Improve Load Balancing Performance and Efficiency Using Equally Spread Current Execution Algorithm working with response time clustering in Microservices

Research Project
Msc Cloud Computing

Prashant Bharat Agavane
Student ID: x18165435

School of Computing
National College of Ireland

Supervisor: Manuel Tova-Izquierdo

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Prashant Bharat Agavane
Student ID:	x18165435
Programme:	Msc Cloud Computing
Year:	2020
Module:	Research Project
Supervisor:	Manuel Tova-Izquierdo
Submission Due Date:	23/04/2020
Project Title:	Improve Load Balancing Performance and Efficiency Using Equally Spread Current Execution Algorithm working with response time clustering in Microservices
Word Count:	5959
Page Count:	19

Signature:	
Date:	22nd April 2020

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.	☐

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

Improve Load Balancing Performance and Efficiency Using Equally Spread Current Execution Algorithm working with response time clustering in Microservices

Prashant Bharat Agavane
x18165435

Abstract

Microservices architecture is the formation of a network of multiple services communicating with each other through service request and response messages. To achieve high availability and make business unit fault-tolerant, microservices use cloud computing services for hosting multiple instances of each service distributed geographically. In the process of handling these service API calls, an effecting load-balancing technique play a crucial role to achieve efficient performance by reducing response time. After a detailed study of available load balancing techniques and algorithms in the literature review carried out, we have proposed a combined load balancing approach that uses Equally Spread Current Execution(ESCE) algorithm and response time clustering method based on historical response time for load balancing. In this project, research on finding a performance gap between the Round robin algorithm and the proposed load balancing approach has been carried out. For this performance comparison, I have recorded response time data of multiple instances and scenarios for statistical analysis. This statistical analysis concludes the effectiveness of proposed implementation for minimizing processing delay by choosing the optimal microservice/instance.

1 Introduction

To develop a large scale software application(s), a shift has been observed in recent years regarding the development approach. The previously monolithic approach of software development was most practiced by the researchers and practitioners which asked to develop the complete business logic as one complete whole chunk of code. The implementation of such a large piece of code was very difficult because of large size, complexity, number of interdependencies among the business logic and strongly coupled modules Niu et al. (2018). For this purpose, highly skilled developers were required who completely understand and have the experience to handle a large chunk of code. In case of any fault or bug, testing became a huge task in terms of time and effort, because of numerous dependencies that made it difficult to trace the fault and recover the operations.

1.1 Micro-service Architecture

In order to mitigate the aforementioned difficulties, microservices architecture was proposed, that promotes the notion that small collection of services should be made, that are independent and isolated from each other, have their data so that they can be scalable and elastic in case any failure occurs Rusek et al. (2016). According to Fazio et al. (2016), the microservice architecture is lightweight and can be easily updated and deployed because each application is considered as a single entity. The endpoints in the microservice architecture are small and data can be communicated through them easily. The largest example of the implementation architecture, according to Fazio et al. (2016) can be cloud applications. By adopting microservices architecture many advantages can be gained such as loose coupling Bravetti et al. (2019), rapid recovery in case of disaster and increased reliability because in case of failure only a small portion of service will get affected, the whole system will not get compromised because of a single failure Rusek and Landmesser (2018). With the ease of using microservices, there are some challenges associated with it as well. As the size of application increases, the number of API calls also increases and to handle this load balancer is required that can manage API calls across the architecture.

1.2 Service Registry in microservices

In a monolithic architecture, there were code level dependencies that were displaced in the microservices architecture with service level dependencies. According to Balalaie et al. (2018) it can't be said that these services have interdependencies, because first, these services have to perform small tasks which are not dependent on other tasks and secondly a resource pool is created, where each developed service has small operational code so that they can perform independently. In case of workload on any of service increases or decreases, these services can be scaled up and more than one instance of a service is creating that are deployed on multiple servers. In order to maintain the addresses of the instances that have been created for the services, a service registry is maintained in a microservice architecture. The signal from each instance is periodically received in the service registry. There are two cases in which an instance gets removed from the service registry, either it has been removed manually or during a specific period that instance has failed to send the signal.

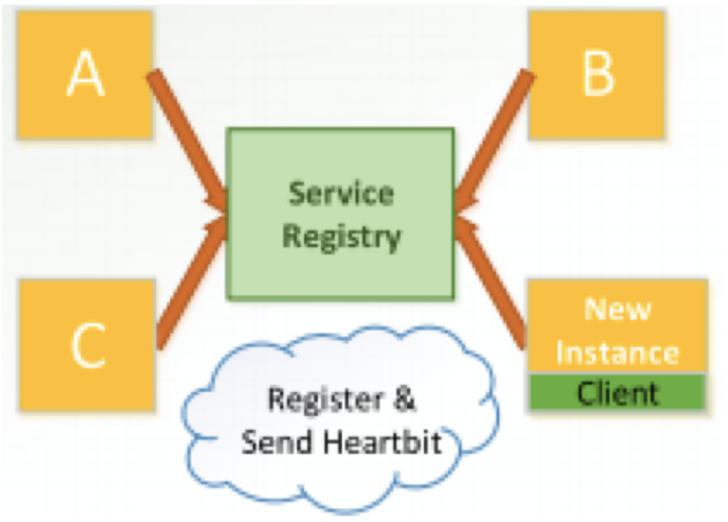

Figure 1: Service Registry

1.3 Architectural challenges in Microservice Architecture

Alshuqayran et al. (2016) have done an interesting mapping study on the challenges that are associated with microservices architecture. According to this study, microservice architecture is a novel field and there is a gap regarding its application among the researchers and practitioners. After performing an extensive review they have found that security is the largest challenge and hasn't got much attention from the researchers. Among other challenges found by the study includes; tracing the request of microservice among all business function hops, performance monitoring of the application, deployment problems and generic performance-related issues such as QoS issues, etc. According to Yu et al. (2019) management is the biggest challenge faced in the microservice architecture. According to their study, there should be fewer loads present on the instances of the microservices because it provides better robustness in case if an instance faces failure or there is a need to make a change in the instance. Because of the interdependencies, maintaining the load on all instances is very hard since a change in the load of one instance can cause unbalance in many other connected instances.

