National College of Ireland

Eliminating the downtime faced by the IaaS hosted web applications during vertical scaling

MSc Research Project
MSc Cloud Computing

Tanya Chopra
Student ID: x18177271

School of Computing
National College of Ireland

Supervisor: Vikas Sahni

National College of Ireland
Project Submission Sheet
School of Computing

National College of Ireland

Student Name:	Tanya Chopra
Student ID:	x18177271
Programme:	MSc Cloud Computing
Year:	2020
Module:	MSc Research Project
Supervisor:	Vikas Sahni
Submission Due Date:	17/08/2020
Project Title:	Eliminating the downtime faced by the IaaS hosted web applications during vertical scaling
Word Count:	9548
Page Count:	20

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.

I agree to an electronic copy of my thesis being made publicly available on TRAP the National College of Ireland's Institutional Repository for consultation.

Signature:	
Date:	14th August 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.	☐

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):	

Eliminating the downtime faced by the IaaS hosted web applications during vertical scaling

Tanya Chopra
x18177271

Abstract

In the computer world today, cloud computing is becoming a major division that is evolving as a result of technological advancement. The scope of this technology is also very broad. Even though Platform as a Service (PaaS) solutions are known by a lot of people than IaaS, there is a lot of scope for personalized solutions with Infrastructure as a Service (IaaS). The dynamic workload of any web application becomes a problem for the end-users. As it is not predictable, the IaaS availability is affected. One way to overcome this drawback is by using vertical scaling of the VMs. While the VM is being up-scaled or down-scaled, the waiting time is avoided for the end-users. The most crucial part of the solution to this problem is to cut down the downtime with this method. This auto-scaling happens automatically, depending on the workload faced when a web application is hosted in the IaaS. Creating a VM's image of a similar size is done by the 'MakeShift Cross Scale Algorithm'. This algorithm is used to track the workloads as it changes. This process is done without actually changing the existing VM. Cloning the existing VM into the new machine helps in scaling up the existing VM, according to the user requests. Once the process of cloning is completed, the old VM is decommissioned. Thus, this project eliminates any downtime that occurs during the whole process. With the help of vertical scaling, the performance of the system is improved through high availability in this research project. This approach of vertical scaling of the applications has resulted in high throughput. The average cloning time of the VM is approximately 8.8 minutes and the average deletion time of the old VM and its resources is approximately 4.5 minutes.

1 Introduction

Cloud Computing is an advanced technology, where many Information technology service providers use this platform to access readily available data. Maintenance of the servers is not done by the users in the environment that they use. Hosting applications happen in cloud platforms and the users do not have to worry about the performance or storage. These factors are taken care of, by the application providers. The users only manage the data and the code in web applications.

Control over the installed applications remains with the user. They also have control over the security, data, duration of a process, and the intermediate layer. The workloads keep changing for the applications and they are not predictable. Distribution of resources for applications that require higher performance and availability becomes a problem in such instances. The users scale up or scale down the machine when the workload differs.

Scalability includes elasticity and high availability of the environment, which is the real benefit while using IaaS (Al-Said Ahmad and Andras (2019)).

There are two ways such as horizontal scaling and vertical scaling, with the help of which differing workloads are managed. In this method of horizontal scaling, the existing server is divided into new instances, which helps in handling the load by spreading it out. This technique has a gap in which a specific process wherein spreading the workload across the VMs cannot be done and it needs more computing power. Computing power and the physical resources of a machine are increased with the help of vertical scaling (Shelar et al. (2016)). This technique usually takes less computation time to handle a high demanding single-threaded request. Downtime occurs only in vertical scaling whereas horizontal scaling does not have that problem. When web applications are used in any cloud environment, the major problem with vertical scaling is the downtime faced. Thus, there is a restart depending on the application's availability. This happens for various reasons like switch over between the existing VM and the new VM that is requested with differing configurations.

The high availability of applications is the main purpose of this project. This is achieved by auto-scaling techniques, depending upon the requirement of the user. While eliminating the downtime, the process also focuses on reducing maintenance requirements. A simple explanation for this process is: with the change in the workload faced by the CPU, a new instance of the machine will be created while the applications keep running in the existing VM (Feng et al. (2019)).

