Cloud Performance Tests - Latency and Throughput

Mafalda Matos
CERN, Geneve, Switzerland

Abstract
The goal of this project is to benchmark and compare the performances of load balancers in the areas sdn1 and octavia_t, based on latency and throughput. Results were obtained for the cases of 5 Clients with N Servers and N Clients with 50 Servers, with N being 1, 5, 10, 25 and 50. From these results, it seems that: Latency (the average) seems to be similar in both regions; Throughput (the average) seems to be smaller in the octavia_t region; 25.000 was the observed limit for Requests per Second. The conclusions are then that the tests could be further validated by rerunning them, as well as, going forward, that the team should be mindful of the limitations found and possibly investigate where the performance issues come from if they start having an impact on real use cases.

Keywords
Cloud Infrastructure; Loadbalancing; Benchmarking

1 Introduction

The goal of this project is to benchmark and compare the performances of load balancers in the areas sdn1 and octavia_t, based on Latency and Throughput.

This section will now briefly introduce each of the necessary concepts to understand the project, starting with the load balancers.

A load balancer is a device that stands between a user and a server group, as seen in figure 1, which then distributes the network traffic between the servers in a way that maximizes speed and resource utilization. [1]

In the case of this project, software load balancers were placed in the regions of sdn1 and octavia_t.

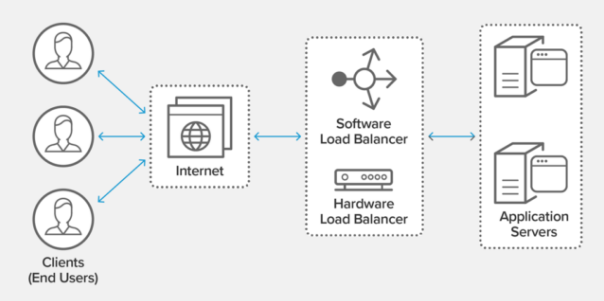

Figure 1: Load balancing diagram. Source: NGINX

There are several algorithms that determine the best way to do the distribution, but the one that was used for this project is Round Robin. In this algorithm, the requests are distributed between the application servers in a round robin way, that is, in some sequential way.

Latency refers to the delay that happens between when a user takes an action on a network or web application and when it reaches its destination, which is measured in milliseconds. [2]

Throughput measures the average rate at which messages successfully arrive at the required destination, a more specific example can be seen in figure 2. It can be used for different metrics, but the one the project was focused on was requests per second.

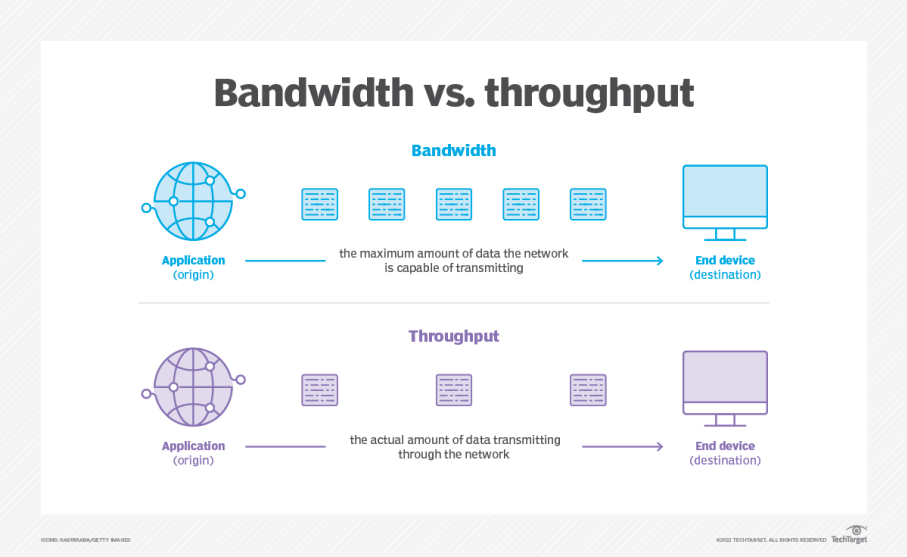

Figure 2: Bandwidth vs. throughput diagram. Source: TechTarget

As mentioned previously, sdn1 and octavia_t were the main subject of comparison. These are two different load balancer implementations - sdn1, which is based on TungestenFabric using haproxy running on the physical machine, and octavia_t, which is based on Openstack Octavia, running haproxy in a VM.

2 Setup

First, before testing the different regions of load balancers, some general tests with different regions of VMs within sdn1 were done. This was done in order to build a general script that could work in different situations and to obtain preliminary results.

Without going into too much detail, these tests were inconclusive since running the same test twice, one immediately after the other, would give contradictory results. This will be brought up again later in the Analysis section.

Moving on to the project itself, 4 types of tests were defined:

- 1 client with 1 server
- N clients with 1 server
- 1 client with N servers
- N clients with N servers

In this list, the "client" and "server" represent any of the clients or servers available so that all of them have been tested. This means that every possible combination is included in this set of tests. Moreover, "N servers" represent a load balancer with N servers.

In this project, the numbers to be used for clients and servers were 1, 5, 10, 25 and 50, with every combination being explored, as can be seen in image 3.

The programs chosen for the setup were pdsh and wrk2. In order, pdsh is a parallel ssh program that allows you to run commands at the same time in multiple hosts. It will be used to connect to the

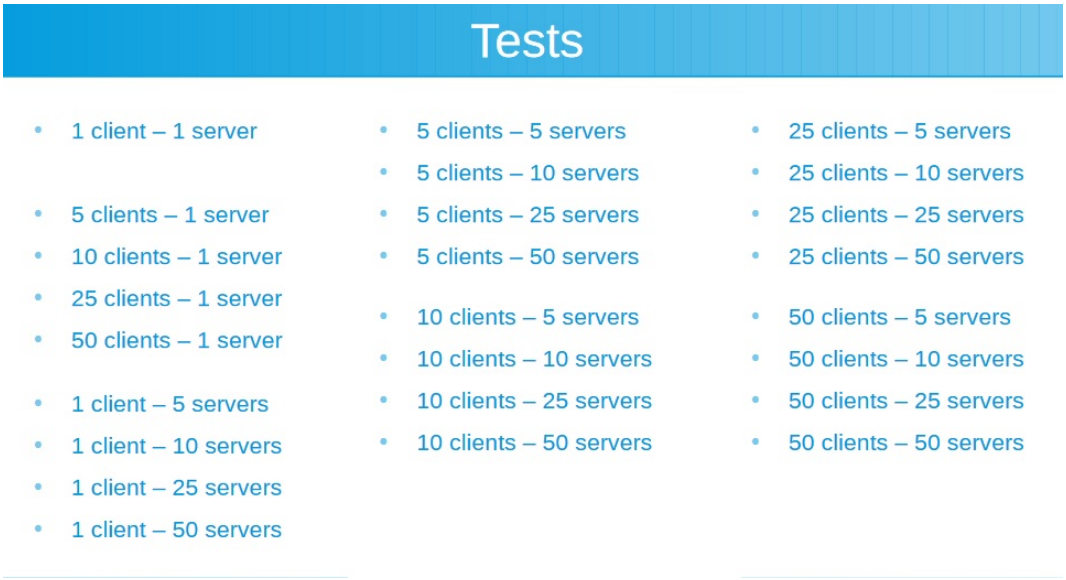

Figure 3: All test combinations

client(s) from an outside VM which we will call Master VM.

