National College of Ireland

ANALYSIS OF DYNAMIC APPLICATION LOAD BALANCING IN KUBERNETES USING CDN

Research Project
MSc Cloud Computing

Nitu Kumari
Student ID: x21215995

School of Computing
National College of Ireland

Supervisor: Diego Lugones

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Nitu Kumari
Student ID:	x21215995
Programme:	MSc Cloud Computing
Year:	2023
Module:	Research Project
Supervisor:	Diego Lugones
Submission Due Date:	14/8/2023
Project Title:	ANALYSIS OF DYNAMIC APPLICATION LOAD BALANCING IN KUBERNETES USING CDN
Word Count:	5624
Page Count:	22

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	Nitu Kumari
Date:	17th September 2023

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	☐
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	☐
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	☐

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

ANALYSIS OF DYNAMIC APPLICATION LOAD BALANCING IN KUBERNETES USING CDN

Nitu Kumari
x21215995

Abstract

Rapid growth in web traffic has a negative impact on timely data transmission, making it a significant issue. Web servers are becoming increasingly overburdened due to a rise in the number of users and the quantity and size of content, as a result of immense data consumption. Web service providers and enterprises dependent on the Internet are impacted. Content delivery networks are a convenient way to reduce server and network traffic and increase end-user response times. By selecting the server that is geographically closest to the user, a Content Delivery Network (CDN) accelerates hardware delivery, resulting in faster load times and a more streamlined browsing or streaming experience. Concurrently, Kubernetes has acquired popularity as a container orchestration platform that is used to manage and extend applications. It effectively distributes incoming traffic across multiple groups of independent servers hosting applications. Experts have investigated the use of CDN and Kubernetes together, has improved load balancing and the user experience. Kubernetes, load balancing techniques such as weighted round-robin and IP hash and Content Delivery Networks can be used to maximize the benefits of both approaches. CDNs reduce latency and enhance response times by bringing content closer to users. Kubernetes prevents servers from becoming overloaded during this process by dynamically distributing requests across pods, thereby assuring efficient resource utilization. The integration of these technologies enhances efficiency and the overall user experience. The proposed combination of Content Distribution Networks, Kubernetes, and efficient load balancing algorithms is intended to increase resource allocation, load balancing capabilities, and user content distribution. The purpose of this research is to investigate the complexities of this well-designed process to develop a system that is straightforward to use.

Keywords - User experience, Content Delivery Networks (CDNs), Load balancing, Kubernetes, Resource allocation, Response times, Latency, IP Hash,Weighted round-robin algorithm.

1 Introduction

Cloud computing has drastically revolutionized business tasks since it makes it simple to have instant access to computing resources Shafiq et al. (2021). Two significant technologies, such as cloud computing and containerization, are highlighted in the research background material for their revolutionary effects. Users and organizations can now access and utilize an extensive range of services based on their specific requirements because of the robust and flexible architecture of cloud computing. In order to successfully

scale the infrastructure for such massive loads, it is a usual practice in modern computing to deploy more servers Nagarajan (2019). But in the context of cloud computing, load balancing—the evenly distributed allocation of workload over numerous servers—is a typical approach. Due to more individuals using the Internet and other online services, web traffic has significantly increased recently.

Load balancing has developed into a critical technique for maximizing resource allocation and distributing incoming requests among multiple servers in order to handle increasing internet traffic and ensure a smooth user experience Afzal and Kavitha (2019). To maximize resource utilization and prevent server congestion, effective workload management across multiple servers requires the implementation of load balancing solutions. Load balancing solutions play a crucial role in ensuring uninterrupted operations, optimizing resource efficiency, and delivering superior services in a cloud environment. These solutions not only improve the scalability, dependability, and fault-tolerance of a web application's architecture overall, but also accelerate websites by distributing incoming requests effectively Radhika and Duraipandian (2021). Due to difficulties with load balancing and preserving optimal performance for high-traffic websites, a Content Delivery Network (CDN) was developed as a potential remedy. The CDN consists of a large number of servers that have been intelligently distributed around various global locations. This method greatly reduces network traffic, lowers latency, and improves the efficiency of information distribution as a whole. Websites can improve user experience and address load balancing issues in modern high-traffic web settings by using a CDN to give users faster and more dependable access to content.

1.1 Content Delivery Networks (CDNs): Enhancing Web Performance and User Experience

Numerous "digital lifestyle" populations are increasing their daily content consumption demands Yang et al. (2023). Effective material delivery has become essential for websites with a lot of traffic due to the Internet's rapid development and the rising desire for a good user experience. Content delivery networks (CDNs) are currently seen as a potential solution to handle multiple concurrent user requests and ensure fast and accurate content delivery. By providing content from the server that is most convenient for the user's location, a CDN delivers a strategically fast-positioned network of servers that reduces network congestion and delays. The ability of CDNs to distribute content to several servers globally for faster delivery and more scalability is the reason for their widespread appeal. The suggested routing method automatically allows users to directly get access to the the content depending on their membership by exploiting the CDN's capabilities, maximizing the user experience and resource allocation Varma et al. (2023).

To reduce latency and speed up the delivery of the requested content, the CDN directs the user's request to the closest edge server. Users receive content using this neighborhood-based content delivery function from a server that is close to them, resulting in an easy and smooth user experience. The scalability and durability of online services are greatly improved by CDNs, in addition to accelerating response times. The application can handle heavy traffic loads without performance deterioration during times of high demand thanks to the ability of workloads to be dynamically distributed among numerous edge servers via CDN. In order to maintain duplicate copies of the same piece of content on different servers, CDNs also take advantage of content replication. This redundancy ensures that even if one server goes down, material may still be accessed

from other servers, improving the overall stability of the online program. Users' streaming experiences are optimized for them via Content Delivery Networks (CDNs) using the adaptive bitrate streaming approach, especially when watching multimedia content like videos. In general, CDNs have become a vital part of the internet's infrastructure, enabling businesses to effectively transmit material, improve user experience, and maintain high performance even in the face of tremendous amounts of traffic. The user experience, video streaming quality, and application performance of web apps that use CDNs are all enhanced for the user.

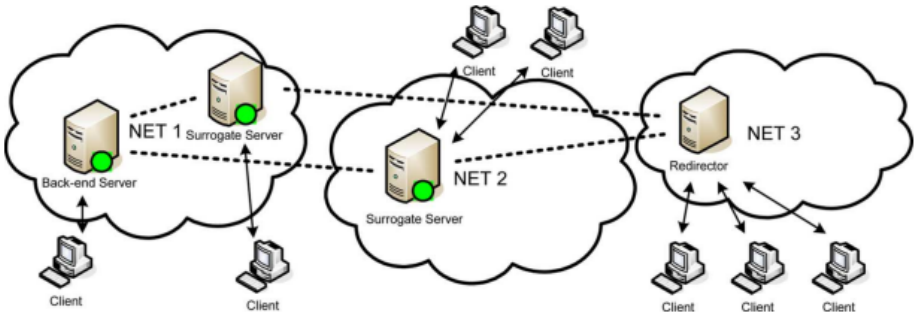

Figure 1: Content Delivery Network

1.2 Load Balancing in Cloud Environments: Challenges and Solutions

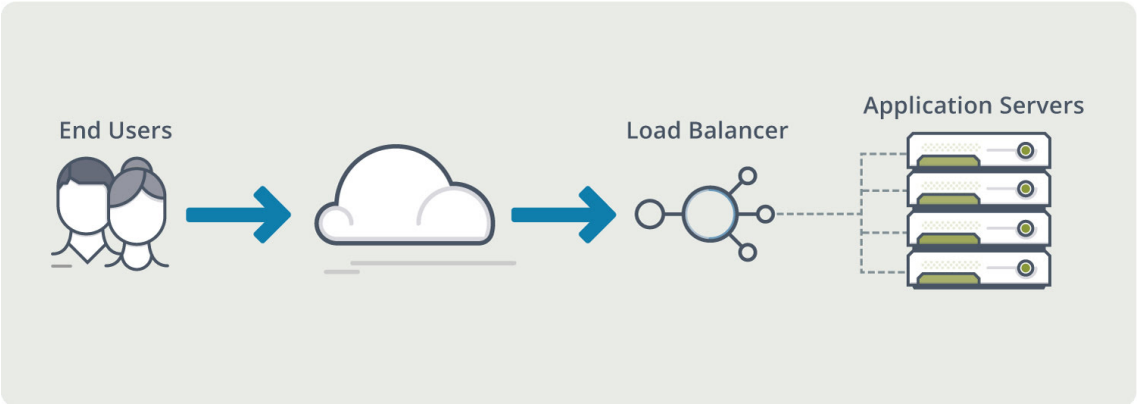

Figure 2: Importance Of Load Balancing

Figure 2 demonstrates that load balancing is required for distributed environments to function effectively. As the cloud computing is growing rapidly, the increasing customer demand for more services, and the ensuing integration, cloud load balancing has become a fascinating and crucial area of research Darji and Nakrani (2021). It intends to use a scalable network of nodes to transparently transmit, evaluate, and provide services. Primarily, it distributes resources and stores information in an open environment. Consequently, data storage is rapidly expanding. Website traffic has significantly increased as a result of the enormous increase in Internet usage and the widespread adoption of web services. Modern and busy websites must concurrently manage a large number of user

or customer requests, running in the hundreds or even millions, in order to offer correct text, photos, videos, or data.