1.4 Load balancers in microservices

As discussed above, load balancing is an important aspect of microservices architecture for the development of a business unit. Communication between the services is done via API calls, so there are multiple instances and load has to be maintained between all available instances. For this purpose load balancing algorithms are designed. Numerous approaches exist in the literature regarding load balancing algorithms and there are two major categories in which these algorithms can be divided. These are server-side load balancing approach and client-side load balancing approach. According to Duc et al. (2019) load balancing provides a mechanism that can help in providing better throughput of the system as well as in achieving enhanced efficiency with the help of efficient utilization of the resources.

In the network services it is essential to get high performance and for this purpose cluster technology is used because a single server can't handle all tasks required to complete the operation. Among a cluster there are several servers, utilization of resources is very important so that requests coming on these servers can be handled in an equal manner. When the resources are not managed on the servers evenly, some become overused while some remain idle that contributes to minimizing the cluster performance. In order to cater to this problem load balancing algorithms are used so that all tasks can be handled evenly thus increasing the capacity of processing and quality of the service at the same time.

According to research, load balancing serves the purpose of distributing the load among available resources so that throughput can be increased while decreasing the response time at the same time. It also helps in performance improvement and makes utilization of resources efficient. Yi et al. (2018), have enlisted several load balancing algorithms and compared all of them with their proposed load balancing approach. According to this study, the researchers have used Traditional Round Robin and Random Algorithms, task scheduling optimization algorithm, Weights Fixed Load Balancing Algorithm, Server Dynamic Allocation Weight Algorithms, etc.

1.5 Research questions

- To what extend can we improvise response time of client service consumer request in microservices architecture by changing load balancer techniques and algorithm?
- How effectively ESCE algorithm based on clustering mechanism can be used to allocate the tasks efficiently based on the historical response time data?

1.6 Objective and Motivation

The objective of this research is to reduce the response time of service being called and improvising load balancer technique by implementing clusters and ESCD algorithm. As stated above, the actual challenge that exists is microservice architecture is balancing the load among the instances of microservices and increasing the performance of the system by minimizing the response time in the cluster of servers. So this study will implement dynamic load balancing algorithms, specifically using Equally Spread Current Execution Algorithm.

As the microservice is a service, which is running on the server and it takes some inputs and as an output, it will provide some services. The specified inputs and output of this architecture can be in various formats such as JSON, XML, and UTF-8 string. The client can access the microservice using the HTTP protocol i.e. issuing requests to the server and getting a response in return. While the request is handled by the server, there is always a latency or delay between the client sending requests and receiving the responses because of a number of reasons. It might be because of travel time for signal, processing time on the server, irregularities in the network, and load on the server, etc.

This study will take only two parameters under consideration including:

- **Network Latency:** Time delay caused by the network.
- **Processing Time:** Time required for processing the inputs.

In order to provide the best QoS to the client, the latency needs to be reduced that can be achieved by assigning the task to the closed server and processing time can be reduced by assigning the task to the highest performing server. Network latency and processing time both are dynamic factors and change values with time. Sometimes the nearest server is overloaded so the benefits that can be achieved from less network delay are neutralized by excess in processing time. Sometimes the cost of network delay on a farther computer is mitigated by the decrease in processing time. In some cases, the extra processing time on a nearby server is compensated by the decrease in network delay due to less distance.

Now the actual problem arises when there is a need to predict the values of network delay and processing delay. Mostly when a web-service is accessing it is not accessed directly, rather its front end is accessed which is a reverse proxy server. This reverse proxy sends our request to a backend server, which then sends it to an instance of that web-service running as a cloud instance. The backend server will use a service registry to maintain the record of allocation of the task to the micro-service, for example, instance1 = task34. The problem is how to determine which instance should be allocated for the task. Directly getting absolute values of network delay and processing delay is impractical as it depends on a lot of factors (for example resources allocated to the instance, geographical location on the server, network load, server load, performance of hardware, etc).

Existing static load balancing algorithms use the pre-programmed simple approach like round robin, min-max but these approaches can perform well in simple scenarios. Like all the server is at similar distances, have nearly same hardware capability and network load is uniformly spread. But, these ideal conditions are rarely seen in the real world. So the Solution for aforementioned problem(s) can be achieved with the help of implementing Equally Spread Current Excution Algorithm. The ESCE algorithm will help in allocating the task to the micro- service instance such that the number of task is inversely proportional to the response time. The instance that delivers faster response time will get more task and those with slower response time will get less number of tasks. The response time is found by averaging the response times in near past.

2 Related Work

Fazio et al. (2016) have provided a complete insight into the concepts that are required to understand the microservice architecture and its applications. According to this study, using this architecture can help in developing such applications that have high scalability and availability. The virtualization concept of the cloud-based environment explains the best applicability of microservice architecture. Amongst all applications using a microservice architecture, load balancing and performance have proven to be the biggest challenges while its implementation. In this section, the extensive work of the researchers will be presented, which will provide all available techniques that have been implemented for the purpose of load balancing in the microservices architecture.

Load balancing tends to optimize the resources over the computer clusters with the help of network links to increase throughput while decreasing the response time. Load balancing maintains the load in such a manner that any resource will not become overloaded so that data can be exchanged without delay Alamin et al. (2017).

2.1 Load balancing techniques & algorithms

Gan et al. (2019) states that two types of load balancing approaches have been used by the practitioners including randomized load balancing techniques as well as deterministic load balancing techniques. There are many classifications of load balancing algorithms. For instance, Different researchers have done an extensive survey on the load balancing algorithms. According to the authors, majorly they can be characterized as a state of the system and based on who initiated the process. Under the state of the system characterization, they can be further divided into static and dynamic algorithms. Static algorithms can be either optimal or non-optimal, whereas dynamic can be distributed or non-distributed. Authors have made four different categories and enlisted several techniques including the Cuckoo Optimization Algorithm (COA), load-balancing strategy using genetic algorithm and Ant Colony Optimization (ACO), honeybee-based load balancing technique called HBB-L, and Agent-based load balancing techniques.