Microsoft Azure API is invoked by the 'MakeShift Cross Scale Algorithm' which rectifies the issue. When the required configuration is created, a new VM is cloned with a greater capacity. The old VM is still in use until the new VM is completely functional. After this process, the old VM is decommissioned. As a result, the application is made fully available even during the process of vertical scaling without any downtime (Lorido-Botran et al. (2014).

The main focus of the auto-scaling is to provide uptime with a good service quality while handling the disruptions. This research is focusing on the downtime of IaaS hosted web application and eliminates it with the help of vertical scaling. This research project holds great importance as it focuses on the experience of the end-user.

1.1 Problem Addressed

Based on the research question "How to eliminate the downtime of web applications hosted in IaaS during vertical scaling in Azure Cloud platform?", the following research project is rectifying the issue. The major problem arises with a dynamic workload is solved by having a system that is ready to perform according to the requests received. When the workload is high, the throughput will be affected in the form of downtime. This is eliminated as the users will not wait for their application to run and they have instant results despite the workload changes faced by the VM. The uptime, high availability, and quality of service are achieved by vertical auto-scaling.

1.2 Objectives and Motivation behind the Project

The main objective of this research project is eliminating the downtime that affects the throughput of any system. Using a horizontal scaling is not applicable for all instances and this process is also not cost-effective as there is a need to have extra resources at all

times, regardless of the need for it. Whereas, vertical scaling is cost-effective while there are problems with this method. One such major problem here is the downtime that is faced by the end-user. It is because of this problem; people spend a lot of money and choose horizontal scaling rather than going with a cost-effective solution.

The goal of every service provider is to satisfy their customer needs. Any problem that results in unhappy customers is addressed immediately. This is the major motivation for this project for having satisfied end-users with the optimal throughput of any system. When the efficiency of a system goes down, it becomes time-consuming and rather unpleasant. So, this project works in eliminating this downtime. The usage of resources is efficient at all times. This is achieved by carefully developing a system that accommodates the changes in the workload efficiently. Vertical auto-scaling provides the best results that are related to throughput and efficiency.

1.3 Structure of the Report

This research paper consists of the following segments which give a brief explanation of each segment. It includes Related Work in Segment 2, Methodology in Segment 3 which explains the techniques and equipments used in the project, Design Specification of this project is given in Segment 4, Implementation is placed in Segment 5, Evaluation with the results are given in Segment 6 along with Discussion and Conclusion and Future work in Segment 6.5 and 7 respectively.

2 Related Work

The purpose of this research is to eliminate the downtime of the applications during vertical scaling in IaaS hosted environment because of the instant workload changes. There have been few similar research done in the past and gaps are identified from those research papers. In this research, past approaches related to vertical scaling are discussed to improve the high availability of the applications along with shortcomings.

In the previous studies, there were some research papers related to auto-scaling during different kinds of workloads. In general, Cloud service providers manage the auto-scaling in the case of PaaS based web hosting. But in the case of IaaS based web-hosting, the Virtual Machines configurations are changed depending upon the workload, which was done programmatically. In the case of horizontal scaling, the downtime does not occur because VMs are added or deleted depending upon the workload changes. In vertical scaling, the configurations of the VMs are changed depending upon the workload changes, which requires a restart process to make the changes effective and this results in downtime of the application. In the MakeShift Cross Scale Algorithm, continuous monitoring is performed for the workload. Once the threshold is achieved, an alert is triggered and vertical auto-scaling is performed along with cloning of the VM. Upon cloning of the VM, resources are allotted appropriately.

2.1 IaaS based web server : Vertical scaling

According to (Ali-Eldin et al. (2012)), the scaling up was quick within SLA constraints whereas scaling down only happened when there was not much workload with the resources present at that point of time. The scaling up and scaling down processes were controlled using adaptive horizontal controllers for elasticity. The design of the controllers

was very efficient which managed flash crowds within defined timelines. A well-defined event simulator was used to check on the performance of the controllers. An elasticity engine was built using nine separate approaches in the first situation. The regression-based controllers were used for comparison purposes in the second situation. As a result of workload changes, the performance was evaluated in the third situation.

As a result of all these scenarios, it is concluded that the resources are not made free immediately by the controller during the scale down operation in the case of a decrease in the workload. It made sure that the changes are not immediate due to proactive scaling. In the case of reactive scaling, the controller immediately makes the resources free in the case of a decrease in the workload. The final outcome of the research suggested that for scaling up, a reactive scale is used whereas for scaling down, a proactive scale is used.