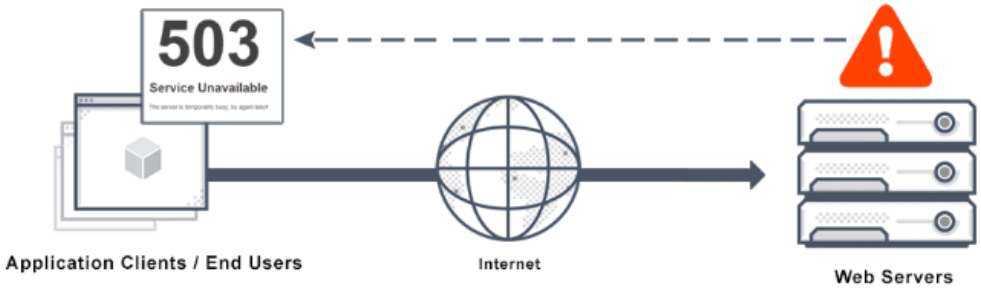

Figure 3: Server Overloaded Problem

Modern IT best practices frequently involve adding additional servers to the infrastructure in order to efficiently manage these enormous volumes. Content Delivery Networks (CDNs) have emerged as a promising solution by enhancing cloud load balancing. A CDN consists of geographically dispersed and strategically positioned servers. This method increases information delivery efficiency while decreasing latency and network congestion. By outsourcing content delivery from origin servers to peripheral servers, content delivery networks (CDNs) improve response times, video streaming, and application performance. This research investigation aims to manage the dynamic load of Kubernetes by demonstrating the effectiveness of Content Delivery networks.

The goal is to optimize content delivery to consumers, improve resource allocation, and enhance load balancing by combining the global reach and efficient load balancing capabilities of a content delivery network (CDN). In order to improve online speed and user experience in cloud environments, we intend to execute a comprehensive performance evaluation and analysis of the effectiveness of using CDNs for load balancing in Kubernetes installations.

1.3 Research Question

What are the potential advantages of combining Content Delivery Networks (CDNs) with Kubernetes for load balancing in web applications, and how can this combination enhance web performance and user experience?

1.4 Research Objective

This research seeks to investigate and develop a novel method for dynamic load balancing for web applications running on Kubernetes clusters. The purpose of the research is to incorporate Content Delivery Network (CDN) technology into load balancing in order to improve web application user performance and response times.

2 Related Work

This section provides a comprehensive analysis of the literature on "Dynamic Load Balancing Using CDN in Kubernetes Environments." Insights into various load balancing techniques and approaches in the context of Content Delivery Networks (CDNs) and Kubernetes settings are the goal of this research. By carefully examining earlier research and studies, this is accomplished. The major objective of this literature review is to discuss about the present state of load balancing strategies, including their benefits, drawbacks, and difficulties. Additionally, it seeks to provide new avenues for investigation and advancement in this field by incorporating knowledge from pertinent academic studies.

2.1 Content Delivery Networks (CDNs) and Load Balancing

The paper by Punit Gupta Gupta (n.d.) proposes an error-sensitive load balancing method for Content Delivery Networks (CDNs) to improve scalability and dependability. The study investigates the increase in network traffic and demand for content services provided by the CDN. Utilizing load balancing, request routing, and resource replication improves CDN performance. It presents a generalized strategy for load balancing to overcome existing limitations and proposes a fault-sensitive method to improve server reliability. The article by S.P. Sitorus, E.R. Hasibuan, and R. Rohani investigates Sitorus et al. (2021) load balancing strategies and their impact on Content Delivery Networks (CDNs). Simulations are used to illustrate how various load balancing strategies improve server service performance by reducing latency and packet loss. The study acknowledges certain limitations, such as the fact that the simulated environment does not replicate real-world conditions precisely. To precisely evaluate the effectiveness, additional comparisons with other algorithms are necessary. The research of Nakanishi et al. (2020) focuses on optimizing content delivery in the cloud by analyzing enhanced video content quality and increased traffic. The study proposes an effective method for content delivery that accounts for network distance and dynamically configures the content cache on the edge network using the BGP AS path. This Kubernetes-implemented strategy improves client access while diminishing HTTP response time. Additional Internet testing is slated for real-world applications. The paper Simić et al. (2023) compares collaborative advertising with non-cooperative online document retrieval using a commercial CDN to demonstrate the efficacy of content delivery in a CDN. Similar collaborative push performance while update and replication traffic is reduced. According to this paper, granular replication issues can be resolved by implementing cluster replication based on client proximity. Offline and online incremental clustering methods are provided to accommodate user access preferences. To encourage their use in CDN scenarios, the benefits of incremental clustering, cluster-based replication, and online popularity prediction systems are highlighted.

2.2 Solutions offered by Kubernetes for Dynamic Load Balancing

Zhang et al. (2018) Zhang et al. (2018) proposed a dynamic load balancing algorithm to the Kubernetes Ingress Controller in order to resolve the deficiency of the integrated load allocation scheme. Before allocating resources, this algorithm evaluates the CPU, storage, and network bandwidth of each node, in addition to the importance and weight

of incoming requests. It also takes the interdependencies between microservices into account to optimize delivery. Qingyang Liu et al. (2020) used the same custom load balancer Liu et al. (2020)to take into account the request weight and the active state of each node. In terms of requests per second (RPS), these techniques reduce prospective expenses and enhance efficiency. Due to its container support and dynamic resource management, Kubernetes (K8s) is regarded indispensable for peripheral computing Nguyen et al. (n.d.) by the authors of Q.-M. , T. Kim, L.-A. Phan, T. Nguyen. They address issues with load balancing that impede query processing in geographically distributed edge scenarios. They propose the sophisticated traffic balancer Resource Adaptive Proxy (RAP) as a solution. RAP continuously evaluates nodes and worker pools to improve load balancing, prioritizing local request processing for lower latency and greater throughput in environments with a high demand for computation. RAP has potential as a load-balancing option for Kubernetes in peripheral computing environments, according to the findings of this study. The paper by T. T. Nguyen, Y. J. Yeom, T. Kim, D. H. Park, and S. Kim provides information about the Horizontal Pod Autoscaler (HPA) engine as well as Kubernetes' capabilities and performance Nguyen et al. (2020). This study investigates the impact of the difference between Prometheus Custom Metrics (PCM) and Kubernetes Resource Metrics (KRM) on the behavior of HPAs. Research improves Kubernetes load balancing by casting light on HPA performance and empowering users to make knowledgeable decisions regarding load balancing strategies based on multiple metrics. Using available assets. In the paper by C. C. Chang, S. R. Yang, E. H. Yeh, P. Lin, and J. Y. Jeng, a standardized method for dynamic cloud resource provisioning with Kubernetes deployment is outlined Chang et al. (2020). Taking system resource utilization and application quality of service (QoS) metrics into account, it addresses the drawbacks of the current Kubernetes approach. The process of platform resource provisioning is modularly constructed, making it easy to implement and replace algorithms without affecting other modules. By using service-specific routing policies, Service Mesh Istio was employed in the paper Shitole (2023) to dynamically distribute traffic among services.