Parida and Panchal (2018) also categorize load balancing algorithms as static (suitable for the environment that requires little variation in load balancing) and dynamic (suitable for applications where flexibility is required and marge variation in load balancing is required). A similar classification has been stated by Duc et al. (2019) where static balancing tends to place workloads in a pre-determined way whereas in dynamic balancing, the workloads are allocated at run time dynamically. This study also states that static methods have proven to be more stable but fail to provide optimal solutions when there exist some variations in the workload. All existing load balancing algorithms attempt to provide better opportunities for workload distribution mechanism among all available resources so that an optimal solution can be provided, keeping response time and cost of data transfer less.

Yi et al. (2018) have explained the complete architecture of microservices and proposed a Microservice cluster load balancing framework, which can be easily understood in the Figure 2.

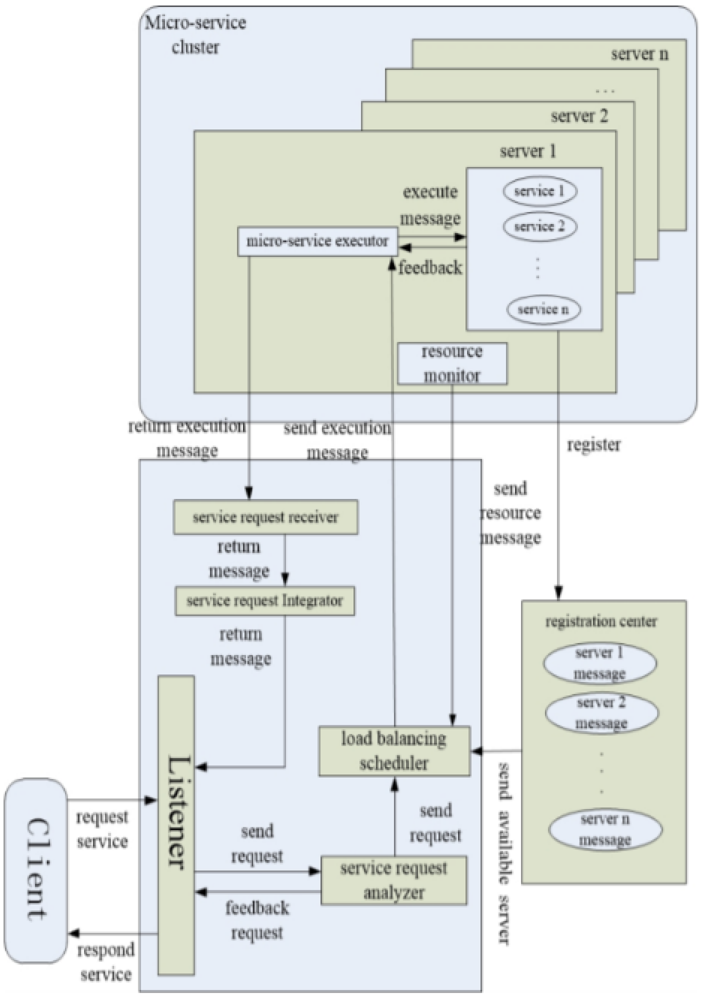

Figure 2: Microservice Architecture Proposed by Yi et al. (2018)

Another collection of load balancing algorithms used by Mesbahi et al. (2016) revolves around the virtual machine. According to the authors, these algorithms include the Round Robin algorithm, Throttled and (ESCEL) Equally Spread Current Execution Load. Khatavkar and Boopathy (2017) have enlisted several load balancing algorithms including min-max, max-min, round-robin, OLB, minimum execution time and minimum completion time. Additionally, this study has provided an extensive review of the pros and cons of all mentioned algorithms with respect to some specified parameters.

2.2 Static and Dynamic Load Balancing Algorithm

After carefully analyzing the existing literature for load balancing algorithms, a broad category found in all studies divides the algorithms in static and dynamic and can be further divided into more categories. For instance, Sajjan and Yashwantrao (2017) have performed an extensive review of the load balancing techniques and algorithms used by the researchers for load balancing. The two broad categories, according to this study are static and dynamic algorithms. As far as static algorithms are concerned, it includes round-robin, min-max, and min-min algorithms that have been used. On the other hand, dynamic algorithms perform best for load balancing and can be implemented in distributed as well as non-distributed systems. The dynamic algorithms that have been used by other researchers, found by this study include, Honeybee Foraging Behavior, Throttled LB algorithm, ESCE (Equally Spread Current Execution) LB Algorithm, Ant Colony LB Algorithm, Biased Random Sampling LB Algorithm, and Modified Throttled LB Algorithm. A very brief analysis of these algorithms has also been provided by the authors of this study and different associated advantages and disadvantages of each algorithm have been presented. The analysis criteria for evaluating the algorithms chosen by this study are virtual machines in which efficient load balancing is required.

2.2.1 Static Load Balancing Algorithm

Khatavkar and Boopathy (2017) have worked on static algorithms for the purpose of load balancing. These are max-min and min-max algorithms and proposed a meta-heuristic BAT algorithm. The authors of this study have combined weighted round-robin and max-min algorithm and proposed a weighted max-min algorithm for load balancing. This study has considered two parameters of cloud computing i.e. response and waiting time of the hosts on the virtual machines. The results established in this study prove that the proposed algorithm outperforms another algorithm.

2.2.2 Dynamic load balancing algorithms

Parida and Panchal (2018) have used a dynamic load balancing algorithm for the purpose of load balancing. According to the authors, in the cloud-based environment, it becomes very hard to balance the load among different nodes, as it has been observed that few nodes remain idle, or some become overloaded. In order to equally distribute the load among nodes in cloud computing, the dynamic load balancing algorithm, with the help of a machine learning technique can solve this problem. This study has also used virtualization as a medium to test the proposed solution, the machine learning will help to collect the dynamics of different data of the virtual machines and balance the load among machines by working on the queue principle. The allocation of resources in this scenario is done by using a round-robin manner.