Next, wrk2 is a load generator program that returns latency, requests per second, errors, etc. from a test of a server. This is what will be run to test our servers. wrk2 has the following parameters, which were used in this project with the values presented:

Table 1: wrk2 parameters

Parameter	Meaning	Values Used
t	Number of Threads	kept at 1
c	Number of Connections	10, 100 or 1.000
d	Time the test is run for [a]	kept at 30s
R	Requests per Seconds Expected	10, 100, 1.000, 10.000, 100.000 or 1.000.000

[a] This value should be at least 20s to be statistically relevant

It's worth noting that every test presented previously will also be run with all possible combinations of these values.

The reason why wrk2 was chosen for this project specifically is because of an effect called coordinated omission. Coordinated omission happens in high latency responses when the load generator ends up coordinating with the server and avoiding measurements during these times. This leads to results that appear to have lower latencies than they actually do. [4]

To create the VMs, the flavor m2.small (1875 MB RAM and 10 GB of disk space) and image CS8 - x86_64 (Linux with CentOS 8 and a 64 bit architecture) were used. More information on the flavor and image of the VMs is available in the Cloud Documentation. [5]

These VMs were all created in the gva-critical region. In terms of setup of the VMs themselves, the following steps were taken:

Server VM

- Install httpd (apache http server)
- Create healthcheck
- Create index.html file
- Start server, add to firewalld
- Add server to load balancer

Client VM

- Install git, gcc and openssl-devel
- Clone wrk2 repository
- Make wrk2

It's worth pointing out that the index.html file represents a simple website (seen in image 4) which occupies 81 bytes of space. This is important specifically for the bandwidth throughput tests because to really push the capacity of the load balancers a big file should be used. Since the goal of this project is to just have an idea of the differences between load balancer regions, and that users typically don't pass big files through the load balancers, this should be perfectly fine.

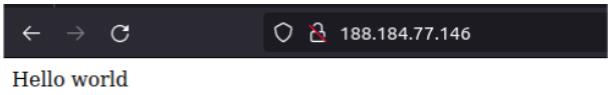

Figure 4: index.html

As for the load balancers themselves, the Round Robin algorithm was used for both. To be able to compare the them, it also makes sense to use the same VMs for both. The biggest difference between the load balancer in sdn1 and the one in octavia_t is that the latter one is in the critical region, whereas the first isn't. More information about the load balancers is presented in image 5.

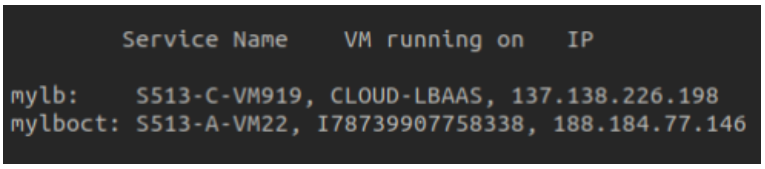

Figure 5: Load balancers information

As mentioned previously, to be able to run the commands from different Client VMs, a Master VM was created. This VM is meant to use pdsh to remotely run the wrk2 commands from the Client VM(s) in an automatized way.

This setup can be used in two ways. The first is to connect a single client with a server or load balancer, in which case the tests are run sequentially. The second is to connect several clients with a server or load balancer, in which case the tests are still run sequentially for different configurations, but run in parallel for the same configuration in all the clients at once. This allows all the clients to be connected to the servers at once, as is necessary for the N clients - server(s) tests.

A scheme of the full setup can be seen in image 6.

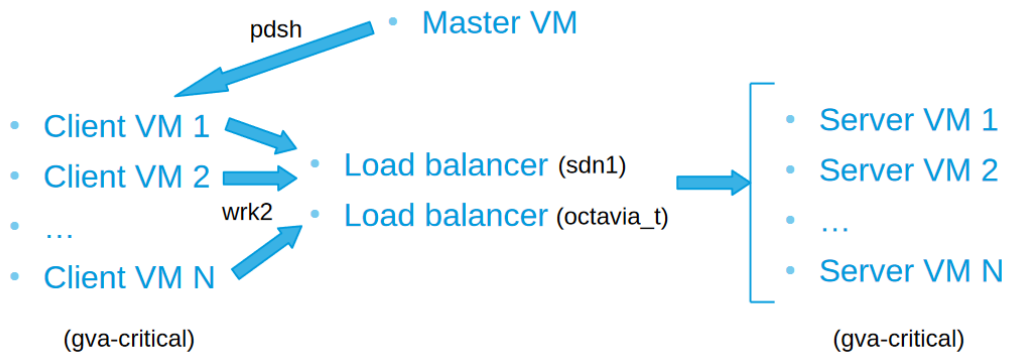

Figure 6: Scheme of full setup

3 Problems encountered

There were some problems encountered in the making of this project.

The first one had to do with the firewall of the server VMs not allowing connections to be established with any other VMs. This was noticed since the commands ssh and ping worked but not curl. The solution was simply adding the port we were using as an exception to the firewall, as seen in the steps detailed in the Setup section in the Server VM list.

The second problem had to do with using `python http.server` to setup the website in the server VMs as way more latency was observed even in the easy tests, for example, 5s of latency for 10 requests per second. Since there were no indicators of why this might be happening (outside of the server choice), the switch to apache http server was done and the issue was fixed. The reason for the issues here is that `python http.server` is a non-production server which is not optimized for performance.

Lastly, when running the tests, the load balancers and the VMs would run out of disk space because of logs. This wouldn't affect the progress of the test until they would run out of space, after which they could not continue. In order to fix the issue, log rotation was implemented.

4 Results

Because there was such a large amount of tests to be potentially run (and a limited amount of time), the 5 clients with server(s) and the client(s) with 50 servers tests were prioritized. These alone should provide enough insights to then further explore as necessary.