2.3 Edge Computing Strategies for Load Balancing

X. Wei and Y. Wang describe load balancing techniques to efficiently distribute data across clouds, increase data processing efficiency and reduce data access latency in their paper Wei and Wang (2023). They suggest load balancing solutions, such as offloading and data replication, to reduce the need for storage on overburdened peripheral servers. This paper proposes a data sorting technique for periphery computing with data ubiquitylation and a significant reduction in data access latency. Authors G. Xu, M. Zhang propose a JCETD (Hybrid Cloud Edge Task Deployment) load balancing method for hybrid cloud edge data centers Dong et al. (2021). For rapid task deployment and global load balancing, it employs reinforcement learning and pruning mechanisms. By comparing the proposed solution to the current standard of care, both completion time and reaction time are significantly reduced. Future work on conventional cloud computing will concentrate on intelligent task placement and other advances in load balancing. The paper Chen et al. (2020) by authors Chen and Zheng proposes a load balancing mechanism referred to as "strategy w" to organize and distribute data in heterogeneous edge computing environments. To ensure effective load balancing and resource utilization, the primary objective is to fairly distribute data across edge nodes with varying

capacities. The method employs weighted Voronoi diagrams and weighted entropy for load balancing. Taking into consideration heterogeneity and cooperation between edge nodes, strategy w is a useful component for edge computing, according to the paper.

2.4 Summary of Literature Review

Reference	Framework	Approach	Advantages	Limitations
Author -Punit Gupta	Error-sensitive load balancing method for Content Delivery Networks	Request routing, and resource replication	Enhanced Dependability ,CDN's server reliability is improved	Doesn't provide extensive performance evaluations
Author -Kento Nakanishi, Fumiya Suzuki	Optimal content delivery method	BGP AS path network distance	Reduced Response Time	Performance issue in large-scale scenarios
Author - Zhang, J., Ren, R	Dynamic load balancing algorithm for Kubernetes Ingress Controller	CPU, storage, and network bandwidth of each node before allocating resources	Prioritized Request Handling	Less robust
Author -Q.-M. Nguyen, L.-A. Phan, and T. Kim	Resource Adaptive Proxy (RAP)	Prioritize local request processing	High Throughput	Performance issue in huge traffic
Author - Y. Dong, G. Xu, M. Zhang	Hybrid Cloud Edge Task Deployment (JCETD)	Cloud-edge aware strategy	Improved Performance	Scalability Issue

The reviewed papers propose a variety of load balancing strategies to enhance the efficiency of cloud computing environments. These techniques are designed for a variety of use cases, such as Content Delivery Networks (CDNs), Kubernetes-based container clusters, and hybrid cloud peripheral data centers. The research highlights the importance of load balancing for optimizing resource utilization, response times, and user experiences. While the presented approaches demonstrate improved performance and scalability, further research is required to address potential limitations and investigate more intelligent load balancing strategies for cloud computing environments that are constantly evolving.

3 Methodology

The research technique is described in this section. In 3.1, the research flow is explained; in 3.2, System Design; and in 3.3, various tools and technologies are discussed.

This research will employ content delivery network and load balancing techniques to dynamically balance requests on Kubernetes clusters hosting web applications. This research provides solutions for the Kubernetes issues of insufficient load balancing, monitoring of service latency, and lack of security between services. SSL/TLS encryption is utilized to ensure secure data transit between CloudFront edge locations and end users, distributing incoming requests evenly across all available resources. The proposed solution provides higher efficiency and faster response times than conventional Kubernetes configurations. This solution addresses previously unsatisfied needs within the Kubernetes ecosystem.

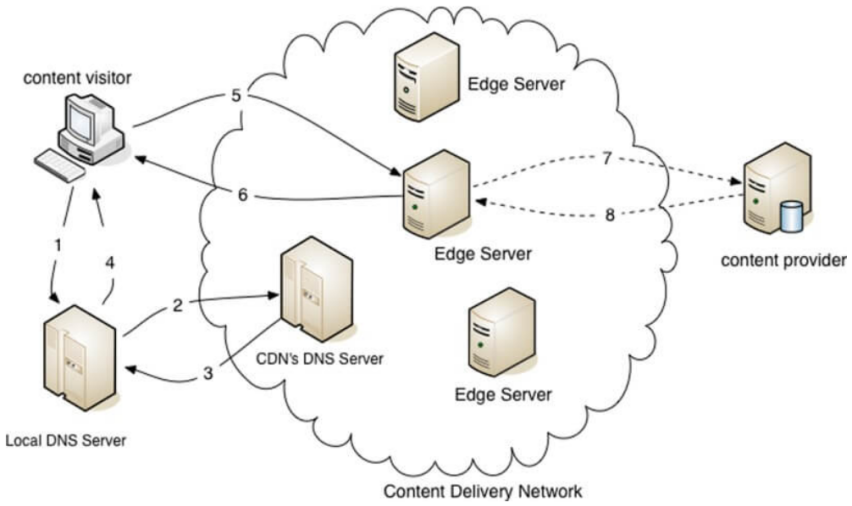

Figure 4: CDN Architecture

The figure shows the underlying structure of a content delivery network (CDN). The architecture of a CDN encompasses the network itself, peripheral servers, content servers, and user requests. DNS resolution directs user requests for web content to the nearest peripheral server. The Edge Server examines its cache and, as soon as the content is discovered (cache hit), distributes the content directly to the user. If the content is not cached (cache error), requests are forwarded to the origin server, which stores the original document. Common hardware can be replicated across multiple peripheral servers to expedite access, and load balancing algorithms maximize resource efficiency by distributing incoming requests among them. Moreover, CDNs prioritize secure data transmission via SSL/TLS encryption and continuously monitor and optimize delivery performance, including content expiration and recall procedures. Website content with improved reliability and quicker response times.

3.1 Research Flow

As shown in the above diagram the research process is divided into different phases, each of which elevate the objectives of the research:

Data Collection: This first stage involves collecting user information using an open-source VPN browser, specifically the IP address used to access the website.

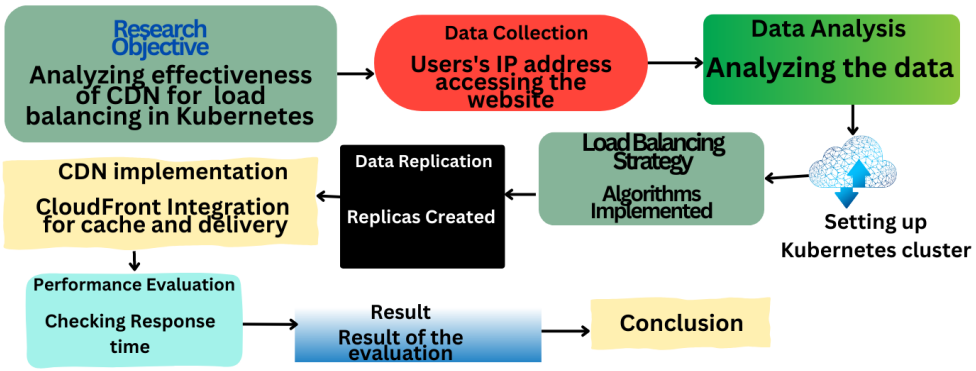

Figure 5: Flow of the research

Data Analysis:This step involves understanding load distribution patterns and identifying improvement opportunities. This procedure aids in comprehending how CloudFront integration affects load balancing.

Kubernetes Cluster: Kubernetes Cluster Management makes it feasible to dynamically balance load with a CDN. It enables efficient resource allocation and coordination to meet different needs.

Load Balancing Strategy:Using Kubernetes and CDN capabilities, the most advanced dynamic load balancing strategies are discovered at this point in the procedure. The objectives are to effectively distribute incoming requests and enhance system performance.

Data replication: Evaluation of the effects of data replication strategies on availability and resilience. The objective is to create data replication systems that are reliable and can improve system dependability.

CloudFront integration: CloudFront's integration into the system enables efficient load distribution and content delivery. This section investigates how CloudFront affects load balancing and overall system performance.

Performance Evaluation: To determine whether the suggested solution is practical, in this step, rigorous performance testing and comparisons with current load balancing algorithms are conducted. The evaluation seeks to determine progress and confirm the effectiveness of the suggested remedial action.

