Comparative Analysis of API Performance Metrics Across Varied VPS Configurations

Mihir Sawant
A048, MCA
NMIMS University

Yash Chowdhari
A010, MCA
NMIMS University

Suresh Chaudhary
A007, MCA
NMIMS University

***Abstract*—The selection of a cloud service provider is a critical decision that significantly impacts application performance. This research paper examines and compares various cloud providers based on essential performance metrics, including compute usage, response time, and bandwidth. The research objectives encompass a comprehensive comparison of select cloud providers, an evaluation of Node.js API performance for various tests.**

Keywords—cloud-computing, metrics, vps, nodejs

I. Introduction

Cloud computing has transformed the way businesses and individuals access and manage data and applications. The selection of an appropriate cloud service provider has emerged as a critical decision. This research paper explores and compares different cloud service providers in the context of speed, response time, and bandwidth. This exploration aims to assist you in selecting a provider that aligns with your specific requirements, taking into consideration other factors such as cost, instance creation time, time required for instance termination, feasibility, accessibility, scalability, and other configurations.

// put any special intro points of suresh and yash...

For this paper we have taken three well known cloud service providers namely, Amazon Web Services (AWS), Digital Ocean and Microsoft Azure

We have also conducted a test on AWS Lambda, which is widely recognized for its robust serverless computing capabilities. Additionally, we have documented key metrics for comparison with a standard EC2 instance provided by AWS, a discussion of which will be presented in later sections.

For the testing phase, we have chosen Node.js, as it is widely used for building high I/O and throughput-based applications. We will provide links to the repositories in the references section, so please feel free to check and explore the code.

In the first section we mainly check the compute and hardare speed, the app consist of test-cases of nod.js code. Where one function runs and prints even number and other function does same for odd numbers

In the second testcase, we test of unexpected and unwanted errors situations in code like race-conditions, deadlocks Intentionally wrote code about deadlock for callback-hells, race-condition for change of answers

Thirdly we have an app in which are ar 4 sub tasks for that api to run, being a parent it is very difficult, basically it has a big json file as a dataset which contains the “make-up data” so their are four api’s first one to fetch all the data and return which is the slowest and other with finding all brandNames then other with fidning single brandName and last one for taking specific property from each record like productCategory and adding details and storing it in other json file.

In the fourth and last section of test, we use an api where we send some big image file and it would convert into (250, 250) and apply greyscale and all, so that we can get idea about this api. it uses multer for file uploads with node.js and api only accepts images as data.

So these was brief about our apis, we also have containarized version of the app in docker also, and in the index.js all the tests are imported and called directly so that we can easily package it and run on any vps easily.

II. Experimental Setup

A. *Common things and standard in all VPS*

1) As we mentioned that we are using three different providers but to keep things simple and avoid confusions, we stick to some common properties in each.

2) The Operating system used in all three is Ubuntu but the version might be different like 22.04 and 20.04 LTS releases.

3) To keep things standard the development is done on node version 18 and for testing also we used node 18 version only.

B. *List of different configurations of VPS that we have tried and tested*

1) ***Digital Ocean :*** *// take details from yash*

2) ***Amazon Web Services :*** *In AWS the service used is the t2 micron which is the free version available with 1 CPU, 1GB memory and basic compute which is okish for deploying basic apps for testing.*

But for AWS it provides a list of regions to choose from, for testing purposes we have picked two regions and then deployed two instances there which are US EAST very far to us and MUMBAI SOUTH which is very close to us, so for geographical differences.

And for Lambda it needs no config or any ssh it directly runs the function, so nothing special to add about it as config.

*3) **Microsoft Azure:** In azure also we have two instances with different configs namely:*

- 1 GB Memory, 1 Intel vCPUs, Max IOPS – 320, Data Disk – 2, 4GB Temp Storage, Ubuntu 20.04 (LTS) x64 Region : (US) East – US
- 14 GB Memory, 4 Intel vCPUs, Max IOPS – 12800, Data Disk 16, 28 GB Temp Storage, Ubuntu 22.04 (LTS) x64 Region : (Asia Pacific) Central India

So we can also see the difference of geography locations here also for testing it in different scenerios.

III. Results and Discussions

Now here we will put our testing metrics here one by one for each VPS with both containerized and non-containerized categories:

1) Amazon Web Services (AWS):

- **Non Containerized**

1. MUMBAI SOUTH REGION

# test_cases	Response Time in ms	Bandwidth in Bps	Cpu usage
TEST 1	2454.5013329982758	-	1.41%
TEST 2	110 output	-	0.53%
TEST 3.1	117.61472399905324	23779536.02	1.56%
TEST 3.2	6.681744998320937	7765837.93	0.14%
TEST 3.3	54.15848300047219	94.80	1.01%
TEST 3.4	36.64969109987027	497.76	1.45%
TEST 4	4745.453149998561	5.91	2.76%

these are some stats of mumbai region, if we compare it with US-EAST region then we can see that response time is quite less and also bandwidth in some cases.