In a single plot all of the different wrk2 configurations are present, with the common denominator being the setup of N clients with N servers. This also includes the results of the tests that connect the individual clients used with individual server VMs used in the load balancer. This should make no difference since the load balancers use the same VMs.

In image 7, the setup of 5 clients with 5 servers is represented, with Throughput plots on top, Latency plots on the bottom and scatter plots vs Requests per Second on the left and boxplots on the right.

In image 8, there is a comparison made between the setups of 5 clients with 50 servers and 10 clients with 50 servers. The layout is Latency plots with 5 Clients on the top and 10 Clients on the bottom, with scatter plots vs Requests per Second on the left and boxplots on the right.

Finally, in image 9 the same comparison is made, this time only comparing the boxplots of the Throughput of the 5 clients (on the top) and 10 clients (on the bottom).

There are both vertical and horizontal error bars represented in all the scatter plots.

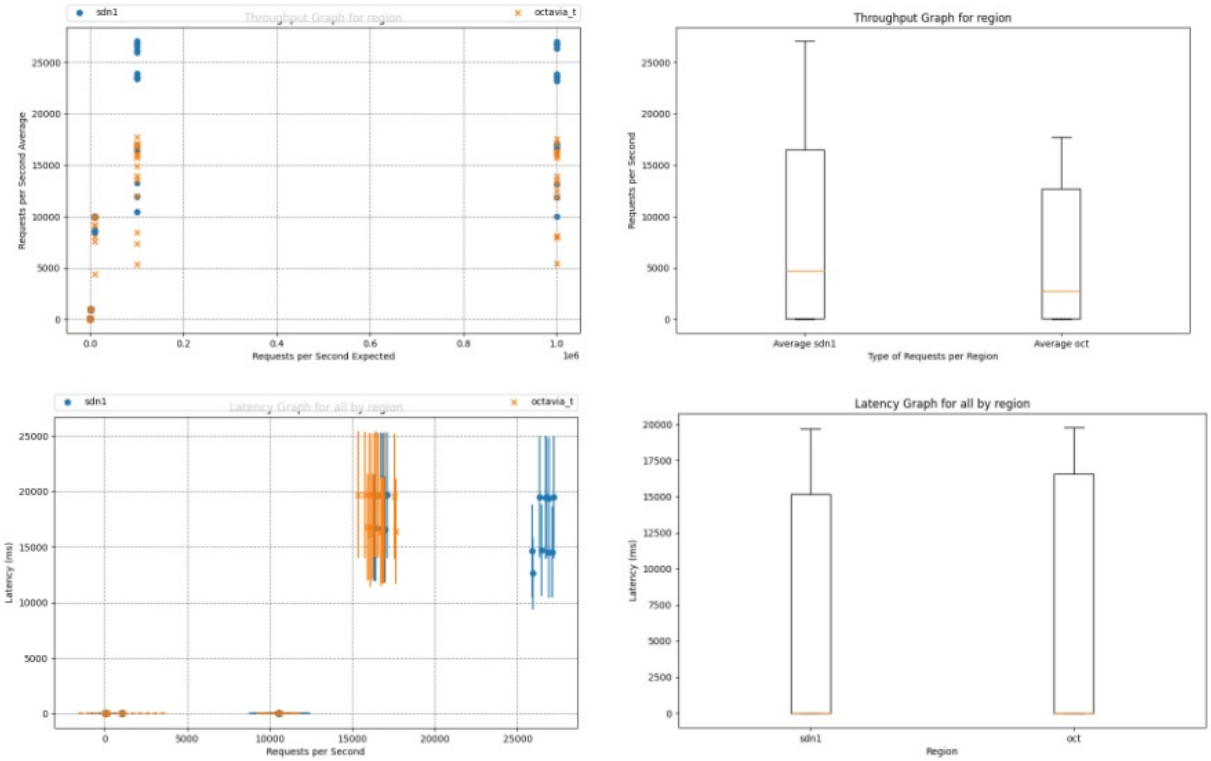

Figure 7: 5 Clients with 5 Servers - Throughput plots on top, Latency plots on the bottom - Scatter plots vs Requests per Second on the left, Boxplots on the right

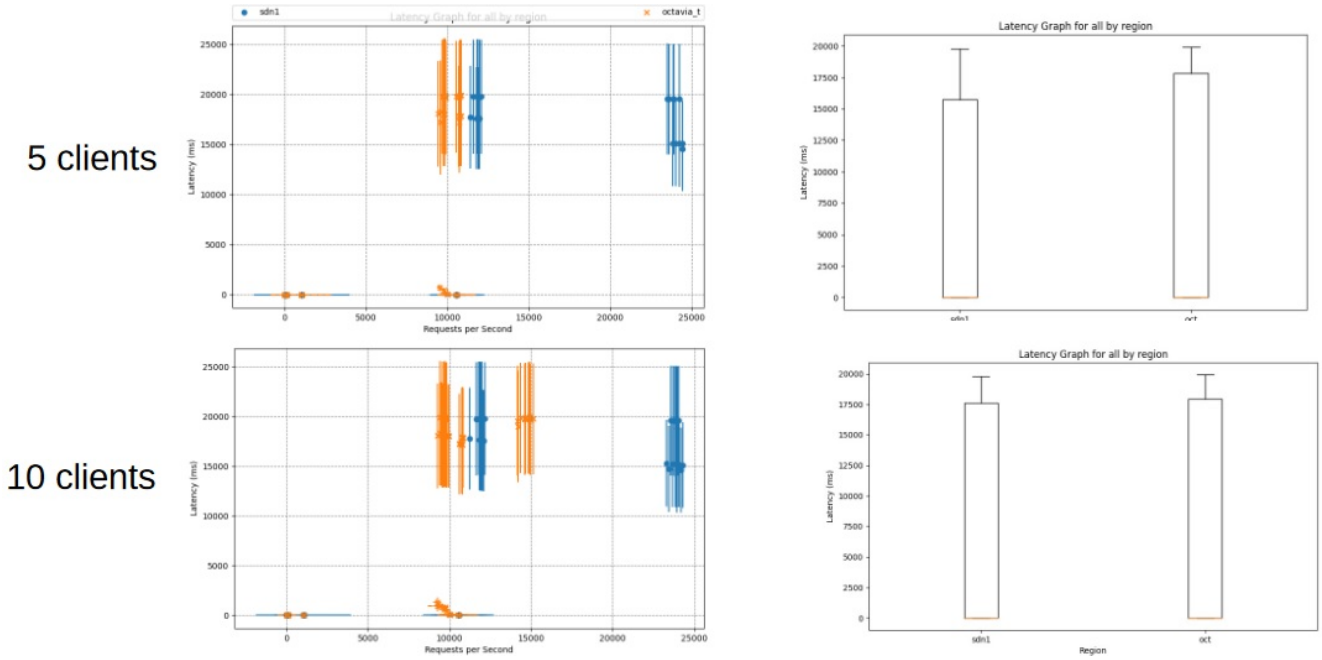

Figure 8: Latency plots with 5 Clients on the top and 10 Clients on the bottom - Scatter plots vs Requests per Second on the left, Boxplots on the right