The research has a clear gap wherein the amount of SLA violation increased and the research majorly looked into horizontal elasticity controller than vertical elasticity controller. So the algorithm "MakeShift Cross Scale" uses vertical scaling with the clear focus of downtime being eliminated when the scale-up or scale-down process happens.

In (Hwang et al. (2014)), it is mentioned that better performance was achieved with fewer clusters in scale-up operations but it lowered the performance when the switch over happened with higher or lower configuration. The gap in this paper indicated the importance of the current research topic. In the paper (Jayasinghe et al. (2014)), an analysis was conducted using six IaaS platforms, which included private and public clouds. The analysis mainly focused on scalability and performance aspect and for that reason, similar configurations were maintained for all six platforms. The results were varied between optimal results to the worst results in terms of performance. It gave a direction to make the algorithm suitable for all platforms to be consistent.

2.2 Autoscaling and service model of IaaS

According to Casalicchio and Silvestri (2011)), a set of techniques was defined, which were considered for the designing and execution of auto-scaling. It also mentioned that Cloud Providers implement the auto-scaling themselves, i.e., the Application Service Provider (ASP) took care of the implementation where the services were leased from the IaaS provider. A unique way was defined in this paper, which considered few requirements functionally that were clubbed together as architectural components under auto-scaling. It considered load balancer, monitoring, resource manager, planner, and analyzer as components. The outcome in this paper suggested that these services were provided by cloud service providers, there were gaps found in these services and they were automatically provided for management of the plan, monitor, and analysis.

In (Hasan et al. (2012)), auto-scaling techniques were described wherein independent metrics were considered for every step of scaling. They were aligned with all other metrics. Three different heuristics were taken as three different parts of their proposed algorithm. The first part was the cloud thresholding techniques; the second one was the multi-domain metric and the third one was the resource integration. Few of the metrics identified; CPU load, storage, response, network etc. But the scope was limited as there was no clear definition in the scaling type and the resources were very limited. So, the algorithm "MakeShift Cross Scale" clearly works on vertical scaling for effectively and efficiently allotting the resources according to the workload without any downtime.

2.3 IaaS platforms : Elasticity and Monitoring

IaaS, SaaS, PaaS, and hybrid clouds are various models that were evaluated in (Hwang et al. (2016)) for their performances. Various benchmarks that were used to check the scaling techniques of various cloud providers were used to check the performance. There were three different levels for performance metrics. The first level had utilization, speed, efficiency, etc. The next level had network latency, data analytics, data throughput, etc. which were covering the potential of cloud platforms. The last level had cost, SLA, security, power, QoS, etc. which were covering the yield of the cloud platform.

The study oversaw EC2 and AWS and mainly focused on three approaches namely vertical scaling, horizontal scaling, and mixed scaling (upscale and downscale in parallel). There were good results in the mixed scaling. In the heterogeneous systems, it performed better during unpredicted workload changes whereas in vertical scaling, the test was done using 800 plus users with 10 GB data and the results were not encouraging. The key inconsistency was, none of the single requests required more resources and as a result, the analysis was not appropriate. As a result, the algorithm "MakeShift Cross Scale" takes extensive load requests as well as a single request which demands more resources.

According to (Han et al. (2012), a lightweight approach was proposed for cost reduction, where the cloud provider in the IaaS platform specified the resources consumed and operating cost as the running cost of a single server. The Cloud platform had an e-commerce data center for testing, where each server had a separate virtual machine dedicated to its performance. Based on the browsing activity of the customers, such as product ranking, searching, etc, browsing workload was calculated. Other actions such as login, placing an order in the e-commerce site which required heavy database queries were categorized as ordering workload. For this particular lightweight approach, another platform that was used as a middleware was required. Emulators were used to generating the requests that were sequence-based on the TPC benchmark.

The interactions from the emulators occurred with a random time interval and the performance was monitored using monitoring service. User-specified the response time requirement, which was used to notify the users when the system scaled up/down. The incoming requests were monitored by the monitoring service and the resource utilization in each server was monitored by the monitors. There were two major gaps in the research, which included the reserving of the services approach that resulted only in scaling at the resource-level. The other gap was, this approach had only one cloud provider, whereas the real-time scenario had applications that were complex.