2. US-EAST REGION

# test_cases	Response Time in ms	Bandwidth in Bps	Cpu usage
TEST 1	2857.193567999988	-	1.46%
TEST 2	110 output	-	0.52%
TEST 3.1	142.0795630000066	18522060.14	1.56%
TEST 3.2	6.647331000072882	7276874.36	0.145 %
TEST 3.3	52.829885000013746	114.86	1.00%
TEST 3.4	44.85120399994776	436.81	1.47%
TEST 4	4745.453149998561	5.91	2.76%

the bandwidth is more or less same also cpu usage is similar as compared to that of mumbai region,but response time is more.

- **Containerized**

3. MUMBAI SOUTH REGION

# test_cases	Response Time in ms	Bandwidth in Bps	Cpu usage
TEST 1	3005.5510360002518	-	1.48%
TEST 2	110 output	-	0.55%
TEST 3.1	118.96400400064886	28000663.06	1.59%
TEST 3.2	37.33544999919832	5391435.40	0.17%
TEST 3.3	147.488119000569	86.56	1.03%
TEST 3.4	78.03860499896109	188.24	1.48%
TEST 4	4735.097480999306	5.94	3.00%

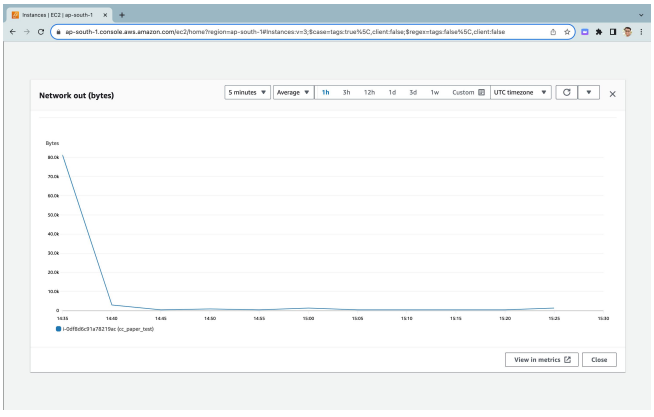
if we compare contanierized then the response time is more as that of normal one, but the bandwidth is less if we see test cases 3.1, 3.2, 3.3 and 3.4.

4. US-EAST REGION

# test_cases	Response Time in ms	Bandwidth in Bps	Cpu usage
TEST 1	3347.404606999946	-	1.46%
TEST 2	110 output	-	0.53%
TEST 3.1	128.9467579999473	21064899.55	1.58%
TEST 3.2	36.02120000007562	5103440.08	0.17%
TEST 3.3	158.32888499996625	73.81	1.01%
TEST 3.4	280.52011499996297	50.58	1.47%
TEST 4	4528.548904999974	6.20	2.76%

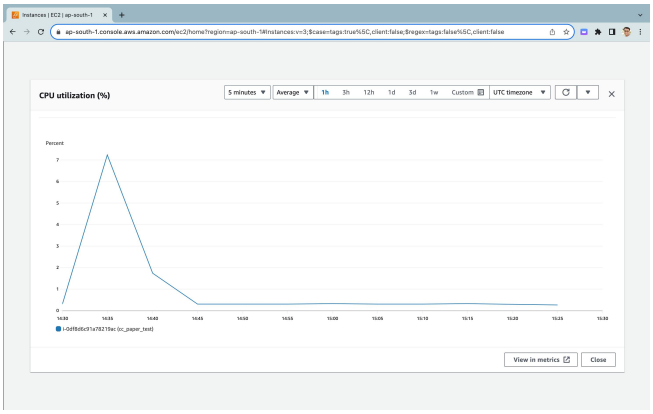
similar results for this region also, bandwidth is less as compared to normal one and cpu usage is more or less same only.

monotoring metrics from aws dashboard:



AWS network monotoring logs which are quite high but as we call it multiple time it gradually decreses.So we can see that caching is used there.

cpu:



AWS CPU utilization spike up at initial use for testcases 1 and 2 but later on got normal and stable.

2) Digital Ocean

Containerized

5. VPS 1 - 2GB Memory, 2 INTEL CPUs,25 GB DISK

# test Cases	Time in ms	Bandwidth in bps	CPU USAGE
TEST 1	2214.7933082580566	-	1.54%
TEST 2	110	-	-
TEST 3.1	96.52122116088867	34922760.60 Bps	2.01%
TEST 3.2	27.319950103759766	3904159.04 Bps	0.29%
TEST 3.3	87.79843711853027	258.74 Bps	0.97%
TEST 3.4	24.525985717773438	126.60 Bps	2.05%
TEST 4	5045.1570472717285	5.56 Bps	10%

So we can observer that 2GB memory takes more time if we compare it with AWS Mumbai region.

