack instance
os6 | 8 × | 16 GB | 64 GB | OpenStack instance
os7 | 16 × | 64 GB | 64 GB | OpenStack instance

Table 1: Instances used in numerical experiments. The description includes the number of virtual CPUs, RAM memory size, and disk space. If not mentioned otherwise, the instance has the Ubuntu 20.04 system.

Figure 3: Dependence of instance launch times on the time of day, GCP with n1-standard-1 on the left, OpenStack with os1 on the right.

given zone. The zone in Warsaw is relatively new at the time of writing, it was opened in 2021, and the hardware in this zone is not under significant load. In turn, the studied American region is older. Its age is already around 5 years, so its load is bigger. Secondly, the prices of instances in both zones are different,
which affects their popularity.

The results depict that the region is an important factor. We eliminate this factor from other experiments, as the zone is the us-central1-a for next experiments. OpenStack’s geographical comparison was not performed because we used the Warsaw University of Technology private cloud.

4.3 Dependence of instance launch times on machine type

In the experiment below, we will examine the effect of the machine type, i.e. the number of virtual processors, RAM size, etc. on the instance startup times. In GCP we compare: f1-micro, n1-standard-1, and n1-standard-8 (from Tab. 1). In OpenStack we compare: os2, os3, and os5.

As can be seen in Fig. 5, we have the average launch times (in seconds): 13.6 (n1-standard-8), 12.9 (n1-standard-1), 13.5 (f1-micro), 10.8 (os2), 10.7 (os3), 10.9 (os5). There is no correlation between machine type and startup times. This means that the bigger machines startup just as quickly as the smaller ones, contrary to intuition. Our results confirm the findings from the study for GCP [9]. In the case of OpenStack, our findings are in contradiction to the conclusions of [19]. However, the authors themselves admitted that the correlations were not strong. In the article [13], mere observation of the graphs clearly shows that instance launch times on Rackspace did depend on machine type, while such correlation was not apparent in AWS. It’s worth noting, however, that Rackspace may or may not display similar behaviour to OpenStack – these are not the same technologies. OpenStack is still under development and 20 new editions have appeared since 2012. The outdating of the above research becomes even more likely when looking at the instance startup times. The instance, which the authors named type 1 (256 MB RAM and 10 GB disk) started on average in 35 seconds. In our experiments, a bigger instance starts in about 10 seconds.
4.4 Dependence of startup times on the disk size

The number of virtual processors and RAM does not influence the instance startup times, as depicted in Section 4.3. To compare the startup times dependence on disk size, we used three instances in GCP, all of type \textit{n1standard}, varying in disk sizes: 10GB, 50GB, and 500GB. We also present the results of a similar experiment in the OpenStack cloud, using \textit{os1}, \textit{os4}, \textit{os6} and \textit{os7}.

By analysing Fig. 6, we get a clear conclusion – the disk size does not seem to affect while the instance is running, the average times (in seconds) are: 12.9 (10GB), 13.4 (50GB), 12.9 (500GB), 11.5 (os1, 5GB), 10.9 (os2, 10GB), 10.7 (os5, 10GB), 10.8 (os6, 64GB). Combining machine type and disk size also does not result in significantly different values.

4.5 Dependence of instance startup time on the system

An important element that should be taken into account when setting the parameters of an instance is the operating system. In GCP we used \textit{n1-standard-1} machines, with Ubuntu, Debian, and Clear Linux systems, where Clear Linux is a system with very small size and requirements. In OpenStack, we used \textit{os1} machines, and we decided to compare two systems – Ubuntu and Cirros, where Cirros is a lightweight system designed for the needs of OpenStack, mainly for testing purposes.

The differences in the operating system from the Linux family installed on the instance are negligible and do not correlate with the size of the operating system. The results are depicted in Fig. 7 and in Tab 2.
Figure 6: Dependence of startup times on the instance disk size, GCP on the left with *n1-standard* instance, OpenStack on the right.