Result: The performance evaluation's results are displayed and analyzed. The research's successes and conclusions about load balancing efficiency and system performance are presented in this phase.

Conclusion: In the concluding phase, search results are summarized and based on the results, inferences are drawn. In addition to recommending the optimal configuration

for a load balancer, the significance of the proposed solution is also examined.

The research methodology intends to methodologically handle load balancing problems in cloud/Kubernetes setups while utilizing Content Delivery Networks for efficient and dynamic load distribution. The research objectives are achieved with the assistance of each phase, which also offers comprehensive data on the effectiveness of the recommended course of action.

3.2 System Design

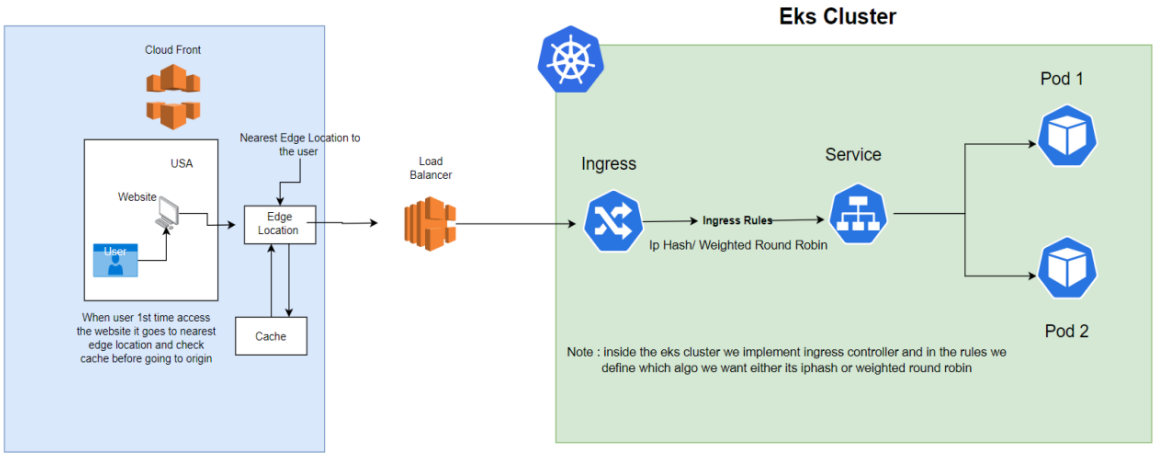

Figure 6: System Design

To maximize the transmission of web content utilizing Content distribution Networks (CDNs) and load balancing methods within a Kubernetes cluster, the following phases comprise the design methodology for this research project.

3.2.1 User Input and DNS Resolution

A user's request for a domain name is forwarded to the DNS server of their local ISP, which converts it into an IP address. The DNS server then queries the CloudFront DNS server.

3.2.2 CloudFront DNS Response

The DNS response provided by CloudFront depends on the location of the user. The CDN edge server is represented by the IP address of the edge location that is physically closest to the user.

3.2.3 CDN cache checks for content

After obtaining the DNS response, the CDN edge server checks the content cache for the requested file. If the content is already in the cache, the user receives it immediately, which reduces response time and results in a cache hit.

3.2.4 Integration of Load Balancer Algorithm and Kubernetes

If the requested data is not in the cache (a cache miss), the user request is transmitted to the Kubernetes cluster via the services. Here, load balancing algorithms such as IP hash and weighted round robin are utilized to optimally distribute incoming traffic among the Kubernetes nodes.

3.2.5 User response and data retrieval

The Kubernetes cluster's containers evaluate user requests and generate the necessary responses. The process of acquiring the data concludes with the transmission of the user's response via the CDN interface server.

3.2.6 Cache and Data Storage

When receiving repeated requests for the same piece of content, the CDN edge server stores the information in its cache. This caching method enhances the user experience by reducing response times for repeated requests for identical content.

By employing this design strategy, the proposed solution intends to reduce latency, improve reaction times, and enhance the overall performance of the online application, thereby effectively delivering content to users from various geographic locations. By combining CDNs, Kubernetes clusters, and load balancing algorithms, we can overcome the difficulties of dynamic load balancing, enhance data replication, and maximize the advantages of using cloud infrastructure for hosting web applications

3.3 Tools and Technologies Used

In this research, Elastic Kubernetes Service (EKS) on AWS was used to run the research using Linux-based instances of type e2 t3.medium. The EKS cluster consists of two nodes. The web application for the research was made using JavaScript and HTML.The following tools and technologies are employed:

3.3.1 Kubernetes

Kubernetes is utilized to coordinate the deployment, scaling, and operation of containerized applications. The complexity of production deployments required a substantial amount of human resources. Due to the necessity of fixing node problems and manually configuring services, scalability was challenging. By offering the necessary tools for managing immutable infrastructure, Kubernetes (K8S) solves these problems.

Kubernetes Componenets

A. Master Node

The cluster's management node, or master node, aids in managing deployments, accessing the API, and other related duties. In addition to an agent that interacts with the master, container runtime (Docker, rkt, etc.) may be handled by a number of worker nodes.

Here API server endpoint https://1405BC66D16554F52C7D7BEF2092BE87.gr7.us-east-2.eks.amazonaws.com allows to connect to the master node.

B. Node

In a Kubernetes cluster, the worker machines execute containers. Container Runtime, Kubelet, and Kube-proxy are all active on each node. Kubelet, an essential Kubernetes

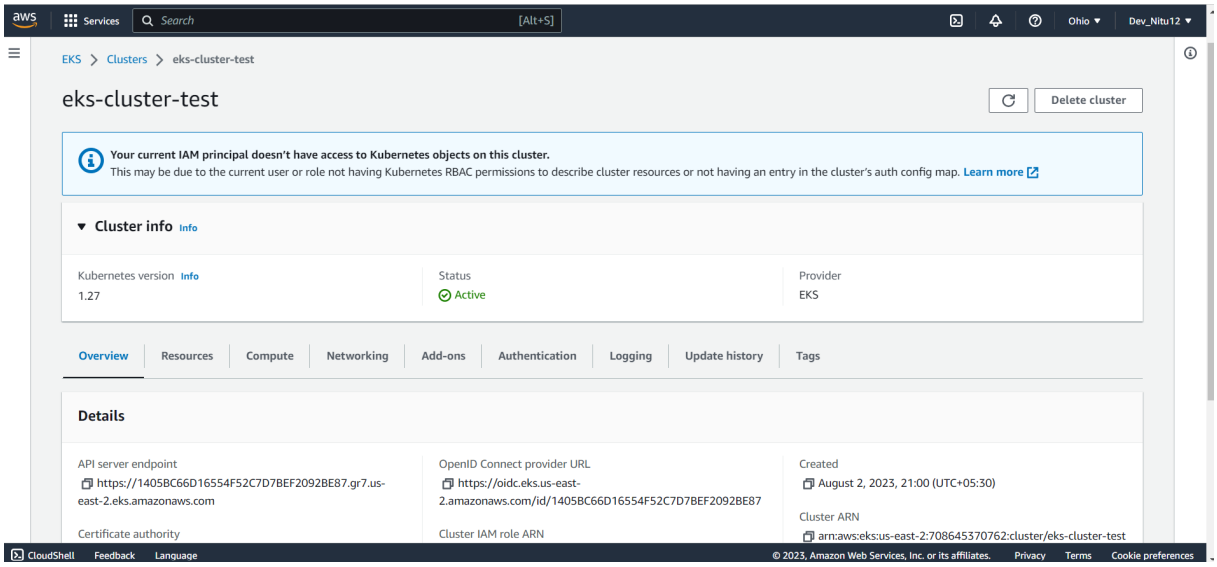

Figure 7: EKS Cluster and master node details

component, manages the cluster's distinct nodes (servers). It is installed on every node and functions as a node agent to ensure that the Pods and Containers specified in the cluster configuration are initiated and maintained properly.