6. VPS 2 - 8 GB Memory, 160 GB DISK

Test Cases	Time in ms	Bandwidth in bps	CPU USAGE
TEST 1	2588.4153209999204	-	0.30%
TEST 2	110	-	-
TEST 3.1	119.40210400009528	28775456.03 Bps	2.01%
TEST 3.2	12.310721999965608	3444572.23 Bps	0.29%
TEST 3.3	186.93602499971166	142.05 Bps	0.36%
TEST 3.4	72.6384959998540 6	126.60 Bps	2.05%
TEST 4	4642.838243999984	6.05 Bps	6%

The only difference is that Test case 3.2 and 4 are a little faster than that of VPS 1. Also bandwidth taken is quite less in some cases.

7. VPS 3 16 GB Memory, 4 AMD CPUs, 200 GB Disk

Test Cases	Time in ms	Bandwidth in bps	CPU USAGE
TEST 1	3323.0729210000136		0.72%
TEST 2	110	-	-
TEST 3.1	251.23712600002182	26501745.95 Bps	1.52%
TEST 3.2	13.619169000012334	3633764.91 Bps	0.07%
TEST 3.3	64.67442299998947	90.11 Bps	0.71%
TEST 3.4	109.2468120000267	133.26 Bps	2.00%
TEST 4	4536.8618379999825	6.19 Bps	7%

Test case 4 was quite fast as compared to both VPS 1 and 2 and bandwidth taken for test case 3.1 and 3.3 is also less as compared to other VPS. Also CPU usage is reasonable.

CPU usage chart from Digital Ocean we can see that VPS 1 spkies up and goes up while the curve for VPS3 is less and VPS 2 has more steeper curve as we use.

3) Microsoft Azure

- **Non Containerized**

8. MUMBAI REGION

# test_cases	Response Time in ms	Bandwidth in Bps	Cpu usage
TEST 1	1395.5725230000098	-	1.41%
TEST 2	110 output	-	0.53%
TEST 3.1	117.61472399905324	23779536.02	1.56%
TEST 3.2	6.681744998320937	7765837.93	0.14%
TEST 3.3	54.15848300047219	94.80	1.01%
TEST 3.4	36.64969109987027	497.76	1.45%
TEST 4	4745.453149998561	5.91	2.76%

response time is less in some cases as compared to AWS Mumbai region, but bandwidth is same to that of AWS Mumbai region also CPU usage.

9. USA REGION

# test_cases	Response Time in ms	Bandwidth in Bps	Cpu usage
TEST 1	2374.346026999876	-	1.5%
TEST 2	110 output	-	0.52%
TEST 3.1	104.7517699999734 8	29809223.22	0.19%
TEST 3.2	7.87338799983263	5533945.45	0.145 %
TEST 3.3	42.78537800000049	109.76	1.59%
TEST 3.4	32.72366599994711	476.39	2.2%
TEST 4	4896.73550900002	5.73	2.78%

the bandwidth taken is quite lot in USA region and also the response time is more to that of Mumbai region.

- **Containerized**

10. MUMBAI SOUTH REGION

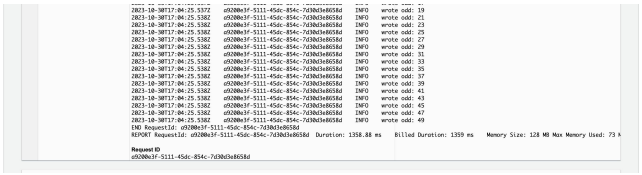
# test_cases	Response Time in ms	Bandwidth in Bps	Cpu usage
TEST 1	1975.7346719999332	-	1.46%
TEST 2	110 output	-	0.58%
TEST 3.1	78.64837299997453	46771532.73	1.55%
TEST 3.2	5.988082000054419	6698512.20	0.17%
TEST 3.3	30.673718999954872	189.56	1.04%
TEST 3.4	76.51984500000253	189.82	1.5%
TEST 4	4910.847454999923	5.71	3.00%

11. US-EAST REGION

# test_cases	Response Time in ms	Bandwidth in Bps	Cpu usage
TEST 1	3494.7462229998782	-	1.47%
TEST 2	110 output	-	0.59%
TEST 3.1	131.03656599996611	26805758.91	1.59%
TEST 3.2	9.26289700018242	5226066.90	0.18%
TEST 3.3	57.64360099984333	111.31	1.03%
TEST 3.4	90.31354200001806	163.47	1.49%
TEST 4	4863.076615999918	5.77	3.00%

