Performance Analysis of Microservice based software in serverless environment with Docker

Aryan Attri , Dr Hoor Fatima, Sushant Jhingram

School of Engeenering and Technology,Sharda University GreaterNoida,India

School of Engeenering and Technology, Sharda University
Greater Noida, India

SchoolofEngeeneringandTechnology,ShardaUniversity
GreaterNoida,India

aryanattri11@gmail.com, hoor.iitd@gmail.com,sushantjhingran@gmail.com

Abstract— Microservice architecture has gained significant popularity due to its ability to decompose complex applications into smaller, loosely coupled services. Alongside this architectural approach, the emergence of serverless computing has introduced a new paradigm for deploying and managing applications without the need to provision and manage traditional servers. This abstract presents a comprehensive performance analysis of microservice-based software in a serverless environment with docker, exploring the benefits, challenges, and considerations associated with this combination.

The performance analysis focuses on evaluating key metrics such as response time, throughput, scalability, and cost-efficiency. By leveraging the serverless model, the study examines how the dynamic allocation of resources impacts the overall performance of microservices**.**

Keywords— *Docker, serverless, Cloud, Microservices, Software Development, Deploy, Containers, Lambda*

INTRODUCTION

In recent years, microservices architecture and serverless computing have emerged as powerful paradigms in the software development landscape. Microservices allow complex applications to be broken down into smaller, loosely coupled components, while serverless computing enables developers to focus on writing code without the need to manage the underlying infrastructure. The combination of these two approaches has led to the rise of microservice-based software in serverless environments, offering numerous benefits such as scalability, cost-efficiency, and agility.

Docker is a widely used open-source platform that enables developers to automate the deployment and scaling of applications within containers. With its ability to package an application and its dependencies into a standardized unit, Docker provides a consistent and portable environment for software development and deployment. And use of virtualization has increased dramatically for over last few years [1].

One of the key advantages of using Docker is its ability to eliminate the often complex and time-consuming process of setting up and configuring development environments. With Docker, developers can package their applications and all necessary dependencies into a container, which can then be easily shared and run on any system that has Docker installed.

However, as with any software architecture, performance remains a critical factor that significantly impacts the overall user experience and system efficiency. With the increasing adoption of microservices in serverless environments, there is a growing need to analyze and understand the performance characteristics of such systems. This necessitates an in-depth examination of the challenges, trade-offs, and best practices related to the performance analysis of microservice-based software in a serverless environment. Only difference between different serverless platforms is their own ecosystem as each platforms forces developers to use their ecosystem to use their platform. [2]

The performance analysis of microservice-based software in a serverless environment involves assessing various metrics, such as response time, throughput, scalability, and resource utilization. Understanding how these metrics are affected by factors like service composition, communication patterns, and workload characteristics is crucial to optimizing the performance of the system.

Serverless methodology makes the cloud-based computing much easier to use as provides simplified programming environment, hence attracts developers and organizations [3]. AWS Lambda [4] is a serverless computing service provided by Amazon Web Services (AWS). It allows you to run your code without the need to provision or manage servers. With AWS Lambda, you can focus on writing your application logic while AWS takes care of the underlying infrastructure.

Lambda functions are event-driven, meaning they are triggered by specific events such as changes to data in an Amazon S3 bucket, updates to a DynamoDB table, or an HTTP request through the API Gateway. Once an event occurs, Lambda automatically scales the necessary compute resources to handle the workload.

Microservices architecture offers numerous advantages in terms of agility, scalability, resilience, and DevOps practices. It allows organizations to build and evolve complex applications more efficiently by decomposing them into smaller, independent services. However, it is essential to address the challenges associated with inter-service communication, data consistency, and monitoring to fully harness the benefits of microservices. When implemented correctly, microservices can empower organizations to deliver high-quality, scalable, and resilient applications in today's rapidly evolving software landscape. Microservices architecture have overcome traditional monolythic architecture [5].

This research paper aims to analyze the performance of microservice-based software in a serverless environment. The increasing popularity of microservices and serverless computing has paved the way for the development of scalable and flexible applications. However, the performance implications of deploying microservices in a serverless environment remain largely unexplored. This paper investigates the impact of using microservices in a serverless architecture on various performance metrics such as response time, throughput, and scalability. Through a series of experiments and measurements, we aim to provide insights into the performance characteristics of microservice-based software in a serverless environment.

I. LITERATURE REVIEW

1) Docker: Docker is a platform for Software based organizations as it is an open source platform for software development, deployment and maintanance. Docker is used for speedy developing and deploying applications. Docker can run on any server and can use any environment to run, ship the code. Docker can be used to ship applications on the cloud. Docker can be compared with virtual machine as it works in kinda same way and makes the application lightweight.[6] Docker is a platform for developing, shipping, and running distributed applications. It allows developers to package their applications and dependencies into a portable container, which can then be run on any host that has the Docker runtime installed. This makes it easy to deploy and scale applications across different environments, from local development machines to cloud-based production systems.

Developers all around the world now using Docker a lot in their software development process. And with containerization and microservices, Use of Docker has increased significantly Which can be shown in figure 1.