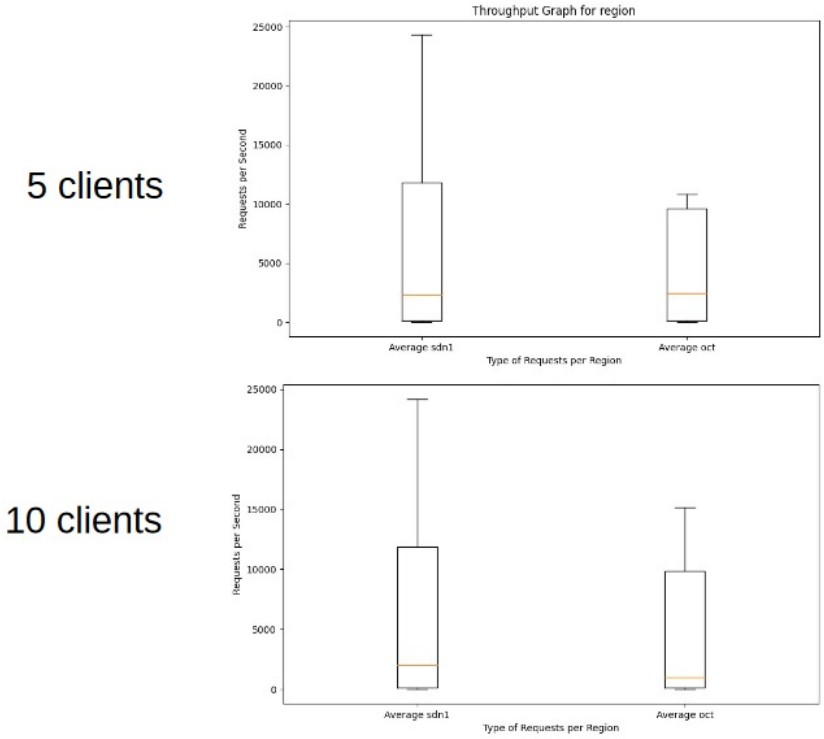

Figure 9: Throughput boxplots with 5 Clients on the top and 10 Clients on the bottom

For the Latency plots, the x axis represents the Requests per Second, while the y axis represents the Latency in milliseconds.

For the Throughput plots, the x axis represents the Requests per Second Expected (defined in the wrk2 command of the test run), while the y axis represents the Requests per Second on Average (the result obtained from the test run).

5 Analysis

From the results obtained it can be observed that:

- Latency (the average) seems to be similar in both regions
- Throughput (the average) seems to be smaller in the octavia_t region
- 25.000 was the observed limit for Requests per Second in all of the tests

Some important things to note:

- Because it's very hard, if not impossible, to track every process a machine is doing, it's very likely that there will be background noise that will impact results in unpredictable ways. This noise is possibly the same reason why the tests for different regions within sdn1 mentioned in the beginning of the Setup Section were inconclusive.
- The plots don't show nan (missing or undefined data) results which were obtained in some of the tests run. This could be misleading, since some of the plots end up representing more information than others as can be observed in the Latency scatter plots in image 8. There was no reason found for getting nan results in the wrk2 documentation or online.

It's important to take these results with a grain of salt, as they are only representative of what is actually going on.

6 Conclusions

Given the observations made previously, it can be concluded that:

- The tests could be further validated by rerunning them and seeing if any background noise impacted them, i.e. comparing the new results obtained with the ones obtained here.
- Going forward, the team should be mindful of the limitations found and possibly investigate where the performance issues come from if they start having an impact on real use cases.

Acknowledgements

I would like to thank my supervisors, Daniel Failing and Spyridon Trigazis, as well as the rest of the team for the opportunity, the help and the coffee breaks provided.

Bibliography

[1] What Is Load Balancing?, Amazon
`https://aws.amazon.com/what-is/load-balancing/`

[2] What Is Latency and How to Reduce Latency?, Fortinet
`https://www.fortinet.com/resources/cyberglossary/latency`

[3] What is Throughput? An Explanation Of Throughput, PacGenesis
`https://pacgenesis.com/what-is-throughput-an-explanation-of-throughput/`

[4] wrk2 GitHub Repository, GilTene GitHub
`https://github.com/giltene/wrk2`

[5] CERN OpenStack Private Cloud Guide
`https://clouddocs.web.cern.ch/index.html`

[6] GitLab Repository of this project, GitLab
`https://gitlab.cern.ch/cloud-infrastructure/summer-students/2023-lbaas-benchmarks`