Figure 7: OpenStack - Dependence of startup times on the system, GCP on the left, OpenStack on the right.
| Cloud  | System    | Average [s] |
|--------|-----------|-------------|
| GCP    | Ubuntu    | 12.9        |
| GCP    | Debian    | 13.1        |
| GCP    | Clear Linux | 13.2       |
| OpenStack | Ubuntu | 10.7        |
| OpenStack | Cirros | 10.5        |

Table 2: Dependence of startup times on the operating system.

Similarly, they store ready-made images, and starting up systems does not take much time, regardless of their size. The average time for launching a Linux machine is 12.9 s.

We also investigate the instances with MS Windows operating system. The results are depicted in Fig. 8. We present only GCP with n1-standard-1. The average time launching for MS Windows instance is 13.6 seconds.

Figure 8: GCP - difference between Windows and Linux in terms of instance startup time on GCP.

4.6 Dependence of startup times on the number of instances in GCP

GCP bulkInsert function allows the user to start up multiple instances at once. In this study, we check whether starting up multiple instances at once has a significant impact on the speed of their launch. We are using n1-standard-1 instance.

The results shown in Fig. 9 are as expected. The number of instances significantly changed the startup time of an individual instance. For a single instance, we got on average 12.9 seconds of startup time; for two instances: 20.7 s; three instances: 21.6 s; and four instances 21.9 s. The main reason for such results is, in our opinion, the fact that the speed of starting the instance groups is as high as the slowest virtual machine’s speed.

4.7 Dependence of startup times on preemptibility in GCP

The preemptible instances should theoretically be a good choice for users who need instances for a short period and do not care about system stability. It is
worth noting that GCP does not guarantee the availability of these instances or any level of quality of the services provided in relation to them. Moreover, the company reserves the right to stop the instance at any time (and after 24 hours they stop automatically). The experiment depicted in this section examines whether preemptible instances have the same launch times as their non-preemptible counterparts. For this we are using *n1-standard-1* instance with Clear Linux.

The results in Fig. 10 depict that the difference is small, on average preemptible instances are launched in 12.0 seconds, and non-preemptible instances in 13.2 seconds. The preparation time is lower because such instances do not share resources with regular instances. Our research suggests no reason to avoid using a preemptible instance, at least when taking startup time into account.

4.8 Instance deletion times

Instance deletion times are a pertinent problem only in private clouds. In the case of most popular public clouds, there are no incurred costs in the deletion process, and thus the average user does not have any interest in the fast speed of the deletion process. On the other hand, releasing resources in private clouds is important.

As can be seen in Fig. [11] instances on GCP are usually deleted in around 120 seconds, however, sometimes these times are closer to 20 seconds. Such long
deletion was depicted in GCP documentation [7], the provider gives the user some time to undo his decision. Due to this process, comparing the two types of machines does not allow us to reach any useful conclusions. Deletion times in OpenStack are nearly exactly the same regardless of the system, machine type, and disk size, in our research deletion takes 1.4 seconds on average. This is an expected result, taking into account the results of the instance startup times analysis.

4.9 Comparison of GCP and OpenStack in terms of instance startup times

Fig 12 presents the speed and the changes in the speed of creating an instance on OpenStack and Google Cloud for similar instances. As the previous experiments showed, the machine type does not affect the startup times.

We see a significant difference of over 2 seconds, 10.8 seconds for OpenStack, and 12.9 seconds for GCP (on average).

4.10 Instance performance comparison

In this section, we compare the results obtained by running UnixBench tool [12] 10 times on GCP and OpenStack instances.

UnixBench is a set of benchmark algorithms that measure a broad spectrum of instance performance measurements. Following the attached link [12] details about each test can be found. The UnixBench measures performance for each test (for example, in seconds for calculation, kBps for copying). Next, the
measured value is compared with the BaseLine instance to produce the score’s quotient (how many times the tested instance is better than the baseline instance). The scores do not correspond directly to any physical unit. The scores are further combined to make an overall index for the system.