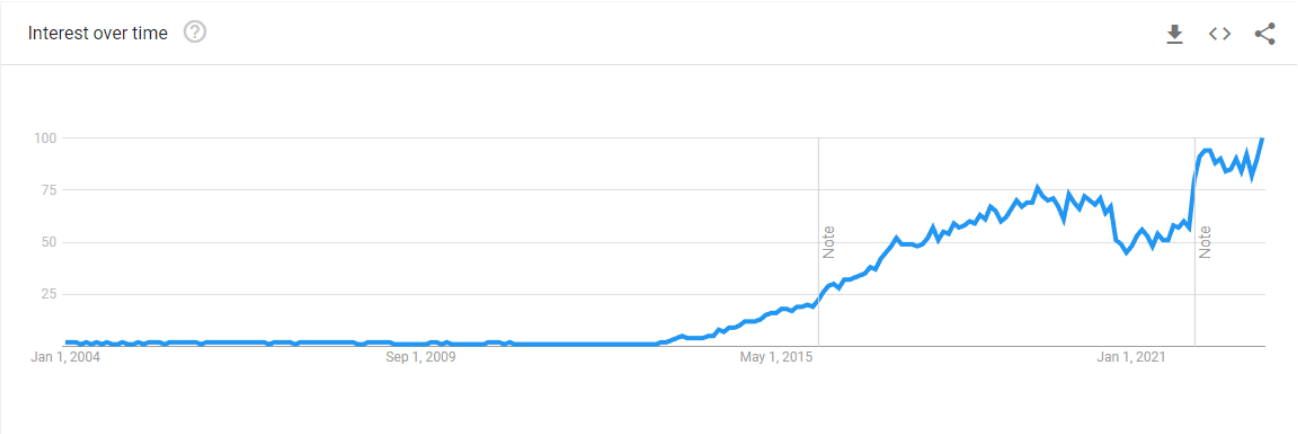

Fig (1): Use of docker acc. To google trends

When a micro business Plans to develop a software for their business, they try to include everything even if they don't need that specific service, which becomes hectic to manage and increases to cost of development and maintenance, Docker plays a big game in this case as this tool gives chance to develop different containers independently, which also allows to deploy and run applications in a container. The concept of using docker is based on the idea of updating with time and work, while minimizing the risk of having heavy maintenance cost, System update cost and time as well. Docker's performance evaluated by Amit M Potdar, Narayan D G, Shivaraj Kengond, Mohammed Moin Mulla in ScienceDirect Procedia Computer Science 171(2020) shows, how much of an impact does docker have in developing and running an application. [7] With time developers get the idea of specific field or part needs an improvement, and put resources at correct time and don't need to shut the whole server, picking up a specific container, editing and updating code base, deploying will work in this kind of situation. Docker also provides a platform for developers to create, publish and manage containers. Docker Hub, a cloud-based registry service makes it easy to distribute your containers and plugins.

2) Serverless:

Serverless refers to a cloud computing model that eliminates the need for developers to manage servers or infrastructure. In this paradigm, the cloud provider takes responsibility for provisioning, scaling, and maintaining the underlying resources required to run applications. Developers can focus solely on writing code for individual functions or microservices without the burden of server management.

By leveraging serverless architecture, developers can streamline their workflow and increase productivity. They no longer have to concern themselves with server provisioning, capacity planning, or server maintenance. Instead, they can concentrate on writing code and delivering functionality.

In a serverless environment, applications are built around functions that are triggered by specific events or requests. Each function executes independently, and the cloud provider dynamically manages the allocation of resources to handle the workload. Resources are scaled up or down automatically based on demand, ensuring optimal performance and cost efficiency.

The serverless model offers numerous benefits, including enhanced scalability, flexibility, and reduced operational overhead. It allows applications to scale seamlessly in response to varying workloads, ensuring efficient resource utilization. Additionally, developers can deploy code quickly and easily, as serverless platforms handle the deployment and operational aspects. Lambda model is much more scalable and elastic than many platforms, even container based services that autoscale[8].

3) Containers:

Containers is the key idea of docker to run and ship an application in a running stage, where docker creates an image of each container for easy shipment and management. The basic idea of server on hardware can be taken place by software-based servers in form of containers. As for Cloud based application, Containers as lightweight tech, which is a powerful technology to virtualize application for management of a application based on cloud. Containers have been a great success recently.[9] A container in Docker is a lightweight, standalone, executable package that includes everything needed to run a piece of software, including the code, a runtime, libraries, environment variables, and config files. Containers are isolated from one another and from the host system, which means that they can run consistently across different environments and that they do not interfere with other software running on the host system. Docker uses a technology called containerization to create and manage containers. This technology allows developers to package their applications and dependencies together in a single container, which can then be easily deployed and run on any system that has the Docker runtime installed.

Container can be viewed in these three parts:

- Builder: Tools to build containers like DockerFile.
- Engine: Applications like Docker Command and Dockerd Daemon which is used to run Docker.
- Orchestration: Technology to manage multiple containers like Kubernetes.

4) Microservices: Microservices have emerged as a popular architectural approach in software development, providing a scalable and flexible solution for building complex applications. In a microservices architecture, a large application is decomposed into a set of small, independent services that can be developed, deployed, and scaled independently. Each service focuses on a specific business capability and communicates with other services through well-defined APIs.

One of the key benefits of microservices is their ability to enable agility and faster development cycles. Since services are independent, development teams can work on different services simultaneously, using different technologies and programming languages that best suit the requirements of each service. This promotes faster iteration, deployment, and updates, allowing organizations to deliver new features and improvements more rapidly.

Scalability is another advantage offered by microservices. By breaking down an application into smaller services, each service can be scaled independently based on its specific needs. This enables efficient resource utilization and eliminates the need to scale the entire application when only a specific component is experiencing high demand. As a result, microservices can handle varying workloads effectively and ensure optimal performance under heavy traffic conditions.

The loose coupling between microservices also promotes resilience and fault isolation. If a single service fails or experiences issues, it does not affect the entire application. Other services can continue to function independently, ensuring that the overall system remains operational. Additionally, fault tolerance can be improved by implementing redundancy and fallback mechanisms within the architecture.

Microservices architecture also facilitates the adoption of modern DevOps practices. Each microservice can have its own deployment pipeline, enabling continuous integration, continuous delivery, and automated testing for individual services. This promotes faster and more reliable deployments, as changes to one service do not impact the entire application. Moreover, teams can adopt different technologies and frameworks for different services, allowing them to leverage the best tools available for each specific task. Most of big organizations in the start developed complete application later than adapted microservices[10].