Annexes

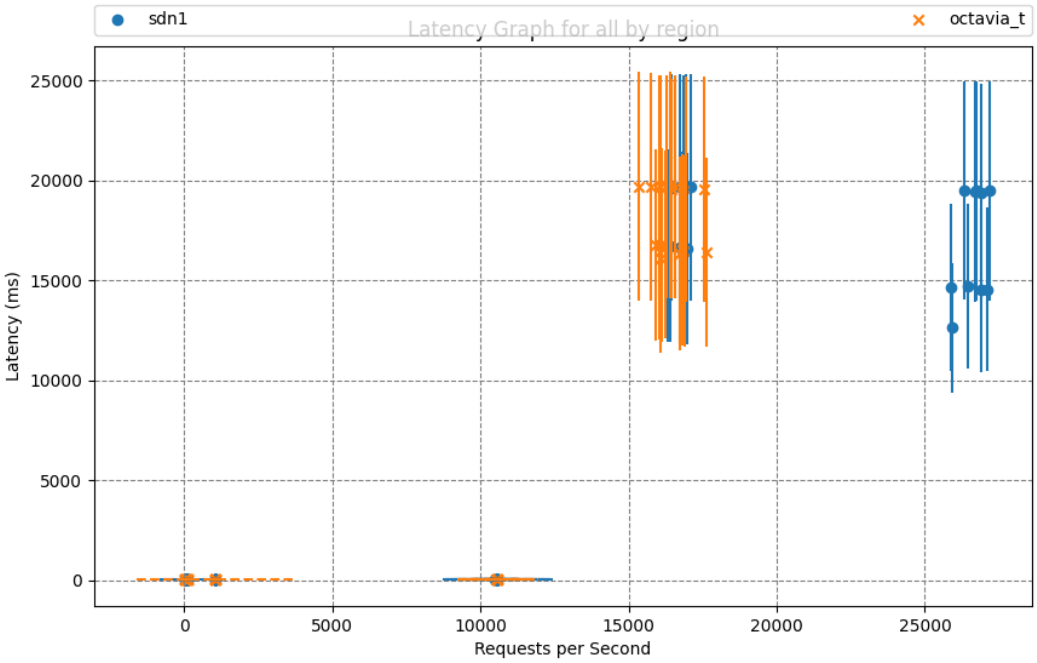

Figure 10: 5 Clients with 5 Servers - Latency vs Requests per Second

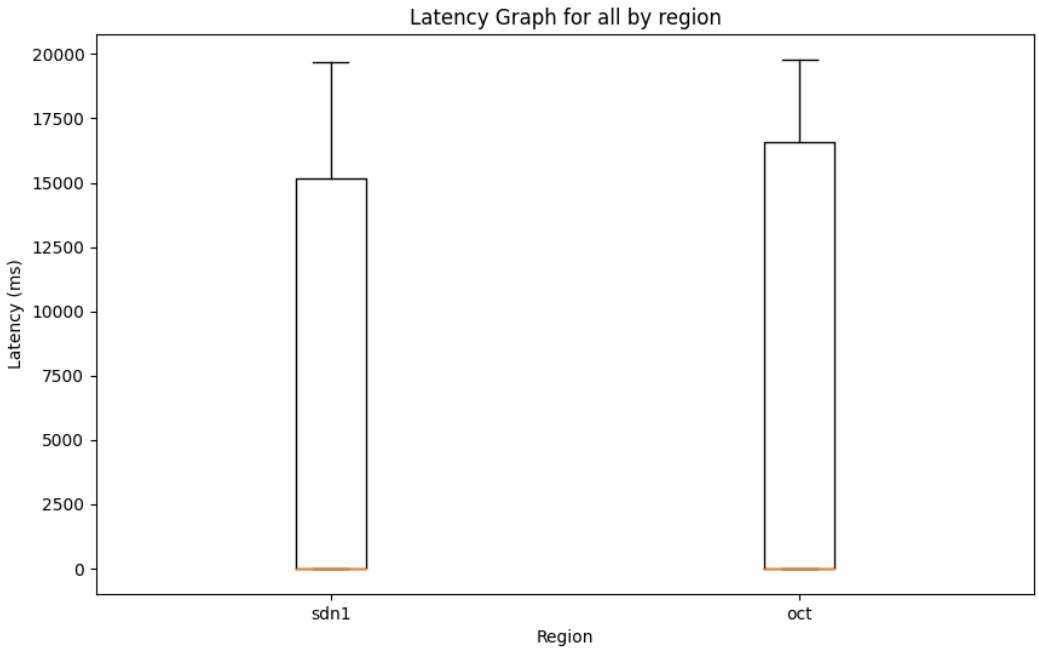

Figure 11: 5 Clients with 5 Servers - Latency Boxplot

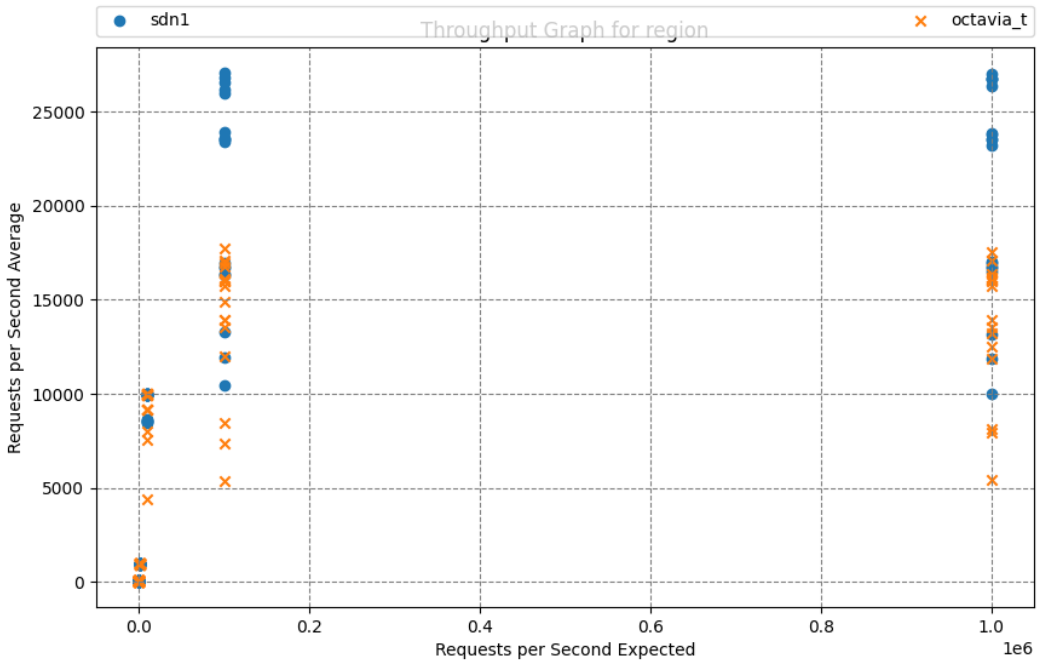

Figure 12: 5 Clients with 5 Servers - Throughput - Requests per Second Average vs Requests per Second Expected