The first experiment is based on the comparison of single-threaded instances, the results were depicted in Tab. 3.

Table 3: Results of UnixBench tests for 1× vCPU, 1 GB RAM instances, \( \mu \) is average value, \( \sigma \) denotes standard deviation.

| Test name                                           | GCP   |          | OpenStack |          |
|-----------------------------------------------------|-------|----------|-----------|----------|
| (a) Dhrystone 2 using register variables            | 34 * 10^6 | 0.02\( \mu \) | 57 * 10^6 | 0.01\( \mu \) |
| (b) Double-Precision Whetstone                      | 5.8 * 10^4 | 0.02\( \mu \) | 9.7 * 10^4 | 0.001\( \mu \) |
| (c) Excel Throughput                                | 3.2 * 10^4 | 0.05\( \mu \) | 5.3 * 10^4 | 0.01\( \mu \) |
| (d) File Copy 1024 bufsize 2000 maxblocks          | 503 * 10^3 | 0.07\( \mu \) | 861 * 10^3 | 0.01\( \mu \) |
| (e) File Copy 256 bufsize 500 maxblocks            | 137 * 10^3 | 0.06\( \mu \) | 221 * 10^3 | 0.01\( \mu \) |
| (f) File Copy 4096 bufsize 8000 maxblocks          | 1.4 * 10^6 | 0.12\( \mu \) | 2.59 * 10^6 | 0.002\( \mu \) |
| (g) Pipe Throughput                                 | 681 * 10^3 | 0.06\( \mu \) | 1.1 * 10^6 | 0.03\( \mu \) |
| (h) Pipe-based Context Switching                    | 146 * 10^3 | 0.05\( \mu \) | 248 * 10^3 | 0.02\( \mu \) |
| (i) Process Creation                                | 9.7 * 10^4 | 0.03\( \mu \) | 15 * 10^4 | 0.02\( \mu \) |
| (j) Shell Scripts (1 concurrent)                    | 6.8 * 10^4 | 0.04\( \mu \) | 11.8 * 10^4 | 0.01\( \mu \) |
| (k) Shell Scripts (8 concurrent)                    | 887 | 0.04\( \mu \) | 1.6 * 10^4 | 0.01\( \mu \) |
| (l) System Call Overhead                            | 396 * 10^3 | 0.04\( \mu \) | 604 * 10^4 | 0.004\( \mu \) |

The conclusions are surprising, to say the least. Not only is OpenStack more stable (1% coefficient of variation compared to 5% for GCP), it also offers better performance in literally any UnixBench test. As a reminder, in the experiment, we tried to eliminate the influence of virtually every factor that the user can control. OpenStack achieved an average of 61% better performance.

There are many factors that could have influenced the result, such as hardware heterogeneity. However, it seems that, as in the article [11], hardware heterogeneity does not play a big role here - it mainly affects the coefficient of variation, and the averages differ most significantly. There are many aspects the impact of which we cannot measure for legal reasons (because the public cloud provider does not provide such information) or for practical reasons (be-
cause the architecture of the private cloud is so complex that it is impossible to isolate certain factors). These include, for example: numerous layers of middleware present in IaaS systems (including the program managing the allocation of resources to individual instances), the virtualization technology used, or the noisy neighbour problem. The most significant factor that was not taken into account in the study yet is the type of processor. It is true that both types of instances had 1 virtual processor, but these processors could be based on a completely different physical infrastructure. Two subsections later we will analyse this aspect as well.