However, the adoption of microservices comes with its own set of challenges. Communication and coordination between services must be carefully managed, as distributed systems introduce complexities related to inter-service communication, data consistency, and transaction management. Additionally, monitoring and troubleshooting can become more complex in a distributed environment, requiring robust logging, monitoring, and tracing mechanisms.

5) Products similar to Docker:

Linux containers proving nowadays to be one of the best platforms to manage software, even a complete operating system and just like Docker there are multiple toolsets to help in managing containers like LXC, Podman, CRI-O and others which can be chosen according to the need and requirement. Modernized factories are also becoming computerized leading them to become a smart factory, and shown by S. N. Srirama,M. Adhikari and S. Paul, in J.Netw. Comput. Appl., vol 160. [11] Docker is playing a major role in software management for factories for multiple software modules between multiple environments.

Using Docker for hybrid clouds for deploying complex infrastructures, Jozsef Kovacs, Peter Kascuk, Mark Em in Advances in Engineering Software (2018), shows using Occopus(a new cloud orchestrator) for independent deployment at user level[12]

II. METHODOLOGY

For research purpose, web software is developed in different environment, one is using docker uploaded on EC2 instance and for serverless environment, a lambda functions ware created having same code with microservices architecture to compare and get analysis of two different environments.

AWS EC2 instance are used with HTTP security group to run web services are created in one of the instance docker is installed and web application file to uploaded using docker file and creating an image and container for the image and in lambda functions, same HTML code was deployed and an API gateway was created, giving an access to same website view and experience.

For docker database aws instance is used and runs docker on aws single instance, EC2 instance to launch a virtual server for storage purpose of data.

With microservices, dividing a big code into multiple small code blocks creating containers of code blocks, we achieve reusability, improved reusability and speed. But Microservices application can also vary depending on environment. [13]

Website used for research paper is uploaded on github repositories and link is here : https://github.com/Aryan4g/Website-/raw/main/site.zip

Using EC2 instance with docker and lambda function and comparing the page insight dev test for both of the links gets the result which can be seen in table 1 derived from fig 3,4.

Table 1:- comparison of instance monitoring parameters when docker is used and when not used.

Parameter/instance	EC2 with Docker	Lambda Function
First contentful paint	**1.1s**	**3.4s**
Speed index	**2.6s**	**3.4s**
Largest Contentful Paint	**1.7s**	**3.4s**
Time to Interactive	**2.6s**	**--**
Total blocking Time	**30ms**	**0ms**
Cumulative Layout Shift	**0.451**	**0.001**

According to table 1 based on figure 3 and figure 4, Speed Index show slight delay when using lambda function but blocking time and layout shift shows lambda function's performance better. Docker as a complete package for software delivery is leading market today proves to be a better solution.[14]

III. RESULT

After comparing same application's deployment with two different cases, one in which docker is used on EC2 Instance and second with serverless lambda function and result is out with betterment and better knowledge about serverless environment and docker. In maintain and updating application, docker proves to be one of the best methodologies in software development industry. It will be easy to make changes in the image and update the container without breaching security too much. The deployment of application using docker image and AWS console is fast and easy, so that least amount of interaction was needed with the code inside the containers and with having better security. While Serverless proves to be better option as Applications built using a serverless architecture can automatically scale up or down based on demand, ensuring optimal performance and cost efficiency. This is achieved through the use of functions, which are small, stateless units of code that are triggered by events.

After deploying the same website on two different environments, one on EC2 Instance using docker and another with microservices architecture on serverless lambda function and parameters of both instances after having a traffic of minimum 50 live interactions within 30 minutes, data is recorded to get proper result for better comparison.