Yi et al. (2018) have also worked on a load balancing algorithm that is coarse-grained, and proposed a new dynamic weight load balancing algorithm upon the framework of the microservice cluster. The dynamic weight load balancing algorithm uses the data of physical servers and the allocation of resources among them and by using the parameters such as different rates of several server connections, rate of memory utilization and provides an efficient way of balancing the load among different servers.

Alamin et al. (2017) have proposed a new merging algorithm by combining two best dynamic algorithms including Equally Spread Current Execution algorithm and throttled algorithm. This study has used virtualization as a testbed, in which the index of virtual machines is maintained. The proposed algorithm of this study has been validated with the help of simulations and authors have recorded response time and Processing Cost with Throttled Algorithm and Equally Spread Current Execution Algorithm and then compared it with their proposed algorithm. Comparative analysis shows that the proposed algorithm outperforms the other two algorithms.

2.3 Machine Learning techniques for Load Balancing

Gan et al. (2019) have developed a debugging system for cloud computing microservices where performance is usually degraded because of Quality of Service (QoS) constraints. The system proposed by the authors is called Seer that makes use of deep learning techniques in order to translate violations occurring in QoS. According to this study the proposed tool has tested local clusters as well as large scale applications that make use of microservices and proved that this system can provide deep insight in predicting performance levels that change because of load variation required by it.

Parida and Panchal (2018) have developed a machine learning-based load balancing algorithm and allocated the resources to virtual machines based on their proposed methodology where the load is first assigned where little variation is detected. Duc et al. (2019) have done an extensive survey on the usage of machine learning methodologies that can help in the analysis of load balancing and resource optimization. From existing literature, they have shown that in the last decade, maximum studies have implemented machine learning methodologies and tend to provide better results as compared to traditional methods of load balancing in terms of providing better prediction of the workloads.

2.4 Other approaches for load balancing found in literature survey

Rusek and Landmesser (2018) have worked on the load balancing algorithm and implemented it on the microservices that are running in the virtualization containers and proved that their decentralized swarm-like algorithm provides better performance than other existing methods. According to the authors, virtualization containers provide a best practice example of practicing the microservices, because recently researchers have been proposing such algorithms that tend to improve the performance of the containers that are running on different machines, and containers in this scenario happen to act as a swarm of agents. This study has established results with the help of existing data as well as ran numerical simulations and proved that using load balancing algorithms in a cloud-based environment outperform other existing algorithms.

Rusek et al. (2016) have proposed a decentralized system using the load balancing algorithm and proved that an improvement in the performance can be observed by its application. The study has also applied the load balancing approach to the microservices that are containerized. The algorithm has been validated by applying it to cloud-based applications as well as processes that implement mobile agent intelligence. The algorithm proposed in this study has taken inspiration from the swarm-like decentralized algorithm using the OpenVZ containerization software and tasks have been executed inside the cloud-based environment. According to Gan et al. (2019), the biggest challenge faced by microservice architecture is interdependencies among the service's instances and balancing the load on such services. They have made use of a model that is based on the directed acyclic graph (DAG) that helps in describing the interdependencies among microservices and helps in balancing the load by minimizing the load on those services that have a maximum load. The proposed methodology of this study also caters to the problem of QoS based load balancing problem. For implementation purposes, this study takes the microservices architecture of the Internet of Things application development. The algorithm proposed in this study has proved that their proposed algorithm called QoS-aware load balancing provides better performance as compared to heuristics.

Lloyd et al. (2018) have worked in different paradigm i.e. serverless computing as opposed to all other studies where they have used the medium of cloud computing or virtual machines etc. According to this study, the load balancing in serverless platforms occurs automatically thus increasing the performance of the microservices. Authors of this study claim that using the serverless computing environment tend to better deployment of the applications.

Niu et al. (2018) have provided extensive insight into how load balancing has been done in the microservices architecture. They have proposed a hybrid technique called a chain-oriented load balancing algorithm (COLBA) that helps to maintain the load balancing in microservices by taking two considerations i.e. requests coming in a heterogeneous manner and inter-chain competition. By performing experiments and comparing their proposed algorithms to the existing load balancing approaches, authors have proved that response time in COLBA has been reduced to 13%. This study provides an interesting comparison between instance oriented load balancing as well as microservice oriented load balancing approaches.

Bravetti et al. (2019) have worked on the automated deployment and proposed a solution so that reconfiguration of microservices can be done by specifying the relationship between different microservices, and their dependencies (both strong and weak dependencies). Based on the computations performed by the proposed model of Bravetti et al. (2019) an optimal deployment solution is provided which provides decisions threefold; which microservice will be deployed next and its distribution, how the connection among the microservices will be established, and overall plan of deployment. The proposed solution is tested over the real-time problem of the microservices architecture of E-mail processing and proved that their solution generates optimal results. Filip et al. (2018) have worked on the scheduling of microservices and have proposed a hybrid scheduling algorithm that makes use of a collection of primitive microservices. This approach aims to develop a balanced approach for the task distribution by making use of the analysis of tasks calculated in one step before and making a pool of all available resources required to complete the tasks. The authors of this study have implemented their proposed algorithm on real-time scenarios and established the fact their algorithm cost-effective and provides an optimized scheduling solution.