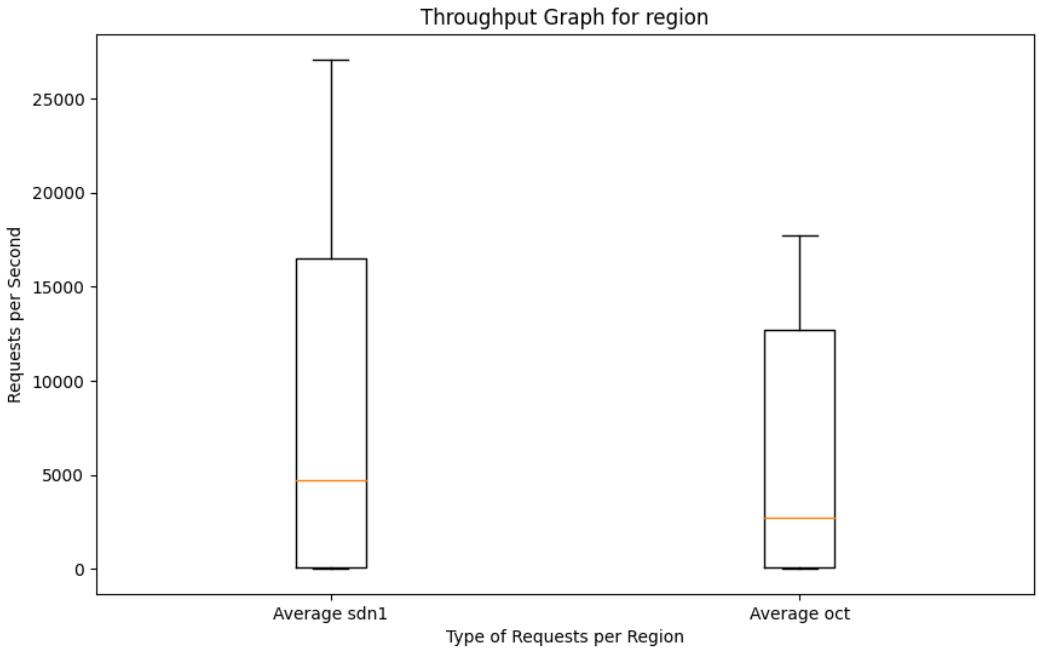

Figure 13: 5 Clients with 5 Servers - Throughput - Requests per Second Average Boxplot