We analyse UnixBench results not only in the context of comparing the values between two instances, but also to compare parallel versions versus single-threaded instances, as depicted in Tab. 4

Table 4: Results of UnixBench tests on 8× vCPU, 16 GB RAM instances. Two versions of the benchmark scripts were used, benchmark using single process (sequential, seq.) and benchmark with 8 processes (concurrent, conc.). Test names, μ and σ definitions are depicted in Tab 3.

| Test | GCP seq. | GCP conc. | OpenStack seq. | OpenStack conc. |
|------|----------|-----------|----------------|-----------------|
|      | μ        | σ         | μ              | σ              |
| (a)  | 34×10^6  | 0.01μ     | 187×10^6       | 0.03μ          |
| (b)  | 5.7×10^6  | 0.001μ    | 41×10^6        | 0.03μ          |
| (c)  | 3.1×10^3  | 0.02μ     | 14.7×10^3      | 0.03μ          |
| (d)  | 468×10^3  | 0.03μ     | 674×10^3       | 0.04μ          |
| (e)  | 129×10^3  | 0.03μ     | 183×10^3       | 0.03μ          |
| (f)  | 1.3×10^6  | 0.06μ     | 2.1×10^6       | 0.05μ          |
| (g)  | 662×10^3  | 0.04μ     | 3.45×10^6      | 0.04μ          |
| (h)  | 50×10^3   | 0.07μ     | 626×10^3       | 0.04μ          |
| (i)  | 5.82×10^7 | 0.07μ     | 29.6×10^3      | 0.03μ          |
| (j)  | 8.1×10^4  | 0.03μ     | 28.8×10^4      | 0.1μ           |
| (k)  | 3×10^4    | 0.02μ     | 4.2×10^4       | 0.11μ          |
| (l)  | 395×10^3  | 0.02μ     | 1.95×10^5      | 0.02μ          |

The fully scalable algorithms should get about 8× better results in 8 cores instances. In our experiments for the file system-related tests, we did not observe significant improvement on multi-core instances, as the tests are dependent on disk I/O performance. Please note that Shell Scripts (8 concurrent) runs 8 scripts in parallel. Such tests cannot run faster on multi-core instances. The interesting tests for comparing CPU performance are Drystone, Whetstone, Excel throughput, Pipe throughput, and Process creation. We get an improvement in performance, in GCP 5.17×, in OpenStack 5.54×. It is close to the theoretical maximum, 8×. It is worth mentioning that OpenStack is better at using CPU concurrency.

GCP allows the user to create instances using the processor type N1 or N2 (or others, not mentioned here). Tab. 5 allow to compare GCP instances with N2 processor with OpenStack instances. For such configuration, the OpenStack instance is only 1.11× more efficient than GCP. The processor type is important for the performance of instances on GCP, especially for concurrent algorithms.

Fig. 13 depicts an overall comparison for the tested instances, this index is combination of all 12 tests presented in Tab 5 Tab 4 and Tab 5. For each
Table 5: Results of UnixBench tests on GCP instance with 8× vCPU, 16 GB RAM and N2 processor. Two versions of the benchmark scripts were used (sequential, concurrent), as depicted in Tab 4. Test names, μ and σ definitions are depicted in Tab 3.

| Test | sequential | | concurrent | |
|------|------------|------------|------------|------------|
|      | μ          | σ          | μ          | σ          |
| (a)  | 42.9 × 10⁶| 0.004μ     | 246 × 10⁶  | 0.002μ     |
| (b)  | 7.3 × 10⁴ | 0.001μ     | 52.2 × 10⁴ | 0.001μ     |
| (c)  | 4 × 10⁴   | 0.01μ      | 18.3 × 10⁴ | 0.01μ      |
| (d)  | 779 × 10³ | 0.02μ      | 795 × 10³  | 0.02μ      |
| (e)  | 212 × 10³ | 0.01μ      | 216 × 10³  | 0.01μ      |
| (f)  | 2.02 × 10³| 0.06μ      | 2.45 × 10³ | 0.05μ      |
| (g)  | 1.18 × 10³| 0.003μ     | 5.2 × 10³  | 0.01μ      |
| (h)  | 63.3 × 10³| 0.01μ      | 861 × 10³  | 0.02μ      |
| (i)  | 8.6 × 10³ | 0.03μ      | 38.1 × 10³ | 0.01μ      |
| (j)  | 10.7 × 10³| 0.02μ      | 36.8 × 10³ | 0.01μ      |
| (k)  | 3.82 × 10³| 0.01μ      | 5.4 × 10³  | 0.02μ      |
| (l)  | 795 × 10³ | 0.005μ     | 3.01 × 10³ | 0.01μ      |

After the test, the lowest benchmark of all the tested instances (a) - (h) was found, then the value of this benchmark was used to scale the results of that test. After scaling the lowest benchmark has score = 1.0, the results for other instances have benchmark ≥ 1. Fig. 13 presents average of scaled scores.