The following table provides the founded load balancing techniques and algorithms in the literature along with the studies shown in Table 1

Paper Title	Publish Year	Method	Advantages	Future Scope / Disadvantages / Gaps
An Efficient Dynamic Load Balancing Algorithm Using Machine Learning Technique in Cloud Environment	2018	Linear regression	Compared the result with round robin and shown that their proposed algorithm is much better	Only calculated the results in respect to execution time, not considered response time. How they generated the data that is also not explained clearly
Dynamic Weight Based Load Balancing for Microservice Cluster	2018	Proposed Algorithm	Compared the results with traditional round robin, random and fixed weight load balancing algorithm.	Not compared the results with other dynamic algorithms such as WSQ, Queue length etc.
Queue-Waiting-Time Based Load Balancing Algorithm for Fine-Grain Microservices	2018	Online learning algo of time weight (OLTW) and Short Queue waiting time load balancing algorithm (SQLB).	Compared the results with RR, WRR and QL. They considered dynamic weight load algorithms.	Works well when the number of microservices are not much.
Load Balancing for Interdependent IoT Microservices	2019	Directed acyclic graph (DAG)	Provided the efficient solution for interdependent microservice main used for IoT applications	This algorithm is mainly designed for load dependency among microservice, this method cannot perform well for independent microservices.

Table 1 : Comparative Analysis of different load balancing algorithms

Literature review carried out in this paper indicates several advantages and gaps for each algorithm/load balancing technique. As QoS is combination of network latency and processing time, it is must to get best output from both parameters. Gap in performance of any one of them leads to degradation of overall QoS. In order to summarize the gaps in algorithms/techniques studied in literature review, none of them concentrates on capturing response time and using it as a parameter to design load balancing algorithm/technique. The studied algorithms are more relied on improvement of individual parameters like execution time , vm load management, queue management, etc. All of these algorithms tends to predictive performance of microservices by improvising the parameters affecting the QoS. The proposed algorithm relies more on working with actual captured data(the response time clustering) rather than improvisation of individual parameter to get best results.

3 Methodology

In this section, we will discuss the various methods which have been used to develop the proposed system. The clustering method is the core of our proposed architecture. where we are comparing our results with the round-robin algorithm. A detailed analysis will be performed about each technique to understand the concepts in more detail.

3.1 Round-Robin Algorithm

The round-robin algorithm is the most common and efficient algorithm to execute the task based on the time quantum. Based on the time slices round-robin distributes the tasks equally in the circular order. The one disadvantage with the round-robin algorithm is, if the time quantum is very high it works as a FIFO (First in First out)algorithm. Round robin is highly used in a cloud computing environment as a scheduling algorithm it can prevent the system from a deadlock. In the real world environment, all the instances/micro-services can not have the same computing capabilities. If the machine with low computing capability has been assigned with more tasks, it will take more time to execute it. As a result, it will delay the response time.

3.2 Equally Spread Current Execution Algorithm (ESCE)

ESCE algorithm is mainly based on the spread spectrum approach. Currently, every task queue of VM will be 0. ESCE Job scheduler maintains the VM allocation table, which is used to store the VM id and number of tasks allocated to VM. Each Virtual Machine maintains a task queue to which stores the tasks for execution. ESCE Job scheduler checks the task count of VM. If the active task count for respective VM is lowest will be assigned for more task. If there is more than one VM whose task count value is lowest it will assign to the first one. Here the target of ESCE algorithm to assign the task based on the task count for specific VM.

3.3 K-means Clustering Algorithm

K-means clustering is an iterative algorithm and a kind of unsupervised algorithm, mainly used for the unlabelled dataset. K-means clustering algorithm is used to partition the data based on non-overlapping subgroups or clusters based on the similarity and common pattern. It is very helpful for applications such as document clustering, market-segmentation, and image segmentation. K-means clustering is based on the centroid and uses the euclidean distance algorithm. The number k should be specified before forming

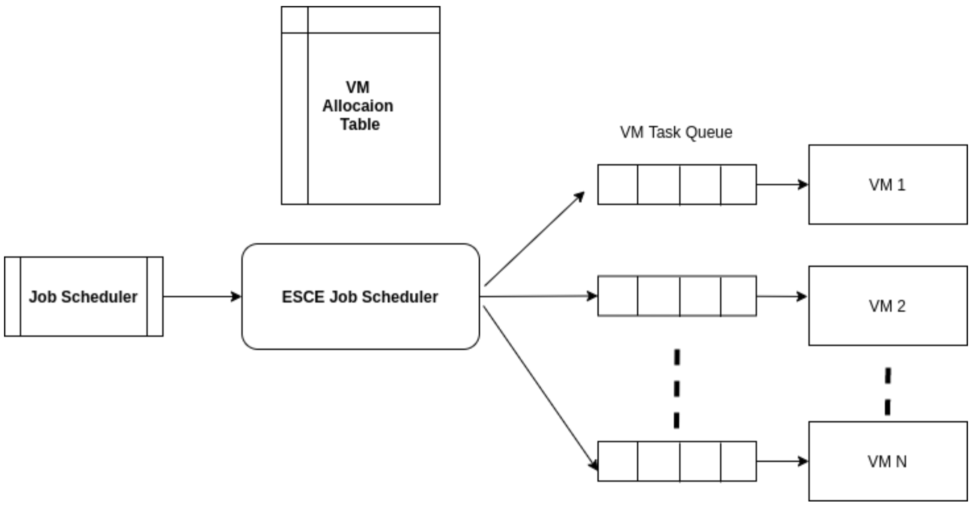

Figure 3: ESCE Algorithm

a cluster. In order to find out the optimal number of clusters Elbow's method can be used.