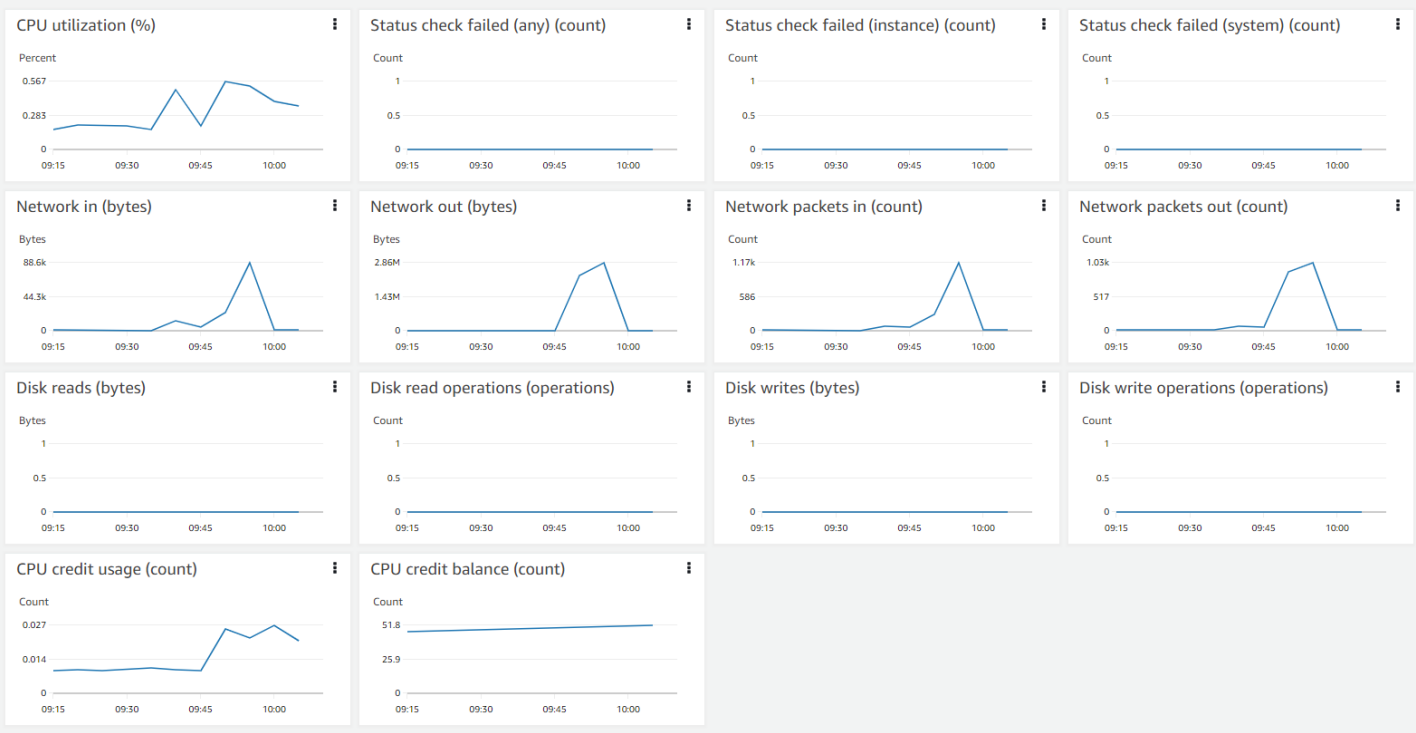

Fig 2:- Shows EC2 Instance monitoring parameter which used Docker

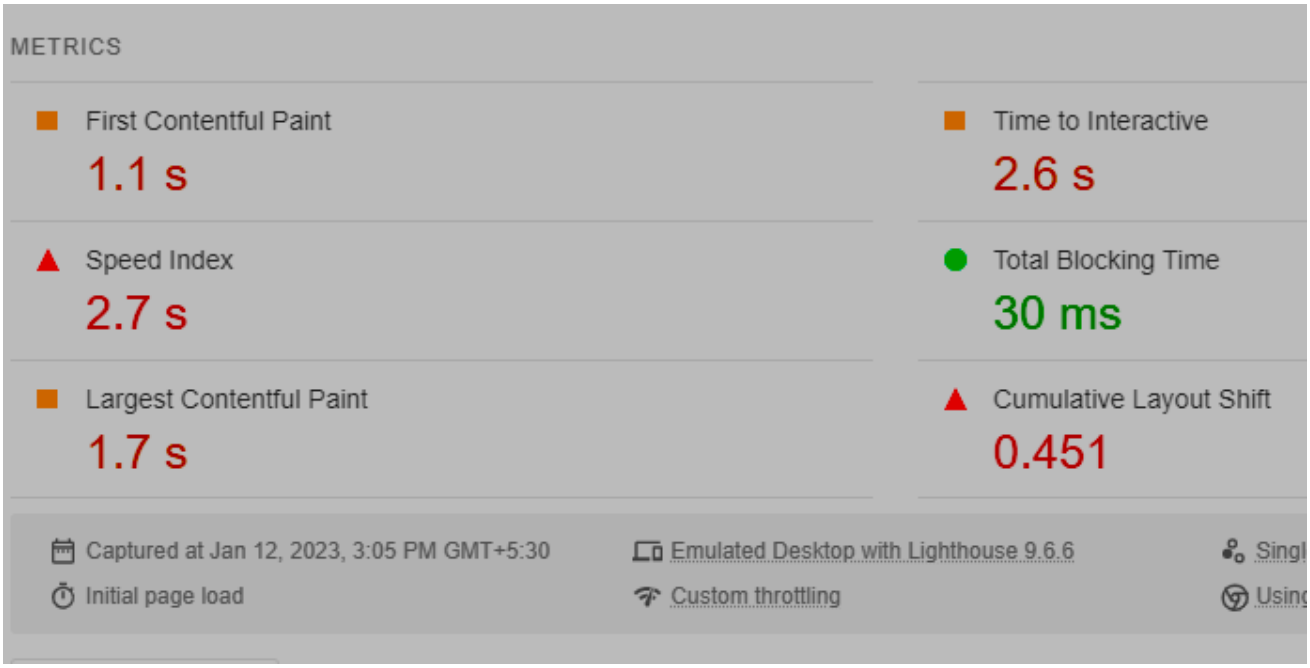

Fig 3:- Shows metrics perimeters provided by page insight using docker

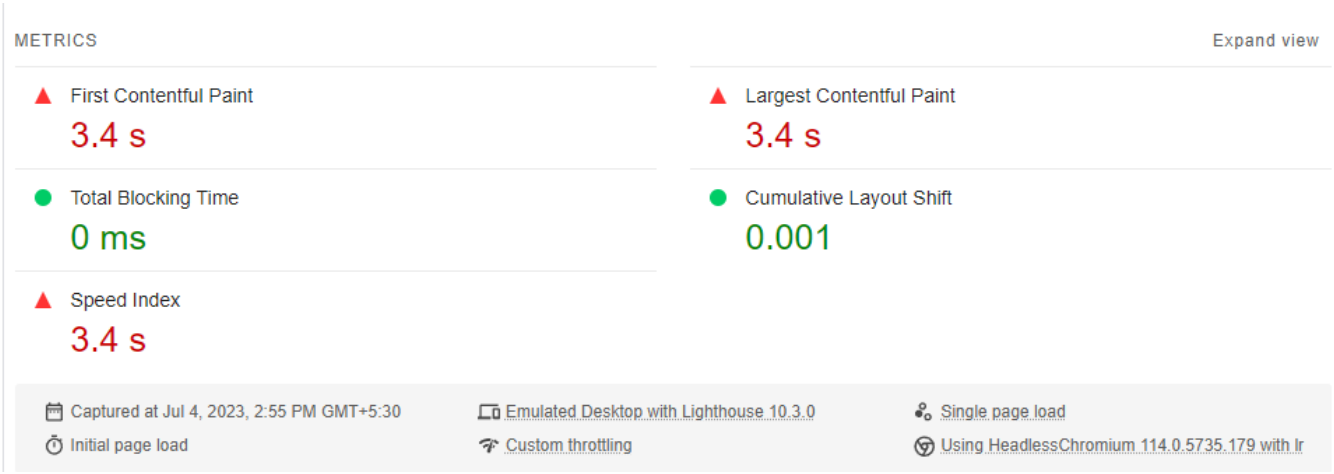

Fig 4:- Shows metric perimeters provided by page insight using lambda