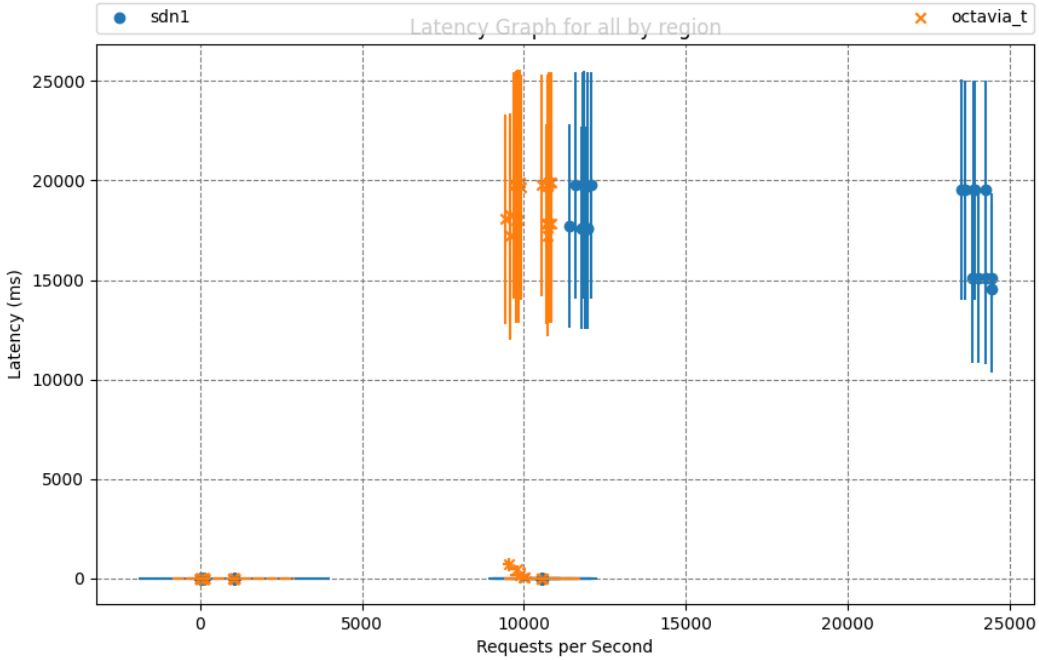

Figure 14: 5 Clients with 50 Servers - Latency vs Requests per Second

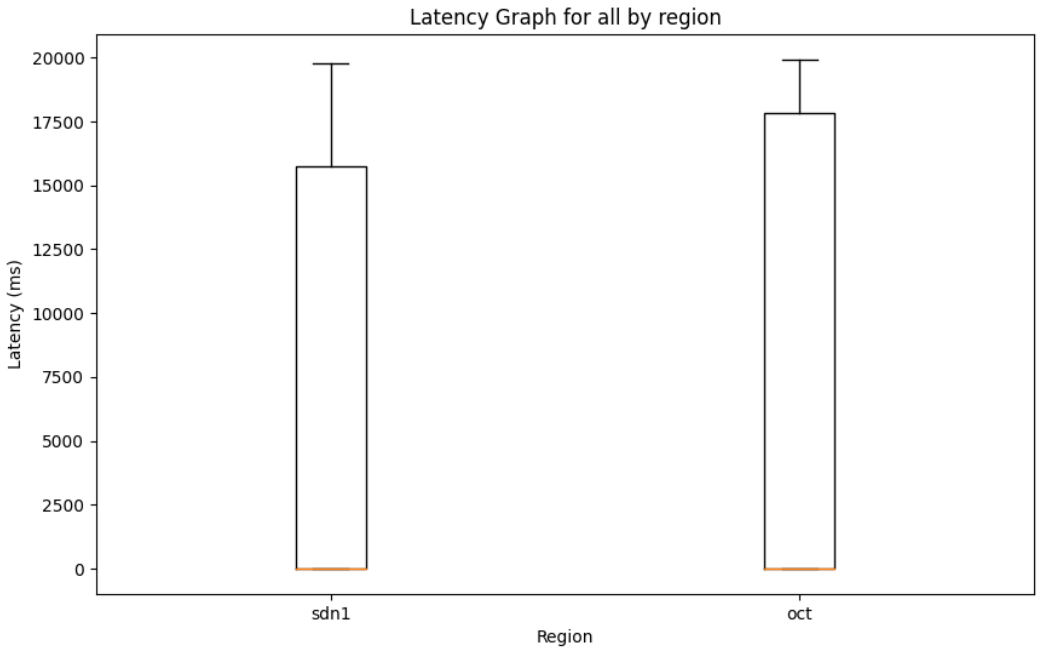

Figure 15: 5 Clients with 50 Servers - Latency Boxplot

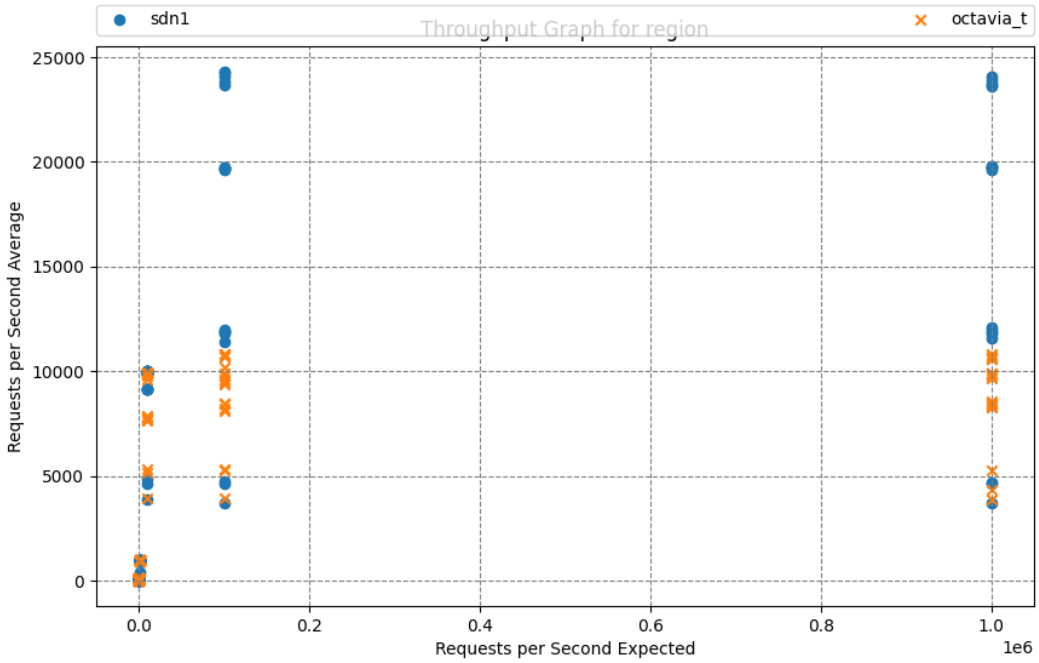

Figure 16: 5 Clients with 50 Servers - Throughput - Requests per Second Average vs Requests per Second Expected

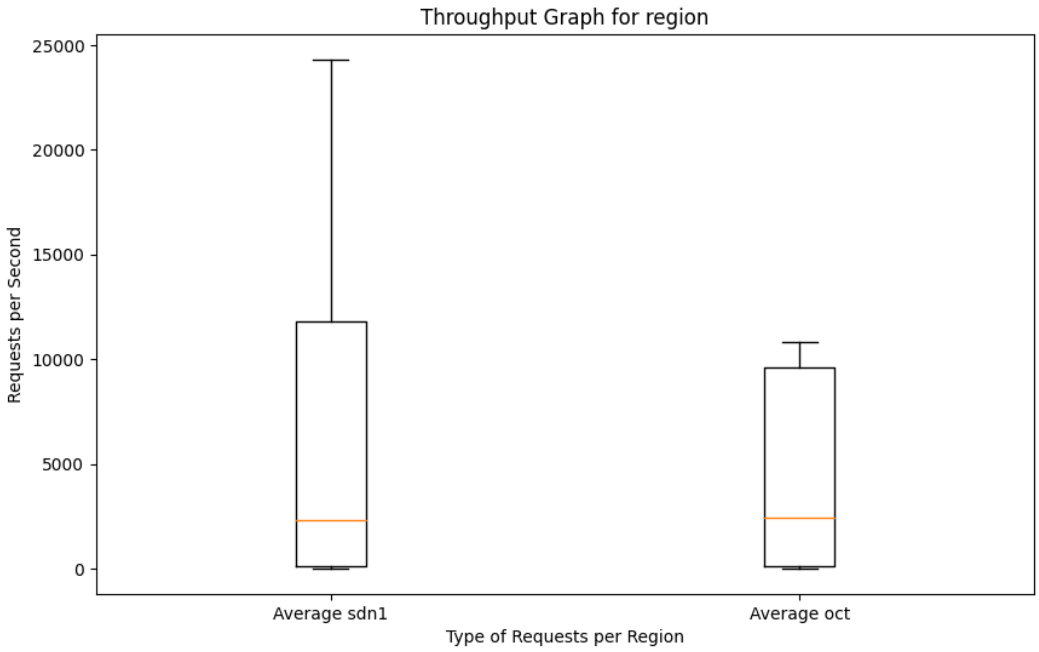

Figure 17: 5 Clients with 50 Servers - Throughput - Requests per Second Average Boxplot