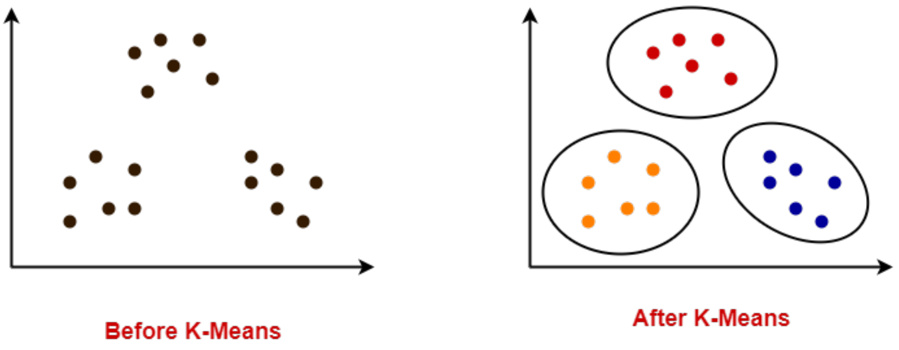

Figure 4: K-means Clustering

4 Design Specification

This section represents the flow of execution of our proposed work for balancing the load over micro-services using the response time. This work separates the complete system into smaller components, which handles the individual tasks via communication channels. The main motive of this method is to reduce the response time of service. Network delay and Processing delay are the two major factors that are liable for increases in response time of service. This work uses Equally Spread Current Execution Algorithm to assign the tasks optimally in order to reduce the response time. The proposed architecture for Load balancing in Micro-service is shown in Figure 5

4.1 Task Generator

Task Generator is the first component of Proposed architecture. Task Generator is used to simulate variations of load on the system. It can generate random size tasks at random intervals of time. Each task has a load factor that decides the required time for processing. The generated tasks are later added to the task queue.

4.2 Task Queue

Task Queue behaves the same as the Queue data structure, it follows the FIFO (First In First Out) order. All the tasks generated from the task generator are store in the task queue. Task Queue is used to handle the incoming and outgoing tasks. The tasks stored in the task queue are later fetched by the Distributor.

4.3 Distributor

The distributor is an important component of this system and responsible for assigning tasks to workers or micro-services. By default, the distributor assigns the task in a Round-robin pattern. The Round-robin uses the time quantum method to assigns the tasks to micro-services. The traditional distributor does not use the clustering method. Our Proposed work uses the service cluster, where we cluster the response time of different microservices with the help of the K-means clustering algorithm. This algorithm provides the decision making capability to a system where we can minimize the processing delay by assigning the tasks to best performing microservice instance.

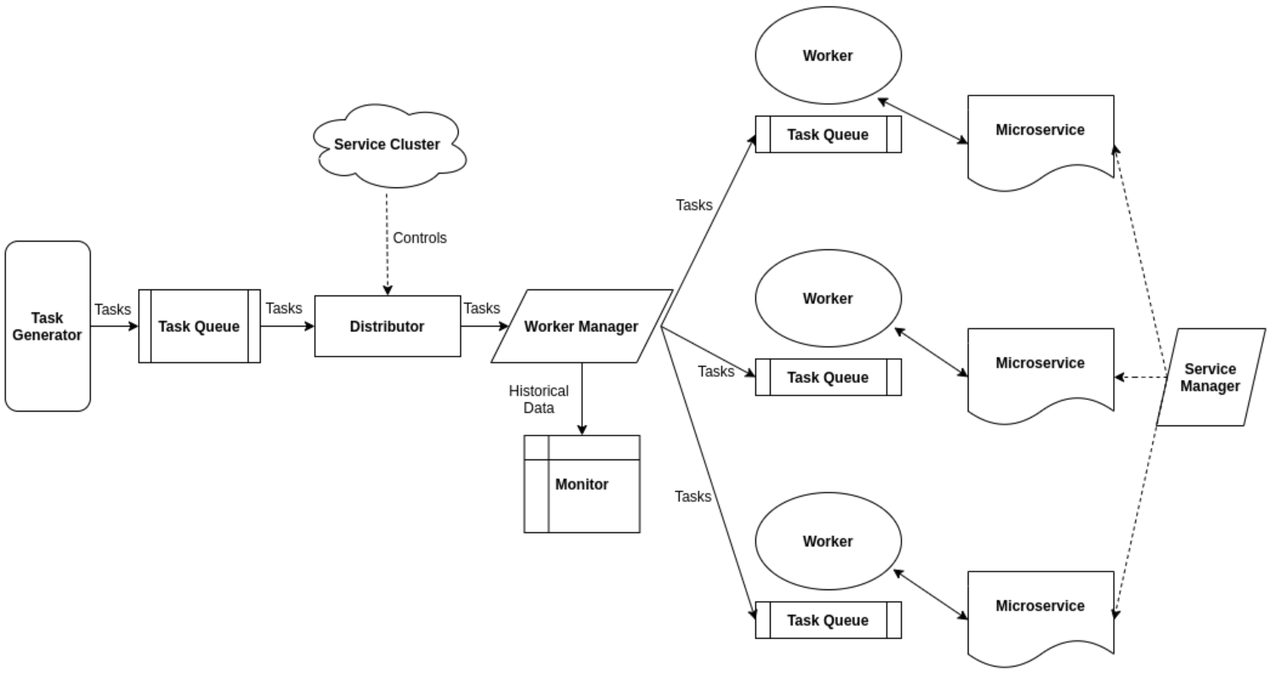

Figure 5: Proposed Architecture

4.4 Worker Manager

The Worker Manager handles the creation of worker threads, they have complete control over multiple workers. The worker manager can create the workers, delete the workers, and can assign tasks to different workers.

4.5 Worker

A worker is responsible for managing the connection with the single micro-service in multi-direction. A worker collects the task sent by the distributor and stores them inside its individual task queue and later dispatches the task to its associated micro-service and receives the response time. Each worker also maintains the history of response time in a task queue, which is used by the clustering algorithm.

4.6 Microservices

Each microservice in the system will run on a host and interact with their workers through a port. All the tasks from worker queue will be processed by microservice and when the microservice successfully executes the task, it returns a Success task id to its associated worker. Each microservice has its dependency on computing capability to execute the tasks.

4.7 Service Manager

The service manager is mainly used to manage the operations for microservices. Service manager can create or delete the microservices as well as they can also start or stop the microservices.

4.8 Monitor

This is the last component of the proposed system, which will be used to provide the graphical visualization of the system. The Monitor class collects the data from all the workers and plot them on the Graph.