Figure 13: Benchmark comparison: (a): GCP - N1 instance with 1 core; (b): OpenStack - instance with 1 core; (c): GCP - N1 instance with 8 cores, non-concurrent benchmark; (d): GCP - N1 instance with 8 cores, concurrent benchmark; (e): OpenStack - instance with 8-cores, non-concurrent benchmark; (f): OpenStack - instance with 8-cores, concurrent benchmark; (g): GCP - N2 instance with 8-cores, non-concurrent benchmark; (h): GCP - N2 instance with 8-cores, concurrent benchmark.
5 Discussion

In our experiments, when we analysed the influence of a factor on performance, we eliminated or reduced other factors. It allowed us to interpret results and give constructive conclusions properly. We noted in some previous works lack of this step.

It is not easy to clearly distinguish all the factors influencing the measurements in complex cloud architecture. There were factors which could not be completely eliminated, such as technology under instances, such as the type of hypervisor used. Moreover, some tests could have been more accurate if the data was collected throughout the years.

The benchmarks results should be treated as suggestions because every particular computational task has its specificity, with its variety of databases transfers, calculations, data flow, etc. Moreover, the benchmarks do not cover all aspects of a computational task, for example, UnixBench does not examine network performance. We recommend not applying our conclusions directly but rather comparing the performance of the selected software system on different vendors.

Another aspect is studying the price-performance ratio and quality of services. Such analyses are much easier to perform when comparing public clouds, where conclusions about the advantage of one service over another seem to be complete only after taking the cost into account. For example, performance varies between regions, but a typical application still needs to consider price differences.

We would also recommend analyzing clouds using more real-life workflows, as was done in the paper [17], where the authors used a custom I/O intensive workflow in their analysis. While the results of such analyses are harder to compare to results obtained in other articles, they are invaluable in providing insights.

It should be noted that the most accurate comparison results could be obtained when using the same physical infrastructure. Nevertheless, we think there is some merit and need to compare those clouds from a user’s perspective, treating them as black boxes. The presented conclusions are very significant for research teams trying to decide whether to buy physical infrastructure and maintain a private cloud or use a public cloud, paying for its’ use.

6 Conclusions

Based on the results obtained in our experiments, the instances offered by OpenStack and Google Cloud Platform have comparable performance. OpenStack achieved 5% better performance in a single run situation and 10% better performance in concurrent multiple startup situations when compared to GCP. However, it is worth noting that the processor types of compared instances were not necessarily exactly the same, which means that the obtained results do not imply the superiority of one of those clouds.

Both discussed technologies effectively eliminated many initial differences, for example, the instance launch times between systems, machine types, times of the day. Both platforms effectively support concurrency.
Availability and requirements

Data used for this research is available on request.

Competing interests

The author declares that he has no competing interests.

Author contributions

M.L. and R.N. identified the problem and designed the approach, M.L. downloaded the data, implemented the software, and performed numerical experiments. M.L. and R.N. interpreted the results and prepared the manuscript. R.N. provided funding. All authors have read and agreed to the published version of the manuscript.

Acknowledgements

This research was funded by Warsaw University of Technology grant CYBERIADA-1. We would like to thank the technical support of OpenStack at WUT.

A Parameters of scripts to measure cloud performance

Our environment can be used in a Unix-command-like manner. Below is the list of flags that were used to modify the parameters, enabling us to carry out our experiment.