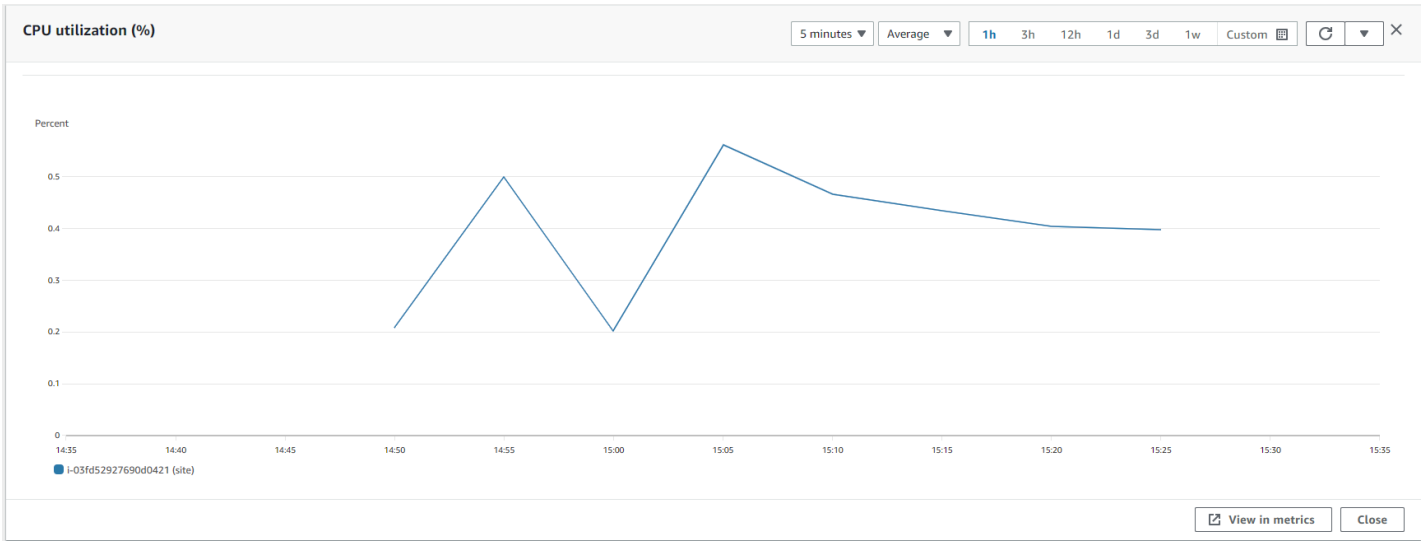

Fig 5:- Shows CPU utilization of docker application on EC2 instance

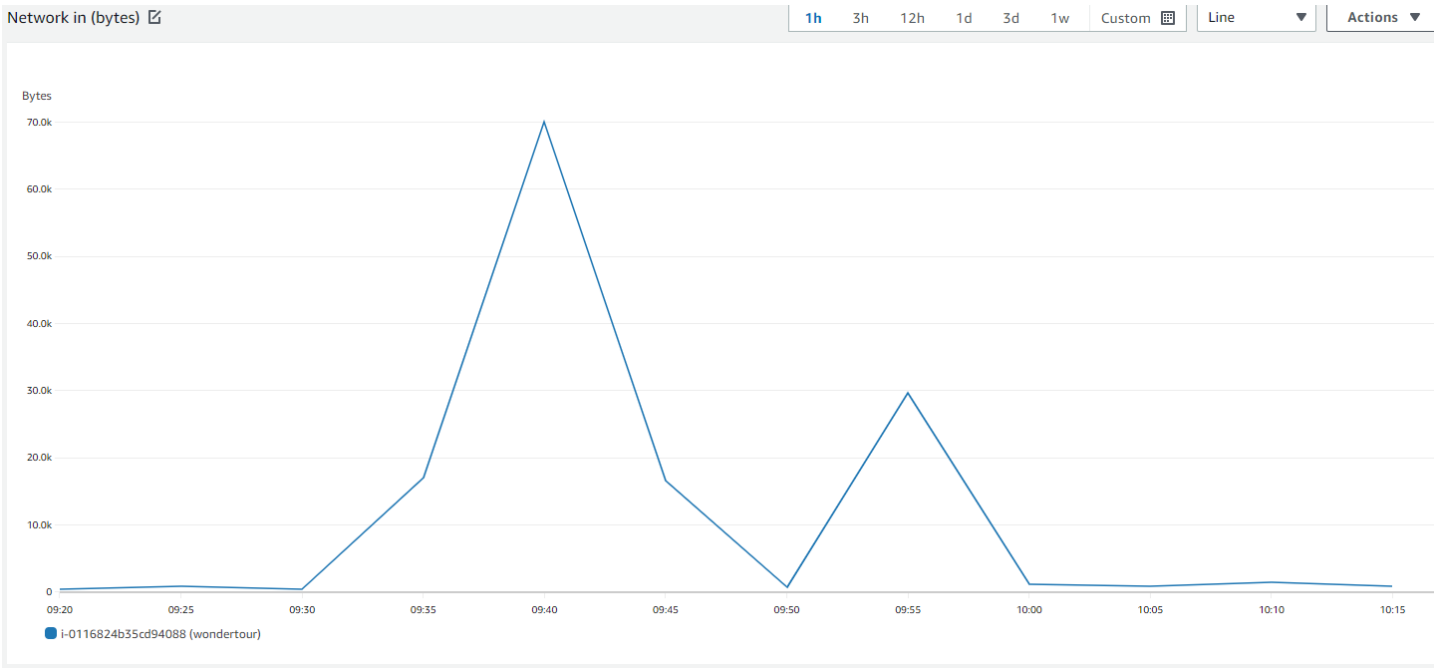

Fig 6:- Shows Network in when docker is used

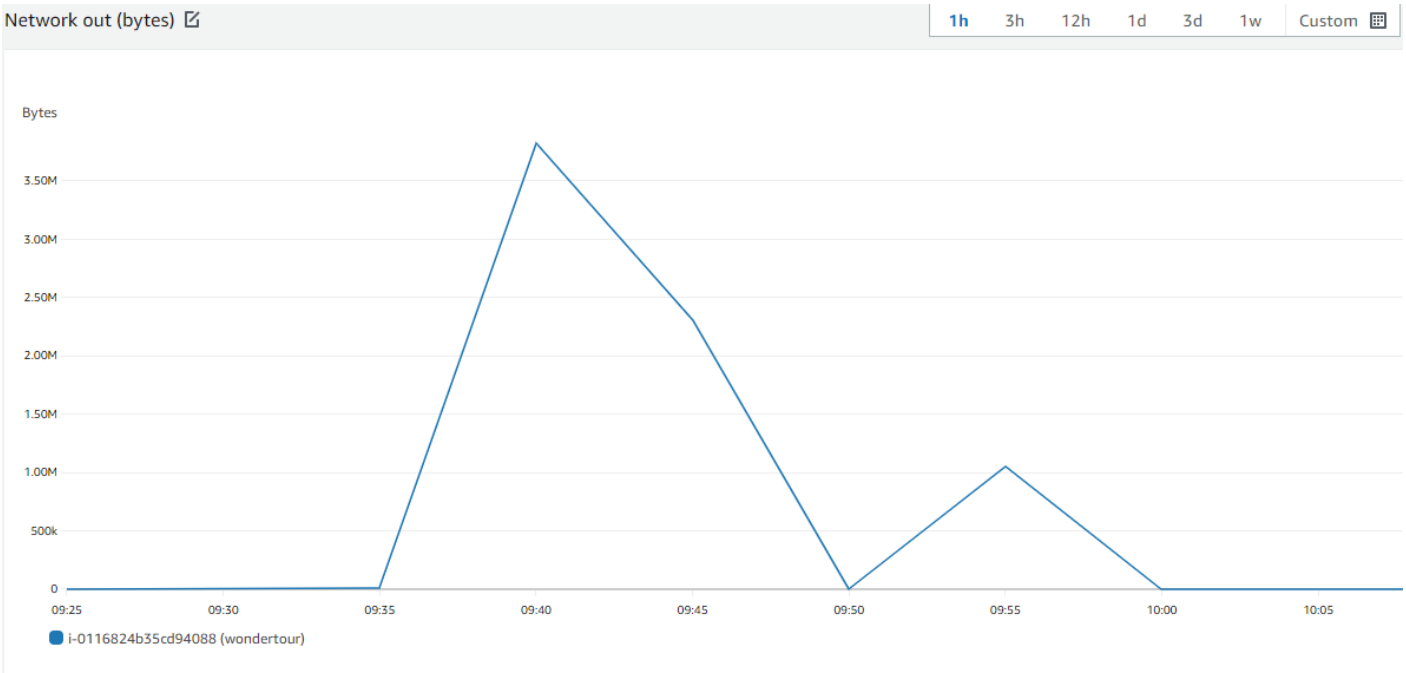

Fig 7:- Shows Network out when docker is used