5 Implementation

The proposed system has been completely implemented in python. The concept of threads has been used to generate the tasks with randomization function. To simulate the internal components of the system, low-level python libraries have been used such as sklearn to apply k-means clustering algorithm, pandas for data manipulations, matplotlib to provide graphical representation. All the execution can be performed on the localhost, which helps us to reduce the complexity of the system. The microservice will use TCP/IP to transfer data as raw bytes. I also have implemented the Round robin algorithm and Equally Spread Current Execution Algorithm to have a comparative analysis of performance between both the methods. The specification of the machine to run the developed system is as follow :

- Operating System : Windows

- CPU : i5 (4 Cores)
- RAM : 4 GB
- Language : Python
- Libraries : pandas, matplotlib, sklearn

6 Results and Discussion

This section will have a comparative analysis between the traditional method (Round-robin algorithm) and the proposed method. We will discuss the results in more detail.

6.1 Experiment 1 / Round Robin Algorithm (Without Cluster)

The Roundrobin algorithm mainly works on time quantum so it assigns equal tasks to every microservice, irrespective of its response time. In this experiment, we have used three microservices to executes the tasks efficiently. Each of the microservice has different task execution capacity. The problem with the Roundrobin algorithm is it can provide more tasks to low performing instance/microservice. This method can execute all the tasks but tasks can take more time to execute. Hence result reflects an increase in response time. The graph for Roundrobin algorithm is shown in Figure 6

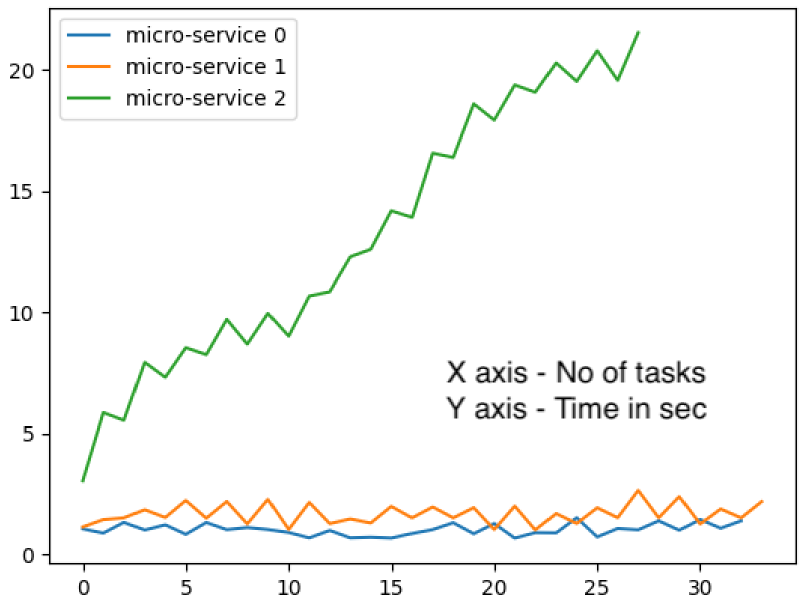

Figure 6: Consecutive tasks Vs Response time of a task using Round Robin algorithm

In the above graph, the X-axis represents the Number of Consecutive tasks. Whereas Y-axis represents response time for their corresponding tasks. As we can see, as response time for Microservice one (Green Line) is very high.

6.2 Experiment 2 / Equally Spread Current Execution algorithm (With Cluster)

To assign the tasks intelligently concerning response time we are using equally spread current execution algorithm with the k-means clustering algorithm. This algorithm will reduce the number of tasks for less performing instance/microservice and may increase the tasks for high performing instance/microservice, overall it establishes a good balance to reduce the response time. This method uses the K-means clustering algorithm to generate the clusters concerning the response time of tasks. After executing the Equally Spread Current Execution Algorithm with k-means clustering, we achieved the result shown in Figure 7

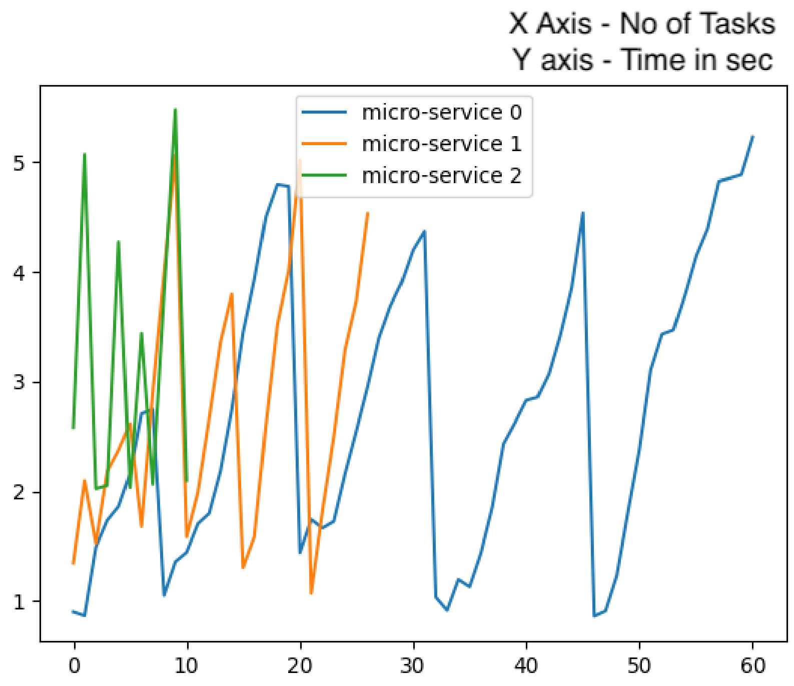

Figure 7: Consecutive tasks Vs Response time of a task using ESCE based clustering algorithm

The three lines Green, Orange, and Blue in the above graph represent the information about the response time of three different microservices. In the above Graph, the X-axis represents the Number of consecutive tasks. Whereas Y-axis represents response time for their corresponding tasks. Here we can check the response time of all microservices is very low as compared to the traditional (Round robin)algorithm. The reason for low response time is we are optimally assigning the task according to the processing capacity of microservice without knowing the specification of microservices.