Flags used in both clouds:

- `-script` - allows to specify the path to the script to be executed as soon as the instance is ready, by default `None`;
- `-number` - number of instances to create, by default 1;
- `-nodelete` - with this flag set, instances will not be deleted, they will only be created. Useful if the user does not want to delete the instances, or if he wants to delete them only after some time;
- `-file` - name of the file where the measurements results will be saved, for GCP by default `gcp_test_results.csv`, for OpenStack by default `openstack_test_results.csv`;
- `-name` - the name of the instance that will be created, by default `test_instance`. If more than one instance is created, the naming will look like `name-####`, with number of instance (0001, 0002 etc.) instead of ####;
- `-iter` - allows the user to set the appropriate number of iterations of the experiment. By default, one iteration will be performed. If the flag is set `nodelete`, the number of iterations, regardless of the given value, will
be 1 anyway - the operation of these two flags is contrary to the logic of the program (either we perform the test cycle or we do not delete the instance, but due to the uniqueness of the names, we cannot create the same instance 100 times);

- **-sleep** - using this flag we set the pause between experiments (in minutes). By default, this pause is set to 0, which means that subsequent experiments will be performed immediately after the end of the previous one;

- **-image** - in both OpenStack and Google Cloud, each instance must be based on a system image. If this value is not given, the one from the configuration file will be used. In the case of Google Cloud, the user should provide a link to the image (because it can be both a public and private image) - it is a result of the fact that the format of the links accepted by the API is strictly defined, I give an example below:

  https://www.googleapis.com/compute/v1/projects/project-name/global/images/clear-linux

  The https://www.googleapis.com/compute/v1/projects/ part is constant, then the user needs to enter a specific fragment, which can be found by going to the list of images on Google Cloud Platform and clicking the image we are interested in. In the case of OpenStack, it is enough to enter the name of one of the available images (they can be found in the Images tab in the Horizon interface).

**Flags unique to GCP:**

- **-disk** - this flag can be used to set the size of the associated disk in gigabytes, the default value is zero. The inability to set this flag in OpenStack is due to the association of the machine type with the disk (GCP does not have this association);

- **-preemptible** - this flag allows the user to set instances as preemptible, which means that Google Compute Engine can interrupt at any time (and certainly will delete them after 24 hours). Instances of this type are not always available [8]. This type of instance is not available in OpenStack;

- **-type** - this flag is used to set the machine type. If this value is not given, the value from the configuration file will be used.

**Flags unique to OpenStack:**

- **flavor** - this flag has been extracted to conform with the OpenStack nomenclature, it allows the user to specify a system image name (and thus does not functionally differ from the "-type" flag). If a flavor name is not given, the value will be taken from the configuration file.

To connect to Google Cloud one should:

- Create an account on Google Cloud.
- Create a service account and download the associated key as a JSON file to the selected folder.
• Set the variables in the configuration file according to the instructions in it.

• Invoke the program by running the gcp_connector.py file in Python with the appropriate parameters.

To connect to OpenStack one should:

• Create an account in one’s OpenStack implementation.

• Log in and download so-called „RC file”, tied to one’s account, from the Horizon interface

• Execute command (and supply the OpenStack password):

  source <RC file name>

• Invoke the program by running the openstack_connector.py file in Python with the appropriate parameters.

References

[1] Flexera 2021 state of the cloud report. Online access (21.11.2021): [https://info.flexera.com/CM-REPORT-State-of-the-Cloud](https://info.flexera.com/CM-REPORT-State-of-the-Cloud), 2021.

[2] Jupyter. Online access (25.01.2022): [https://jupyter.org/](https://jupyter.org/), 2022.

[3] S. P. Ahuja, E. Czarnecki, and S. Willison. Multi-factor performance comparison of amazon web services elastic compute cluster and google cloud platform compute engine. *International Journal of Cloud Applications and Computing (IJCAC)*, 10(3):1–16, 2020.

[4] E. Bisong. An overview of google cloud platform services. *Building Machine Learning and Deep Learning Models on Google Cloud Platform*, pages 7–10, 2019.