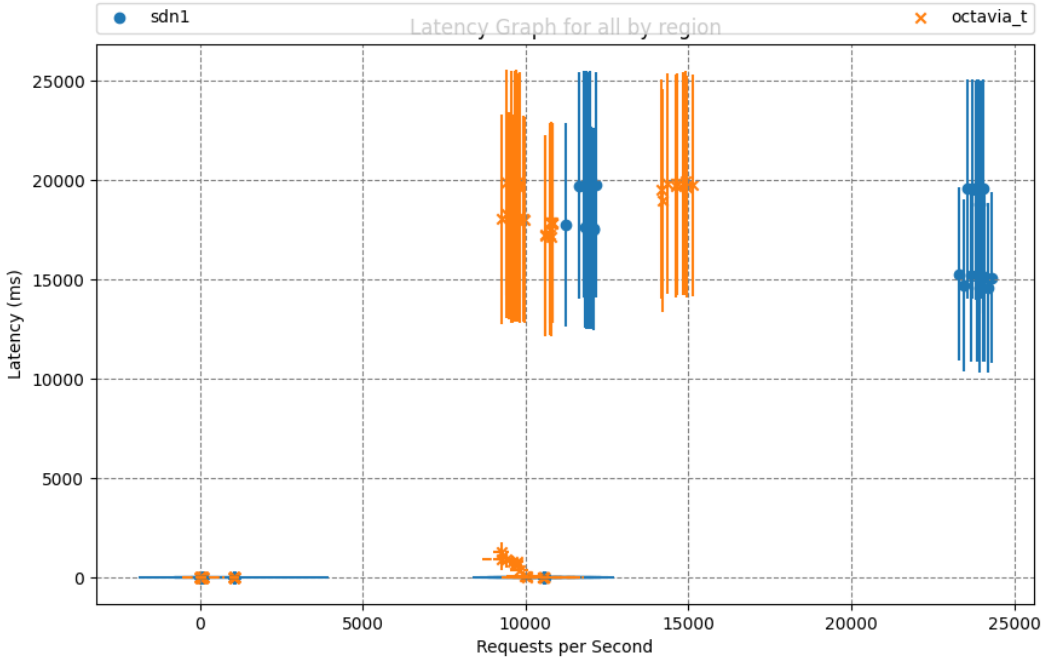

Figure 18: 10 Clients with 50 Servers - Latency vs Requests per Second

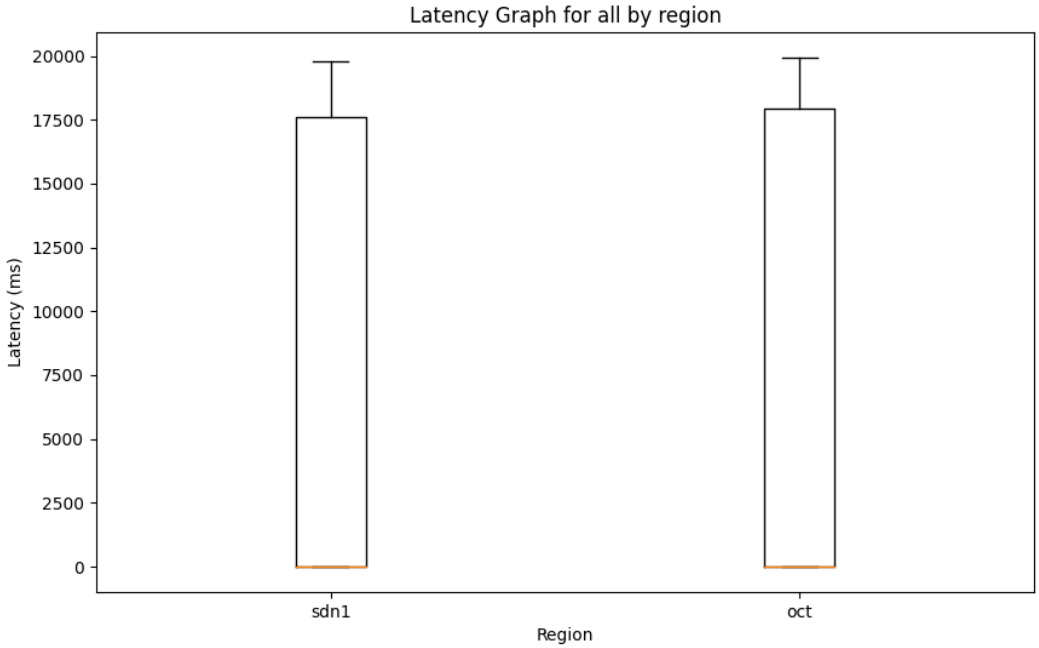

Figure 19: 10 Clients with 50 Servers - Latency Boxplot

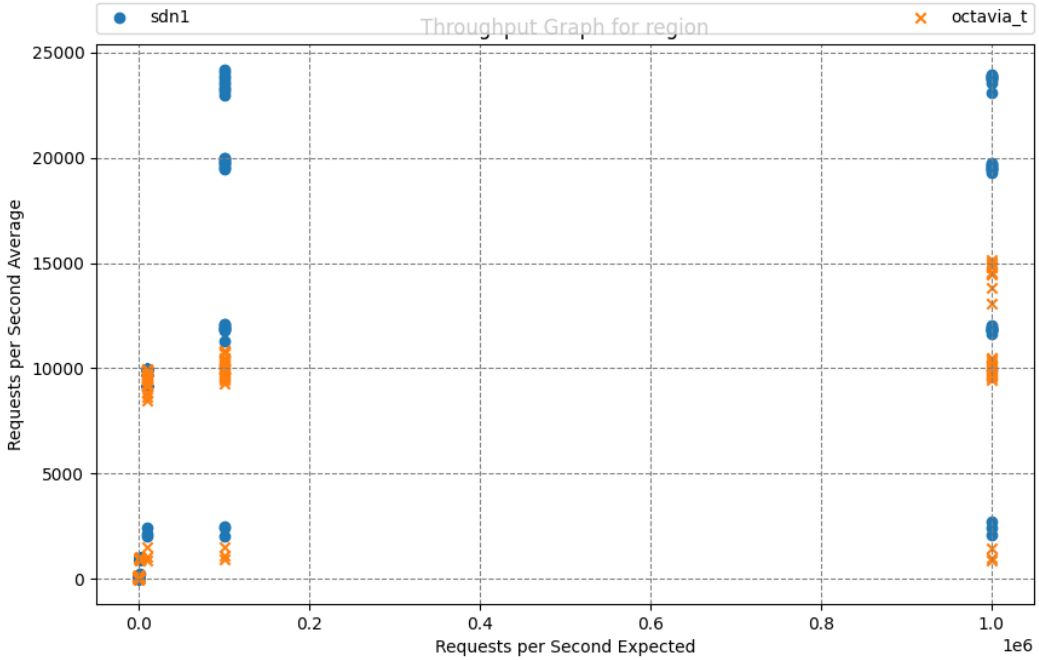

Figure 20: 10 Clients with 50 Servers - Throughput - Requests per Second Average vs Requests per Second Expected

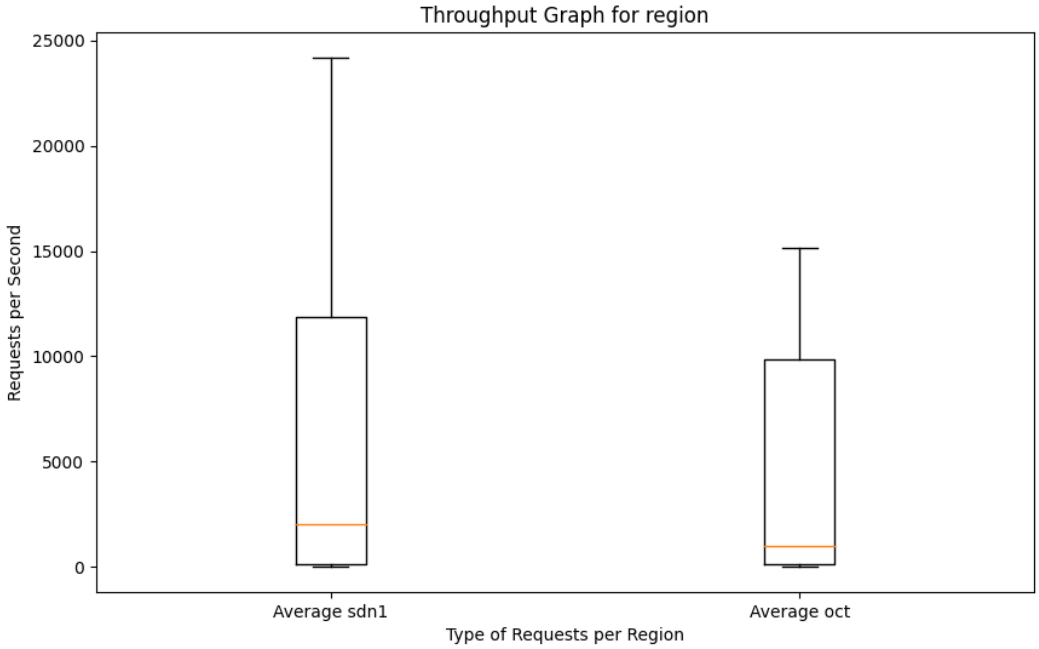

Figure 21: 10 Clients with 50 Servers - Throughput - Requests per Second Average Boxplot