6.3 Discussion

The experiment results clearly indicate that the response time of the Proposed algorithm is very less as compared to the round-robin algorithm(around 70% improvement in overall response time). The round-robin algorithm assigns equal tasks based on the time

quantum without considering the response time of a microservice. The microservice with low computation capacity will be always busy in executing more number of tasks. To overcome this, our proposed algorithm assigns the tasks based on the response time. The idea of proposed work is very simple, we are assigning more tasks to those microservices, which have the least response time. The minimum response time of a microservice is due to the two main reasons either the microservice processes the tasks faster, or there is not much network latency, or both. Minimizing the response time clearly indicates that tasks are executing at a faster rate. The comparative analysis between both the methods indicates that the response time of service has been reduced with the help of ESCE clustering-based mechanism. Additionally, we could also conclude that Proposed architecture can execute the tasks very efficiently. The proposed design serves both of these research project objectives in the following manner:-
1)The proposed design improved overall response time around 70%. The comparative analysis between the two results shows the response time was brought down to 6 seconds from 22 seconds.
2)The proposed design showed its effectiveness in allocating tasks as per the response of each microservice(in the results of a second experiment, microservice with lowest response time gets more tasks whereas the microservice with highest response time gets the lowest number of tasks).

7 Conclusion and Future Work

After experimenting and analyzing the results between two different methods, I can conclude that the proposed model can effectively minimize the processing delay by choosing the optimal microservice/instance for tasks. The proposed solution is simple, effective, and shows around 70% improvement in overall response time(brings down response time to 6 seconds from 22 seconds). This work also shows the importance of unsupervised learning (K-means algorithm) which enables us to take better decisions to reduce the response time and increase the system performance. The proposed model can replace the existing traditional methods and can be used at production level systems or in the cloud computing environment. In the future, we can even predict the response time of the task with the help of machine learning and deep learning techniques. We can add more features such as network traffic, latency, etc to having more detailed analysis. Similar approaches can be applied for various applications such as CPU scheduling, network management, and many others.

References

Alamin, M. A., Elbashir, M. K. and Osman, A. A. (2017). A load balancing algorithm to enhance the response time in cloud computing.

Alshuqayran, N., Ali, N. and Evans, R. (2016). A systematic mapping study in microservice architecture, *2016 IEEE 9th International Conference on Service-Oriented Computing and Applications (SOCA)*, IEEE, pp. 44–51.

Balalaie, A., Heydarnoori, A., Jamshidi, P., Tamburri, D. A. and Lynn, T. (2018). Microservices migration patterns, *Software: Practice and Experience* **48**(11): 2019–2042.

Bravetti, M., Giallorenzo, S., Mauro, J., Talevi, I. and Zavattaro, G. (2019). Optimal and automated deployment for microservices, *International Conference on Fundamental Approaches to Software Engineering*, Springer, pp. 351–368.

Duc, T. L., Leiva, R. G., Casari, P. and Östberg, P.-O. (2019). Machine learning methods for reliable resource provisioning in edge-cloud computing: A survey, *ACM Computing Surveys (CSUR)* **52**(5): 1–39.

Fazio, M., Celesti, A., Ranjan, R., Liu, C., Chen, L. and Villari, M. (2016). Open issues in scheduling microservices in the cloud, *IEEE Cloud Computing* **3**(5): 81–88.

Filip, I.-D., Pop, F., Serbanescu, C. and Choi, C. (2018). Microservices scheduling model over heterogeneous cloud-edge environments as support for iot applications, *IEEE Internet of Things Journal* **5**(4): 2672–2681.

Gan, Y., Zhang, Y., Hu, K., Cheng, D., He, Y., Pancholi, M. and Delimitrou, C. (2019). Leveraging deep learning to improve the performance predictability of cloud microservices, *arXiv preprint arXiv:1905.00968* .

Khatavkar, B. and Boopathy, P. (2017). Efficient wmaxmin static algorithm for load balancing in cloud computation, *2017 Innovations in Power and Advanced Computing Technologies (i-PACT)*, IEEE, pp. 1–6.

Lloyd, W., Ramesh, S., Chinthalapati, S., Ly, L. and Pallickara, S. (2018). Serverless computing: An investigation of factors influencing microservice performance, *2018 IEEE International Conference on Cloud Engineering (IC2E)*, IEEE, pp. 159–169.

Mesbahi, M. R., Hashemi, M. and Rahmani, A. M. (2016). Performance evaluation and analysis of load balancing algorithms in cloud computing environments, *2016 Second International Conference on Web Research (ICWR)*, IEEE, pp. 145–151.

Niu, Y., Liu, F. and Li, Z. (2018). Load balancing across microservices, *IEEE INFOCOM 2018-IEEE Conference on Computer Communications*, IEEE, pp. 198–206.

Parida, S. and Panchal, B. (2018). An efficient dynamic load balancing algorithm using machine learning technique in cloud environment, *International journal of scientific research in science, engineering and technology* **4**: 1184–1186.

Rusek, M., Dwornicki, G. and Orłowski, A. (2016). A decentralized system for load balancing of containerized microservices in the cloud, *International Conference on Systems Science*, Springer, pp. 142–152.

Rusek, M. and Landmesser, J. (2018). Time complexity of an distributed algorithm for load balancing of microservice-oriented applications in the cloud, *ITM Web of Conferences*, Vol. 21, EDP Sciences, p. 00018.

Sajjan, R. and Yashwantrao, B. R. (2017). Load balancing and its algorithms in cloud computing: A survey, *International Journal of Computer Sciences and Engineering* **5**(1): 95–100.

Yi, C., Zhang, X. and Cao, W. (2018). Dynamic weight based load balancing for microservice cluster, *Proceedings of the 2nd International Conference on Computer Science and Application Engineering*, pp. 1–7.

Yu, R., Kilari, V. T., Xue, G. and Yang, D. (2019). Load balancing for interdependent iot microservices, *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*, IEEE, pp. 298–306.