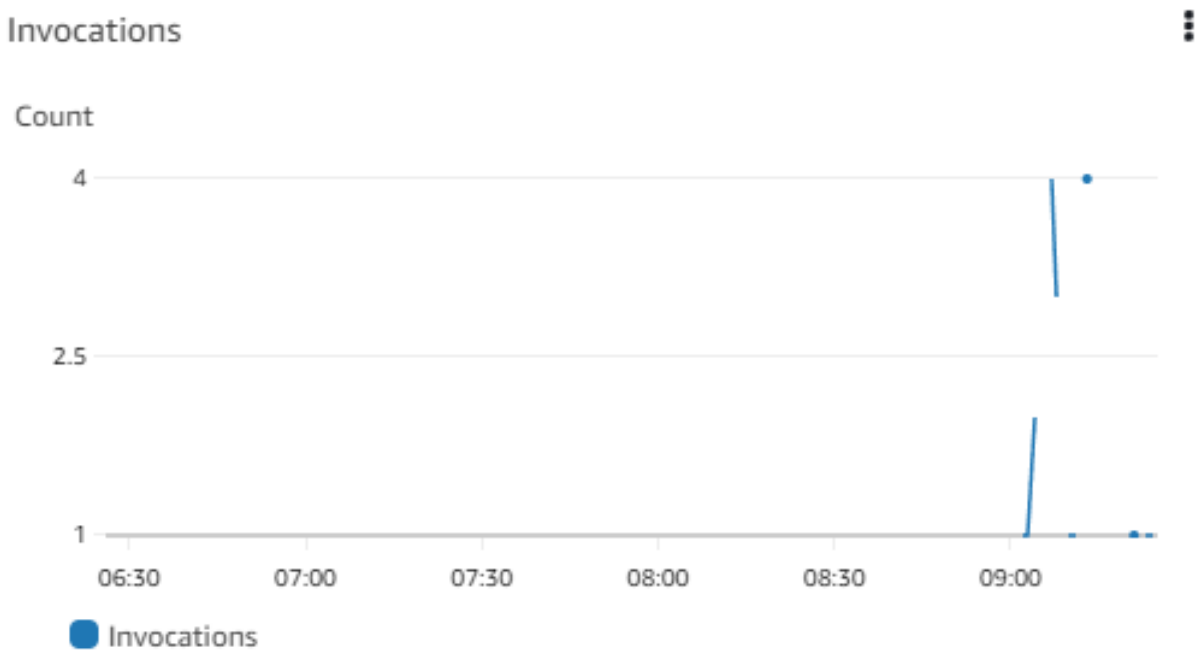

Fig 8:- Shows Incovations Count Of Lambda Function

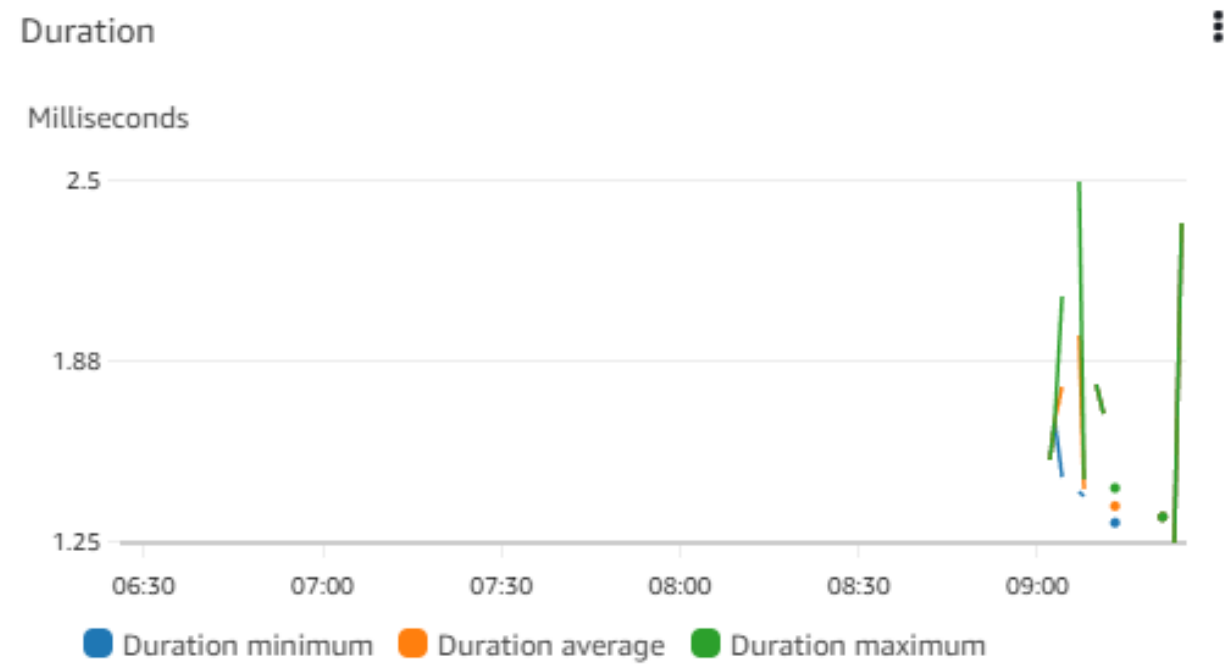

Fig 9:- Shows Duration to process Code

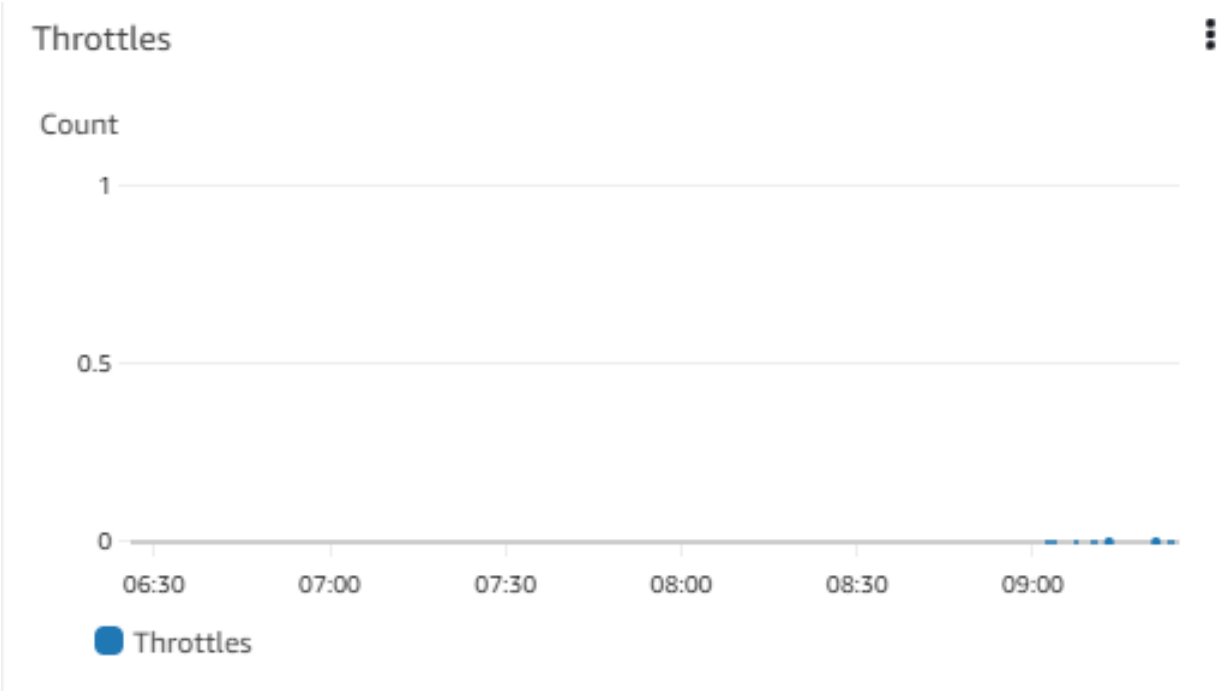

Fig 10:- Shows Throttles Count of lambda function

Fig 11:- Shows Total Concurrent Executives of lambda function

In this paper, the performance analysis of docker based EC2 instance was evatuated and we focused on performance as shown in figure 5, figure 6, figure 7 showing CPU Utilization, Network out and Network in of the instance showing CPU utilization of 0.56% and Max Network In and Out were 70k and 3.8M Respectively.