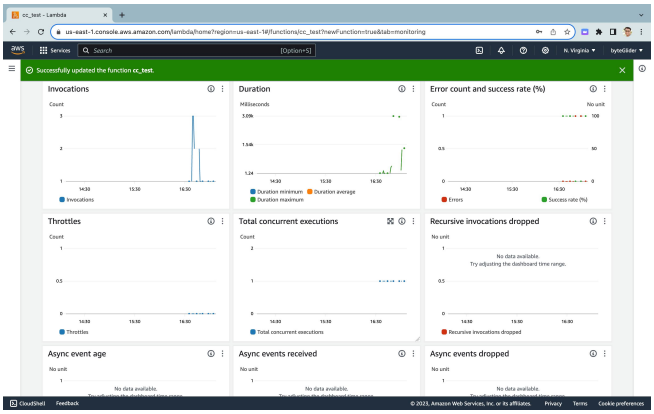
time taken for both the contanierzied ones is less as compared to normal ones and also the bandwidth is more or less same but in some cases the CPU usage is more.

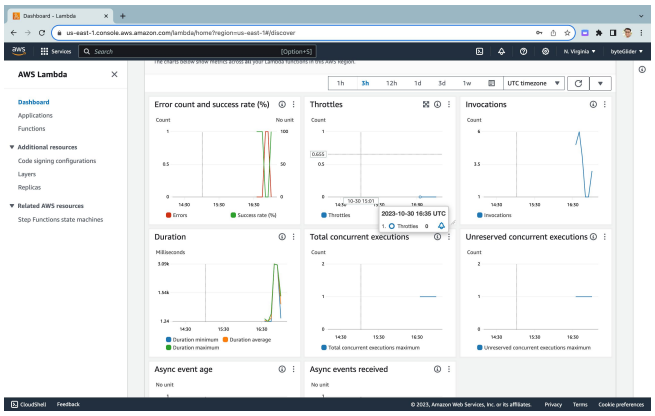
4) AWS LAMBDA: running a sample test case on lambda



AWS lambda serverless test.
monotiring logs and metrics:

lambda dasboard for viewing all the CPU and runtime metrics





we can also see the bandwidth and memory usage section.

REPORT REQUESTID: A9200E3F-5111-45DC-854C-7D30D3E8658D DURATION: 1358.88 MS BILLED DURATION: 1359 MS MEMORY SIZE: 128 MB MAX MEMORY USED: 73 MB INIT DURATION: 179.69 MS

One advantage of Lambda is that it charges only for the runtime it takes, so in this case it would only charge for 1359 ms, which is lot feasible for scalablity.
Azure has similar tool called functions.

CONCLUSION AND FUTURE SCOPE

So we can see that AWS when used with mumbai zone is moe fast in response time and most importantly it reduces 2 the badnwidth also while dockerizing the app in AWS it reduced the bandwidth quite a lot as compared to normal run. Secondly with Azure also we can observer a significant difference in response times due to close regions and also low bandwidth.
Thirdly using digital ocean is very easy as compared to both azure and aws as it is based on droplet concept so creating them and managing is lot easier for developer, so that they can focus more on development.
If we see for scalablity then AWS has lot of in house solutions as it provides lot of other services as well like Cloud Metrics, Load Balaning, Route 53 and security,also creating instance of ec2 is lot faster than azure basic vps instance creation as it is comparatively slower.
By our research, we can say that azure is bit costly as compared to aws and digital ocean and its billing cycle is little complex and more constrained.
So if you are a student who is working on project and needs cloud with less budget with more config and other options then go with Digital Ocean.

If you are a business who are willing to pay for high end services and run enterprise level apps then you can go for Azure and if you look for a sustanable solution which is scalalbe and provides all features at one place then AWS should would have you all covered.

Our future scope is that we would extend the testcases to maybe 10-15 which test other factors also like storage, scalablity and tests to be more of an app so to get full metrics with it.

REFERENCES

[1] Cloud service performance evaluation: status, challenges, and opportunities – a survey from the system modeling perspective, Qiang Duan - Information Sciences & Technology Department, The Pennsylvania State University, Abington College, USA.

[2] Cloud Performance Modeling with Benchmark Evaluation of Elastic Scaling Strategies - Kai Hwang, Life Fellow, IEEE, Xiaoying Bai, Member, IEEE, Yue Shi, Muyang Li, Wen-Guang Chen, and Yongwei Wu, Member, IEEE.

[3] The Who, What, Why, and How of High Performance Computing in the Cloud

[4] Github link for codes:

[5] https://github.com/MihSawant/cc_sample_tests

[6] https://github.com/yashz05/cc-test-cases