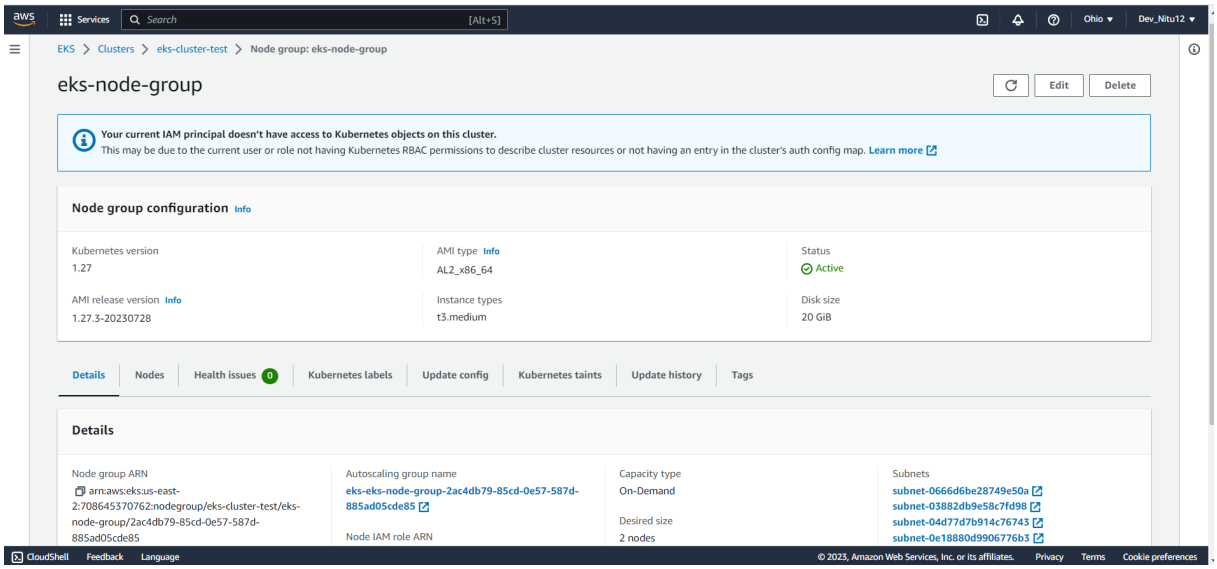

Figure 8: Node Group Details

3.3.2 CloudFront

AWS offers CloudFront, a service for worldwide CDN infrastructure. Using the web service Amazon CloudFront, static and dynamic online material, including.html,.css,.js, and picture files, may be provided to consumers more quickly. The highest potential performance for content delivery is provided by intelligently routing a user's request for content to the edge point with the lowest latency.

- CloudFront gives the user the requested content right away if it is already available at the closest edge point.

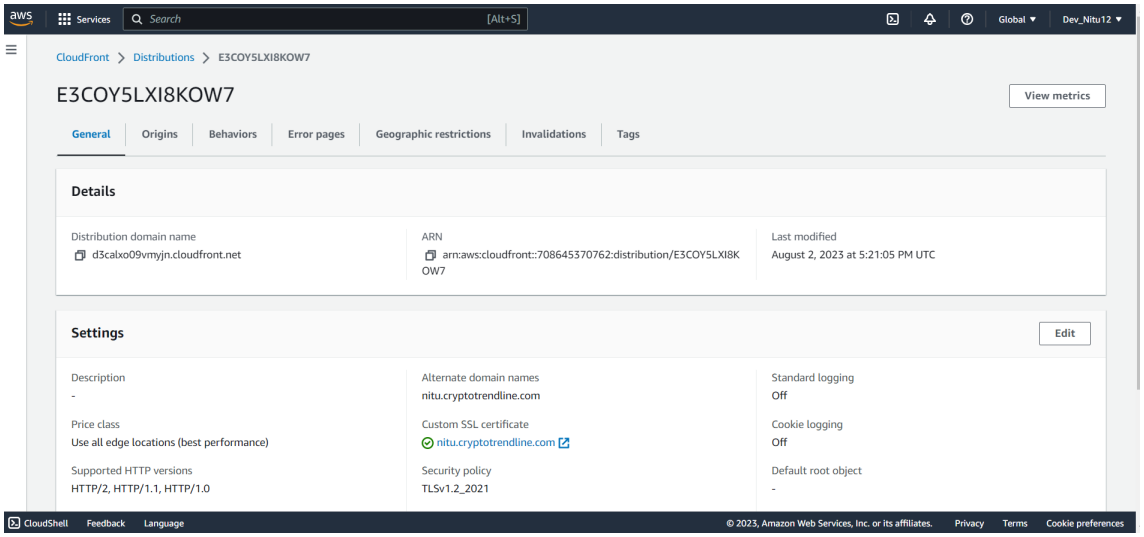

Figure 9: CloudFront Distribution on AWS

• CloudFront receives the content from a preset origin source, such as an Amazon S3 bucket, a MediaPackage channel, or an HTTP server designated as the authoritative source for the content, if it cannot be discovered at the closest edge point.Refer figure 12 showing CloudFront Distribution on AWS.

3.4 Load Balancing Algorithms

3.4.1 IP Hash and Weighted Round Robin

The load balancing methods utilized in the suggested architecture to distribute user requests among the Kubernetes pods are IP hash and weighted round-robin (WRR). The way each algorithm is used is as follows:

IP hash: Based on the client's source IP address, an IP hash load balancing algorithm determines which pod should handle the request. The algorithm uses the client's IP address to build a hash value and assigns it to a pod that is available for use. By guaranteeing that requests arriving from the same client are consistently routed to the same pod, this establishes session affinity and enables stateful applications to function without interruption.

Weighted Round-Robin (WRR): Weighted round-robin adds weights to each pod based on its performance or capacity as an additional load balancing strategy. A greater portion of the incoming requests are sent to pods with higher weights. The procedure rounds-robins the available pods while taking into account their predetermined weights. It can be especially helpful when some pods have greater processing capacity than others to spread the task efficiently.

The suggested approach includes these load balancing approaches to maximize resource utilization and enhance web application performance in Kubernetes setting. While WRR allows for the most efficient request allocation based on the capacity of the pods, IP hash assures session affinity for stateful apps.

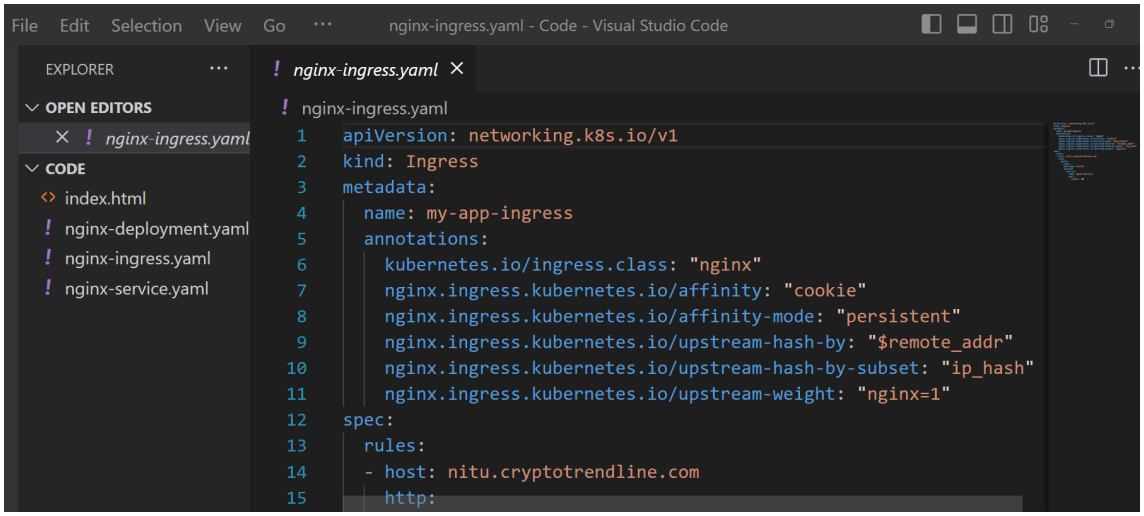

Figure 10: Implementation of Algorithms through Annotation

4 Design Specification

For dynamic load balancing on Kubernetes, the CDN has been integrated with the load balancing algorithms IP hash and Weighted Round Robin. As described in the previous sections, AWS CloudFront is used as a Content Delivery Network because it offers minimal latency in content delivery. IP hash and Weighted Round Robin are utilized to distribute demand across Kubernetes Pods using the demand Balancing Algorithms. AWS EKS is utilized to provide a highly available and resilient Kubernetes infrastructure.