Figure 3 and Figure 4 depicts the measurements recorded to get performance metrics from page insight tool, showing page loading time, First contentful paint opening time, largest contentful paint opening times in both scenarios one in EC2 Instance and other in Serverless environment, where metrics shows docker's performance to be faster but serverless function's total blocking time and cumulative layout shift to be lot better.

Figure 8,9,10 and 11 shows cloudwatch metrics of serverless lambda home function having max invocations of 4 counts takes average of 1.5 miliseconds to process code and maximum of 2.5 miliseconds to process code while having zero throttles having average of 1 concurrent execution.

During test multiple images were created and multiple containers were formed and best part is many containers can be placed in a single host, in this research three different containers names 'website', 'site' and 'wondertour' were places in a single instance and it was easy to handle this situation.

While multiple lambda functions were created to create same website pages, each with different API Gateways to connect and create a link to open the web view of website.

Using microservices in serverless environment already proves to be cost efficient, more scalability and better experience for developer as there is no need to manage server where developer can focus on writing code and functionality.

IV. CONCLUSION

Overall, the Literature suggests that microservice applications in serverless environment with docker can bring significant benefits to organizations, it is important to carefully consider the potential challenges and to have a well- designed strategy in place to address them. Using AWS EC2 instance for hosting which is fast and easy to manage proves to be a good decision. EC2 instances can be created easily having manageable security settings, and security options. Docker with AWS is a power packed method for development, deployment and managing application. It gives developers an environment of easy shipment of

application and access to portability of code with image and containers as well that's why docker is so hyped now a days. And AWS serverless lambda functions provide free hand to developers to focus on functionality and writing code without worrying about managing servers as serverless proves to be cost efficient, provide scalability and efficient.

In this paper, the comparision of docker with other shipping and deployment methods and serverless architecture is performed.

In future, for further research, same application will be developed with microservices environment and Containers having code to deploy in lambda functions will be performed.

REFERENCES

[1] Bui, T., 2015. Analysis of docker security. *arXiv preprint arXiv:1501.02967*.

[2] Baldini, I., Castro, P., Chang, K., Cheng, P., Fink, S., Ishakian, V., Mitchell, N., Muthusamy, V., Rabbah, R., Slominski, A. and Suter, P., 2017. Serverless computing: Current trends and open problems. *Research advances in cloud computing*, pp.1-20.

[3] Jonas, E., Schleier-Smith, J., Sreekanti, V., Tsai, C.C., Khandelwal, A., Pu, Q., Shankar, V., Carreira, J., Krauth, K., Yadwadkar, N. and Gonzalez, J.E., 2019. Cloud programming simplified: A berkeley view on serverless computing. *arXiv preprint arXiv:1902.03383*.

[4] Aws lambda. URL https://aws.amazon.com/lambda/. Online; accessed 1 july 2023.

[5] Li, S., Zhang, H., Jia, Z., Zhong, C., Zhang, C., Shan, Z., Shen, J. and Babar, M.A., 2021. Understanding and addressing quality attributes of microservices architecture: A Systematic literature review. *Information and software technology*, *131*, p.106449.

[6] Srirama, S.N., Adhikari, M. and Paul, S., 2020. Application deployment using containers with auto-scaling for microservices in cloud environment. Journal of Network and Computer Applications, 160, p.102629.

[7] Potdar, A.M., Narayan, D.G., Kengond, S. and Mulla, M.M., 2020. Performance evaluation of docker container and virtual machine. *Procedia Computer Science*, *171*, pp.1419-1428.

[8] Hendrickson, S., Sturdevant, S., Harter, T., Venkataramani, V., Arpaci-Dusseau, A.C. and Arpaci-Dusseau, R.H., 2016. Serverless computation with {OpenLambda}. In *8th USENIX Workshop on Hot Topics in Cloud Computing (HotCloud 16)*.

[9] Pahl, C., Brogi, A., Soldani, J. and Jamshidi, P., 2017. Cloud container technologies: a state-of-the-art review. *IEEE Transactions on Cloud Computing*, *7*(3), pp.677-692.

[10] Thönes, J., 2015. Microservices. *IEEE software*, *32*(1), pp.116-116.

[11] Senington, R., Pataki, B. and Wang, X.V., 2018. Using docker for factory system software management: Experience report. *procedia CIRp*, *72*, pp.659-664.

[12] Kovács, J., Kacsuk, P. and Emődi, M., 2018. Deploying docker swarm cluster on hybrid clouds using occopus. *Advances in Engineering Software*, *125*, pp.136-145.

[13] Jhingran, S. and Rakesh, N., 2021, July. Performance factor impacting behavior of microservices in various hosting domains. In *2021 Fourth International Conference on Computational Intelligence and Communication Technologies (CCICT)* (pp. 160-164).

[14] IEEE. Combe, T., Martin, A. and Di Pietro, R., 2016. To docker or not to docker: A security perspective. *IEEE Cloud Computing*, *3*(5), pp.54-62.