[5] S. Challita, F. Zalila, C. Gourdin, and P. Merle. A precise model for google cloud platform. In 2018 IEEE International Conference on Cloud Engineering (IC2E), pages 177–183. IEEE, 2018.

[6] O. Foundation. Introduction to openstack. Online access (21.11.2021): [https://docs.openstack.org/security-guide/introduction/introduction-to-openstack.html](https://docs.openstack.org/security-guide/introduction/introduction-to-openstack.html), 2021.

[7] Google. Deleting an instance. Online access (16.12.2021): [https://cloud.google.com/compute/docs/instances/deleting-instance](https://cloud.google.com/compute/docs/instances/deleting-instance), 2021.

[8] Google. Preemptible vm instances. Online access (12.12.2021): [https://cloud.google.com/compute/docs/instances/preemptible](https://cloud.google.com/compute/docs/instances/preemptible), 2021.

[9] Google. Understanding and profiling gce cold-boot time. Online access (12.12.2021): [https://cloud.google.com/compute/docs/machine-types](https://cloud.google.com/compute/docs/machine-types), 2021.
[10] A. A. i John Kendall i Yevgeniy Miretskiy i Jessica Edwards. 2021 cloud report. Online access (06.12.2021): https://www.cockroachlabs.com/guides/2021-cloud-report/, 2021.

[11] P. Leitner and J. Cito. Patterns in the chaos—a study of performance variation and predictability in public iaas clouds. ACM Transactions on Internet Technology (TOIT), 16(3):1–23, 2016.

[12] K. Lucas. byte-unixbench. Online access (17.12.2021): https://github.com/kdlucas/byte-unixbench, 2021.

[13] M. Mao and M. Humphrey. A performance study on the vm startup time in the cloud. In 2012 IEEE Fifth International Conference on Cloud Computing, pages 423–430. IEEE, 2012.

[14] C. McAnlis. Understanding and profiling gce cold-boot time. Online access (29.11.2021): https://medium.com/google-cloud/understanding-and-profiling-gce-cold-boot-time-32c209fe86ab, 2017.

[15] P. Mell, T. Grance, et al. The nist definition of cloud computing. 2011.

[16] J. P. Mullerikkal and Y. Sastri. A comparative study of openstack and cloudstack. In 2015 Fifth International Conference on Advances in Computing and Communications (ICACC), pages 81–84. IEEE, 2015.

[17] H. Nawaz, G. Juve, R. F. Da Silva, and E. Deelman. Performance analysis of an i/o-intensive workflow executing on google cloud and amazon web services. In 2016 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW), pages 535–544. IEEE, 2016.

[18] Z. Ou, H. Zhuang, J. K. Nurminen, A. Ylä-Jääski, and P. Hui. Exploiting hardware heterogeneity within the same instance type of amazon ec2. In 4th USENIX Workshop on Hot Topics in Cloud Computing (HotCloud 12), Boston, MA, June 2012. USENIX Association.

[19] A. Paradowski, L. Liu, and B. Yuan. Benchmarking the performance of openstack and cloudstack. In 2014 IEEE 17th International Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing, pages 405–412, 2014.

[20] T. Rosado and J. Bernardino. An overview of openstack architecture. In Proceedings of the 18th International Database Engineering & Applications Symposium, pages 366–367, 2014.

[21] D. Steinmetz, B. W. Perrault, R. Nordeen, J. Wilson, and X. Wang. Cloud computing performance benchmarking and virtual machine launch time. In Proceedings of the 13th annual conference on Information technology education, pages 89–90, 2012.

[22] A. Vogel, D. Griebler, C. A. Maron, C. Schepke, and L. G. Fernandes. Private iaas clouds: a comparative analysis of opennebula, cloudstack and openstack. In 2016 24th Euromicro International Conference on Parallel, Distributed, and Network-Based Processing (PDP), pages 672–679. IEEE, 2016.
[23] S. Yadav. Comparative study on open source software for cloud computing platform: Eucalyptus, openstack and opennebula. *International Journal Of Engineering And Science*, 3(10):51–54, 2013.