AWS EKS Cluster	
Instance Type	t3.medium
Number of Nodes	2
Operating System	Linux
Total vCPU	4
Total Memory	8 GB
Cost	$0.0418/hr per node

Figure 11: AWS EKS cluster details

In this case, data replication is employed. The IP hash ensures that requests from identical clients with identical IP addresses are consistently routed to the same server. WRR allocates each server in the pool a weight based on its performance or capacity. However, when a server's capacity is exceeded or it is down for maintenance, it may be necessary to redirect queries to other servers. Session persistence is achieved through replication of sessions, replicating session data across multiple servers. Consequently, the session information is replicated on each server in the load balancing pool. When a request is sent to another server, the receiving server can access the session data and continue the session uninterruptedly. This method facilitates effective load distribution, scalability, and session continuity in distributed environments.

5 Implementation

During the implementation phase, a Content Delivery Network (CDN) was integrated with the dynamic load balancing system for Kubernetes-based web applications in order to implement the suggested solution. Configuring load-balancing algorithms and establishing the Kubernetes cluster were the concluding phases of the implementation process. The outcomes of the implementation of this proposed methodology are as follows:

5.1 Kubernetes Cluster Setup

On the AWS Elastic Kubernetes Service (EKS) platform, a Kubernetes cluster was created as the first step in the implementation procedure. Two nodes comprised the cluster in US East(Ohio): the master node and the worker node. The selection of AWS EKS was based on its capacity to facilitate Kubernetes cluster administration.

5.2 Web Application Development

The web application is developed using HTML and JavaScript, and its interface displays the user's IP address when they attempt to access a webpage. For user interaction and collecting the data the frontend of the application played a crucial role.

5.3 Data Collection and VPN Usage

In order to simulate user access from multiple locations and IP addresses, a browser-based VPN was utilized. This allowed for the simulation of user access scenarios from various geographic regions, allowing for a more comprehensive evaluation of the load balancing system's effectiveness.

5.4 Integration with AWS CloudFront

AWS CloudFront, a globally distributed content delivery network, was integrated in order to improve data delivery with decreased latency. This required configuring CloudFront to cache and distribute content efficiently, thereby enhancing the overall response times of the application .

5.5 Load Balancing Mechanism

Utilizing the Kubernetes Ingress resource, the implementation included load balancing mechanisms. This was accomplished by defining routing rules and load balancing algorithms within the Ingress.yaml configuration file. Annotations within the Ingress configuration made it easier to choose load-balancing algorithms such as IP Hash and Weighted Round Robin.

5.6 Performance Metrics and Monitoring

In order to evaluate performance and effectiveness of the proposed methodology ,AWS CloudWatch was used. The metrics involves CPU utilization, response time .The monitoring was carried out across different scenarios.

The implementation phase detailed the configuration of the Kubernetes cluster environment, integration with a content delivery network (CDN), web application development, data collection through a browser-based VPN, the creation of data replicas, and the establishment of load balancing algorithms.The configuration was intended to increase response time and provide users with a good browsing experience.

6 Evaluation

The Evaluation section is going to depict the investigations that have been conducted in this research. All the experiments are done on the AWS cloud platform, where the Elastic Kubernetes service is employed for building the Kubernetes Cluster. Nodes and four subnets are defined, with two being public and the other two being private subnets, respectively. The Kubernetes cluster is positioned in the Ohio US-East-2 region, and its version is 1.27. The CloudFront service is distributed globally. The cluster's configuration entails two nodes, and a Node group is utilized to establish a connection with the Master Node. A Virtual Private Network is established throughout the implementation process to support the Cluster. The EC2 instance employed here is of the t3.medium instance type.

6.1 Experiment 1

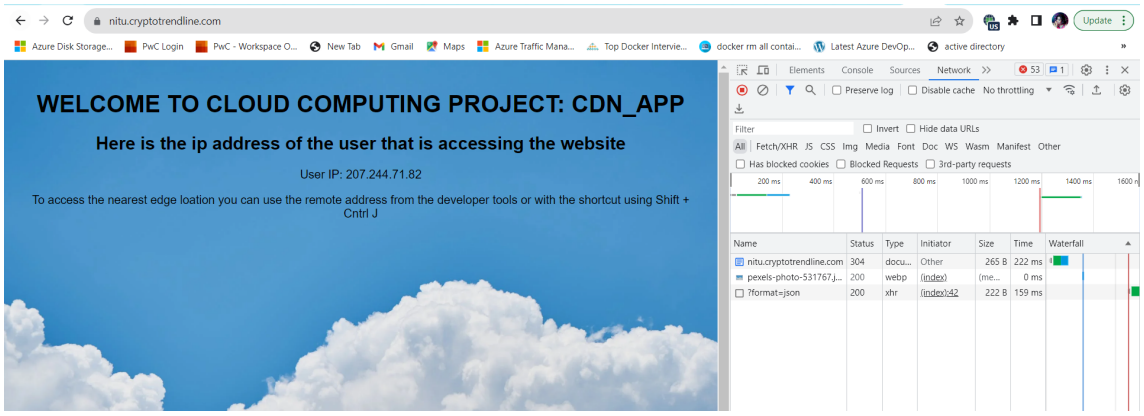

Figure 12: WebApplication Accessed for the first time from US-East-2 Region

In Experiment 1 The first experiment generated user access from the East Region of the United States. Initiating a session to access the Web Application was required. The response time during this initial access was logged to serve as an initial point of reference for future comparisons.

As shown in the illustrations, the observed response time is 222ms. In further experiment, I therefore investigated the use of a Content Delivery Network (CDN) to reduce response time.

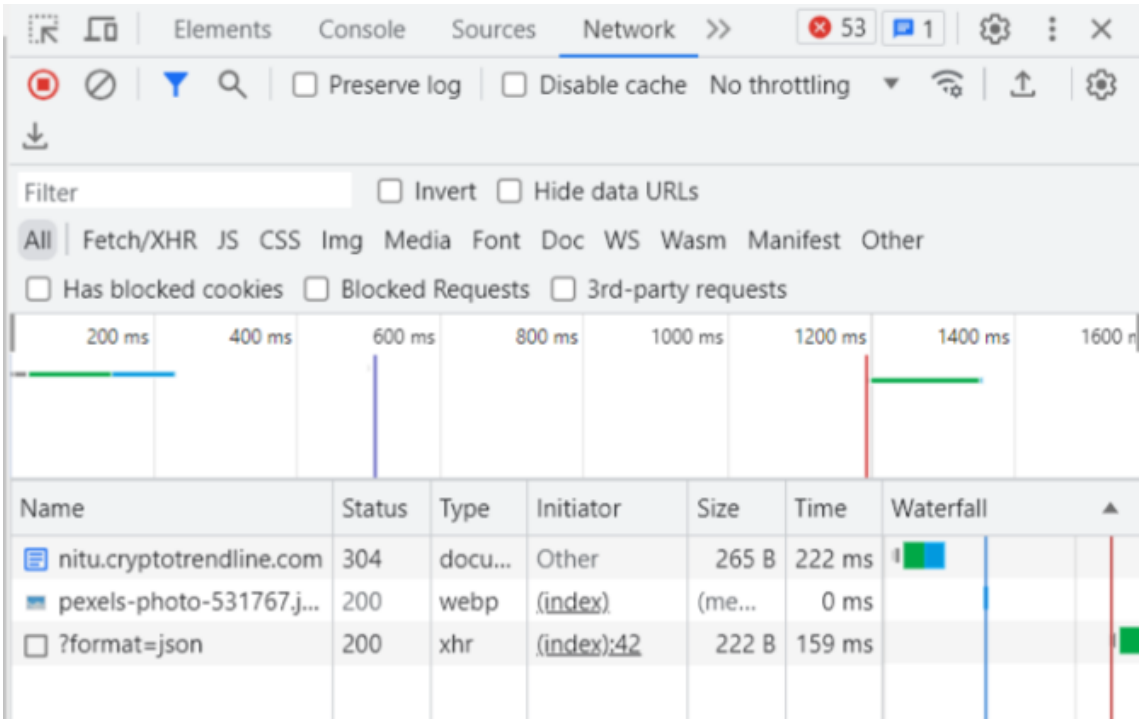

Figure 13: Response Time when WebApplication Accessed for the first time from US-East-2 Region