In (Herbst et al. (2015) based upon few metrics, a new technique was suggested related to elasticity in IaaS based cloud systems with three major parts in the experimental setup which included infrastructure nodes, benchmark controller nodes, and load balancing and management nodes. CloudStack (CS) and AWS were used to configure a defined set of parameters for rule-based elasticity. The major metrics proposed were accuracy and timing. Few presumptions were made on few platforms for scaling the virtual resources, brackets of scaling for resources, etc. The benchmarks were provided using the demand curve and the supply curve required by the metrics. Finally, the resources were commissioned or de-commissioned depending upon timing and accuracy.

There were four processes namely Platform analysis, Measurement evaluation, Benchmark calibration, and Elasticity evaluation, which were included in Bungee elasticity benchmarks and also they were executed on the private cloud, cloud-stack platform, and public cloud AWS platform. These benchmarks were good but focused on horizontal

scaling and not vertical scaling and primarily focused on reactive scaling, which was not required for many cloud requirements. As a result, a new technique is provided, which monitors and subsequently depending upon various parameters, assigns, and re-allocate the virtual machines accordingly.

Monitoring is an important procedure for cloud service providers. Elasticity, Migration, and Scalability are major cloud components that require monitoring. Based upon workload variations, scalability helps in performance efficiencies by scaling up or scaling down the resources as per requirements. To make Scalability more efficient, monitoring systems were executed with more investigations. The purpose of Elasticity was to scale up or scale down the resources as per varying user demands. The monitoring system helped in Elasticity as it traced the availability of resources during and after scaling procedures and helped it manage resource availability. Another component was Migration, which migrated the resources depending upon application requirements and the monitoring system made sure that the data was not lost during migration. In IaaS, few monitoring issues were identified like integrated monitoring and energy efficiency as per (Rodrigues et al. (2016)).

The paper (Singh et al. (2019)) discussed MAPE which was used in all auto-scaling techniques. Through Monitoring, it collected information related to response mechanism, CPU utilization, SLA deviation, etc. As the next step in Analysis, resource utilization and the estimated workload were identified depending on the threshold level, which helped in commissioning or de-commissioning of virtual resources. The next step of planning helped in auto-scaling because of outcome from the analysis phase and this resulted in execution using Cloud Service Providers APIs. This paper was useful in the decision of using efficient monitoring kits and optimum technique of resource utilization using vertical scaling.

In the paper (Nikravesh et al. (2015)), on the basis of the time-series prediction algorithm, the prediction method was used in auto-scaling. The gaps in pro-active and reactive methods were discussed for already developed methods. Under the IaaS cloud, a gap was identified related to reactive algorithm wherein boot-up time for VM was ignored, and due to that under-provisioning of resources happened, which resulted in the SLA penalty. On the other hand, due to a lack of support for unpredicted workloads in the proactive algorithm, predictive auto-scaling was highly recommended. The resource allocation was optimized using the predictive auto-scaling.

There were three kinds of workload patterns concentrated by the author in (Nikravesh et al. (2015)—) such as unpredicted workload, periodic, and growing. The algorithm used Amazon EC2. The purpose of this research was to check various kinds of workloads with perfection in predicting by using NN and SVM wherein NN is Neural Networks and SVM is Support Vector Machines, which were required to look into learning techniques. Various metrics like Root Mean Square Error, R2 prediction accuracy, Mean Absolute Percentage Error, and PRED were suggested for algorithm evaluation. The key inconsistency in this paper was that, which prediction technique was to be used with perfection. The final outcome of the research suggested that SVM was better for periodic and growing patterns of workload whereas NN was better for workloads, which were unpredictable. It was also suggested that one prediction method cannot be used in all cases.

Various metrics were described in (Qu et al. (2018)) which were useful as performance indicators used in the monitoring. Hybrid metrics utilize both low level (uses server details) as well as high-level metrics (at the application level generally used by auto-scalers). Resources were estimated using a few methods. The most common method

was based upon setting up the parameters for triggering scaling operations on a certain threshold. The second method was application profiling wherein the threshold was tested during the execution of application but the gap identified here was, manual intervention in setting the parameters. In the last method, a manual setting of rules and parameters were done. The machine learning technique had a major gap wherein the time consumption was found to be huge, and this impacted the performance of the scaling operation. It was discussed in this paper that vertical scaling had advantages in comparison to horizontal scaling because there were few services that were not copied and executed during runtime, but they continued in the same state during scaling vertically.