6.2 Experiment 2

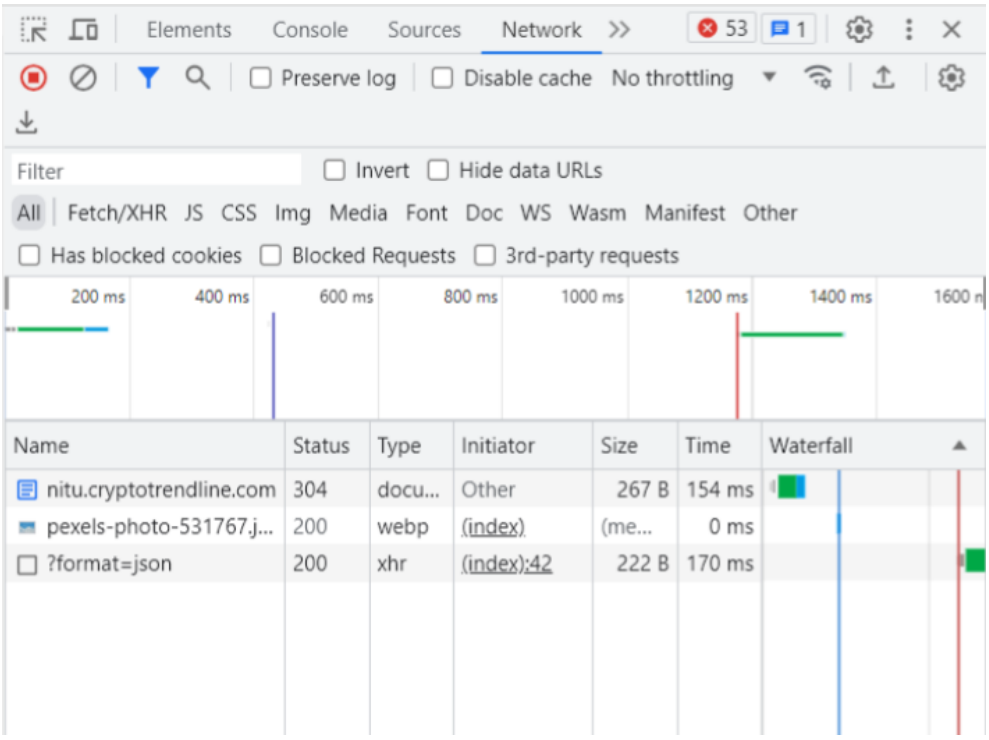

Figure 14: Response Time WebApplication Accessed for the second time from US-East-2 Region

Experiment 2 involves the utilization of a Content Delivery Network (CDN). To optimize content delivery and reduce latency, a content delivery network (CDN) with Load Balancing algorithm was integrated with Kubernetes. AWS CloudFront is used along with IPhash and Weighted Round Robin .The effect of CDN integration on response times and system performance as a whole was analyzed.The observed response is 154ms.

6.3 Experiment 3

In the third experiment, the efficiency of the load-balancing mechanism was evaluated. The ability of the load balancer to equally distribute incoming traffic across the nodes of the Kubernetes cluster was analyzed. The response time of the load balancer was monitored in order to evaluate its effectiveness under varying traffic loads.During a point in time, the target response time was recorded as 0.001 seconds upon first access. The response time for the web application decreased to 0.0008 seconds upon further access. This decrease in response time indicates a significant reduction in latency, demonstrating the efficiency of the implemented strategies, such as the utilization of a Content Delivery Network (CDN) and load balancing mechanisms.

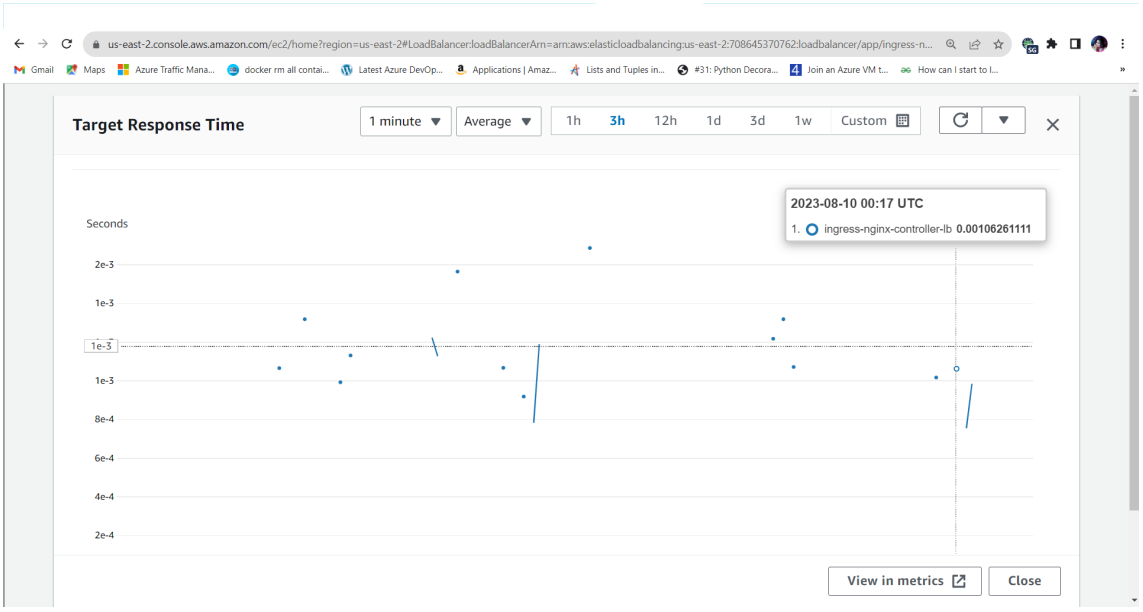

Figure 15: Response Time

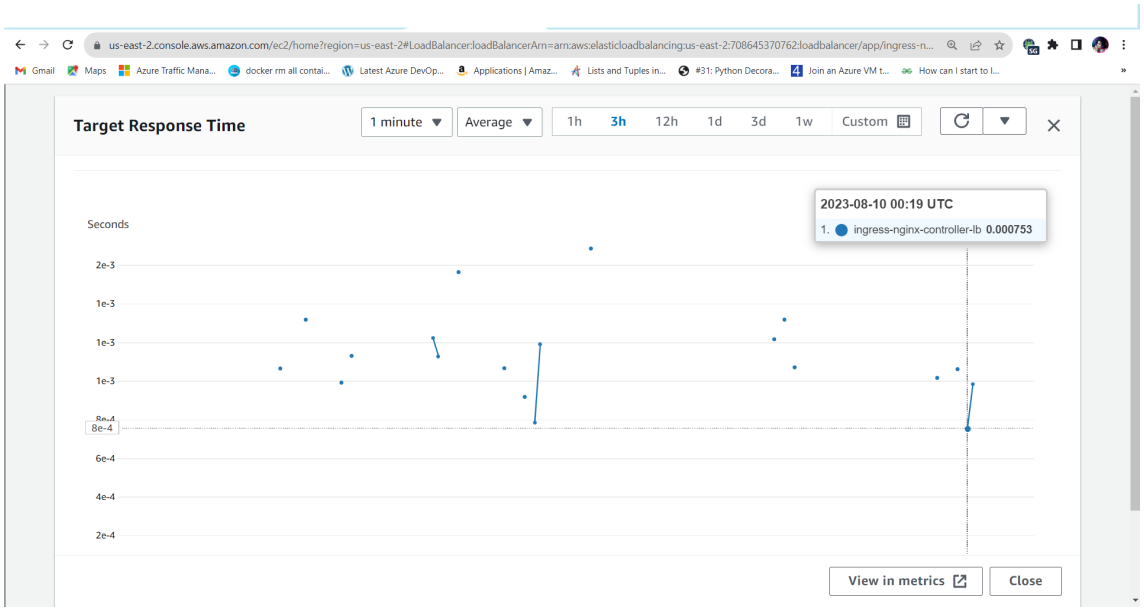

Figure 16: Response Time

6.4 Experiment 4

Experiment 4 shows CPU utilization, an additional experiment designed to evaluate CPU utilization across the Kubernetes cluster's nodes. In this analysis, CPU utilization was measured and contrasted on both nodes. On the respective nodes, it was determined that the CPU utilization rates were 1.68 and 1.75 percent, as shown in figure . As the difference between the two nodes remains minimal, this observation indicates that computing resources are distributed evenly.

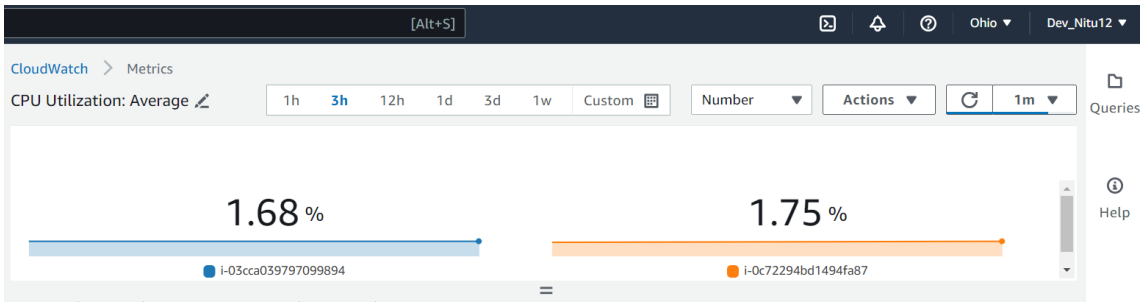

Figure 17: CPU utilization of Nodes

6.5 Discussion

In the comprehensive evaluation, the AWS Elastic Kubernetes Service (EKS) environment, which included AWS cloud, CloudFront CDN, and Kubernetes, was investigated. The Kubernetes cluster, strategically located in the US East region, operated on version 1.27, and was enhanced by a global CDN infrastructure. The cluster consisted of two nodes connected to the master node via a Node group. In addition to the default Virtual Private Cloud (VPC), a second VPC with four subnets in the same availability zone was introduced. The node server was a t3.medium instance of Amazon Web Services.

Experiment 1 established a baseline by creating user access from the US East region and measuring the initial Web Application response time of 222 milliseconds. The second experiment examined CDN integration, specifically CloudFront, which improved response times by optimizing content delivery and decreasing latency. Experiment 3 examined the efficiency of the load-balancing mechanism in distributing incoming traffic across cluster nodes. Monitoring the response time of the load balancer demonstrated a point with a target response time of 0.001 seconds, which then reduced to 0.0008 seconds with subsequent access. This significant reduction in latency demonstrates the effectiveness of the implemented strategies. Experiment 4 examined CPU utilization across nodes, indicating a balanced distribution of computational resources with 1.68% and 1.75% on respective nodes.

The evaluation demonstrated an effective integration, which improved response times, minimized latency, and optimized resource allocation. In accordance with the objectives of the research, this method produced dependable and effective performance in cloud-based Kubernetes environments.

7 Conclusion and Future Work

In conclusion,using Kubernetes and CDN integration, the present research examined dynamic load balancing in cloud environments. The proposed solution made use of AWS EKS with two t3.medium nodes, and experiments were conducted utilizing Browser VPN to generate user access data from multiple locations. The results demonstrated that the integration of CDN facilitated the caching of content on adjacent edge servers, resulting in decreased response times and increased latency. To achieve the intended response time, the load balancer managed incoming traffic effectively. The findings underscored the significance of combining Kubernetes and CDN in order to achieve optimal load balancing and enhance the performance of web applications.

The proposed solution can be optimized further in future work by fine-tuning load-balancing algorithms and exploring more sophisticated approaches to manage variable workloads. In addition, investigating the effect of various CDN configurations on performance could result in improved resource utilization and enhanced user experience.

8 URL

YouTube - https://youtu.be/kgdlXvDSx4U

WebApplication - https://nitu.cryptotrendline.com/

References

Afzal, S. and Kavitha, G. (2019). Load balancing in cloud computing – a hierarchical taxonomical classification, *Journal of Cloud Computing* **8**(22).
URL: *https://link.springer.com/article/10.1186/s13677-019-0146-7*

Chang, C. C., Yang, S. R., Yeh, E. H., Lin, P. and Jeng, J. Y. (2020). A kubernetes-based monitoring platform for dynamic cloud resource provisioning.

Chen, S., Chen, Z., Gu, S., Chen, B., Xie, J. and Guo, D. (2020). Load balance awared data sharing systems in heterogeneous edge environment, *2020 IEEE 26th International Conference on Parallel and Distributed Systems (ICPADS)*, pp. 132–139.

Darji, J. and Nakrani, T. (2021). Analysis of load balancing in cloud computing: Challenges and algorithms, *Advanced and Emerging Applications in Big Data and Machine Learning*, Universal Academic Books Publishers and Distributors.

Dong, Y., Xu, G., Zhang, M. and Meng, X. (2021). A high-efficient joint 'cloud-edge' aware strategy for task deployment and load balancing, *IEEE Access* **9**: 12791–12802.

Gupta, P. (n.d.). Reliability aware load balancing algorithm for content delivery network, *Conference Paper*.
URL: *https://www.researchgate.net/publication/283093121*

Liu, Q., Haihong, E. and Song, M. (2020). The design of multi-metric load balancer for kubernetes, *2020 International Conference on Inventive Computation Technologies (ICICT)*, IEEE, pp. 1114–1117.

Nagarajan, M. (2019). What is load balancer and how it works.
URL: *https://medium.com/@itIsMadhavan/what-is-load-balancer-and-how-it-works-f7796a230034*

Nakanishi, K., Suzuki, F., Ohzahata, S., Kato, T. et al. (2020). A container-based content delivery method for edge cloud over wide area network, *2020 International Conference on Information Networking (ICOIN)*.

Nguyen, Q.-M., Phan, L.-A. and Kim, T. (n.d.). Load-balancing of kubernetes-based edge computing infrastructure using resource adaptive proxy, *School of Information and Communication Engineering, Chungbuk National University* .

Nguyen, T. T., Yeom, Y. J., Kim, T., Park, D. H. and Kim, S. (2020). Horizontal pod autoscaling in kubernetes for elastic container orchestration, *Sensors* **20**(20): 4621.

Radhika, D. and Duraipandian, M. (2021). Load balancing in cloud computing using support vector machine and optimized dynamic task scheduling, *2021 9th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO)*, pp. 1–6.

Shafiq, D. A., Jhanjhi, N. Z. and Abdullah, A. (2021). Load balancing techniques in cloud computing environment: A review, School of Computer Science Engineering (SCE), Taylor's University, Subang Jaya, Malaysia.
URL: *https://expert.taylors.edu.my/file/rems/publication/10956688251.pdf*

Shitole, A. S. (2023). Dynamic load balancing of microservices in kubernetes clusters using service mesh.
URL: *https://norma.ncirl.ie/5943/1/abisheksanjayshitole.pdf*

Simić, M., Stojkov, M., Sladić, G. and Milosavljević, B. (2023). Crdts as replication strategy in large-scale edge distributed system: An overview, *Faculty of Technical Sciences, Novi Sad* .
URL: *http://www.eventiotic.com/eventiotic/files/Papers/URL/390587e1-82c6-4a9e-bef2-627701df841c.pdf*

Sitorus, S. P., Hasibuan, E. R. and Rohani (2021). Analysis performance of content delivery network by used rateless code method, *SINKRON: Jurnal Teknik dan Ilmu Komputer* **7**(4).

Varma, A., Varma, G. V. S., Varma, G. V. S. A., Varma, G. V. N. A. and Varma, G. V. N. A. (2023). Dynamic user routing for paid and free users in web applications using content delivery network (cdn), *International Journal for Research in Applied Science and Engineering Technology (IJRASET)* **11**(7).

Wei, X. and Wang, Y. (2023). Popularity-based data placement with load balancing in edge computing, *IEEE Transactions on Cloud Computing* **11**(1): 397–411.

Yang, H., Pan, H. and Ma, L. (2023). A review on software defined content delivery network: A novel combination of cdn and sdn, *IEEE Access* **11**: 43822–43843.

Zhang, J., Ren, R., Huang, C., Fei, X., Qun, W. and Cai, H. (2018). Service dependency based dynamic load balancing algorithm for container clusters, *2018 IEEE 15th International Conference on e-Business Engineering (ICEBE)*, IEEE, pp. 70–77.