To overcome monitoring issues that were discussed in previous paragraphs, it was focused on monitoring the changes in workloads and provide scalability solutions accordingly.

According to (Bauer et al. (2019)), the Chameleon method was described which had two components reactive and proactive controller. The reactive controller continuously monitored, and based upon stored data, took the action for scaling. The proactive controller performed on the basis of forecast data estimated from the historical data and did the scaling process accordingly. Elasticity, which is a system-oriented metric, and metrics like the average amount of VMs, median and average response time, which were grouped as user-oriented metrics were taken into consideration. The outcomes were very much configuration dependent. This was not used for the applications, which were CPU intensive. The 'Makeshift Cross Scale Algorithm' is inspecting continuously the different loads on the VM, and the resources are allocated appropriately.

2.4 Vertical Scaling and Resource Management

In (Ahmad et al. (2017)), many testing methods on empirical basis were evaluated but without any statistical testing scope. They were used in partial scope without checking real-time applications and varied situations. In the current research, these aspects are very well considered and a sample application is built that increases or decreases the load to check vertical scaling on few real-time situations. These scenarios underlie a path for testing the application's performance in cloud platforms.

The experiment discussed in the paper (Sotiriadis et al. (2019)), considered the VMs as the main focus, and their performance was analyzed based on two infrastructures, which included OpenStack and VMWare VCloud platform. The VM deployment was based on default sizes, which were duplicated in OpenStack and VMWare. Two different platforms were used in order to demonstrate the inter-cloud notion. Six experiments were conducted, and load balancing solutions were analyzed. Using REST APIs, both horizontal and vertical scaling were performed and for real-world scenario demonstration, Cassandra was used, which is an open-source search engine.

In (Sotiriadis et al. (2019)), there was an optimization scheme, which discussed an issue during the VM resizing or migration experienced in this particular analysis. From the results of the analysis, it was found that the scale-out/in the process did not work well when compared to the scale-up/down process in terms of handling the number of requests. Based on this, there was a clear insight that vertical scaling's efficiency was more while handling the requests. Thus, in this research, vertical scaling is relevant. It works for resource management in cloud services.

3 Methodology

Agile methodology is chosen for the implementation of this particular project. Agile methodology helps in eliminating the problems and overcoming the gaps by implementing this project phase by phase. For a detailed explanation, please refer to the Config Manual.

As per (Podolskiy et al. (2018)), depending upon the varied workloads, the automatic management of virtual resources was done, which is known as auto-scaling. The research was focused on the changes in the configuration of the virtual resources or defining the situations when the scaling was required, based upon varied requirements. On the basis of this, a new design was developed for auto-scaling wherein the main focus was to remove the downtime in vertical auto-scaling.

In a Cloud platform, the allocation of resources plays a major role in performance. When the number of requests increases or decreases, the workload varies suddenly. Horizontal scaling creates VM with the same configuration, and the workloads are distributed across VMs. When a single workload costing many resources cannot be distributed across VMs, in such cases, vertical scaling is done, where the memory and other configurations of a VM are increased to accommodate the new workload. The main disadvantage in vertical scaling is the downtime during the resource allocation. This is one prevailing issue in cloud architecture and most cases, PaaS and IaaS are used. In IaaS, the resource allocations are not automatically managed and hence, the combination of manual management of resources is targeting the downtime in vertical scaling.

In the research paper (Podolskiy et al. (2018)), three different cloud service providers (AWS, GCE, and Azure) were compared with the same type of workloads to understand the performance of each platform. Workloads were simulated based on various types like linear increase, linear increase, and constant, random, and triangle, and the simulation was for 20 minutes and the request timeout was set to 6.5 seconds. The VM configurations were same for the three service providers and the performance was monitored. Based on the results, it was found that Azure showed slowest scaling behavior among the three, which further enhanced the research idea's challenge. So, by modifying the scaling methodology, the idea is to enhance the scaling performance of Azure which is lagging.

The algorithm "MakeShift Cross Scale" is used in eliminating the downtime in the applications with the help of vertical scaling. This is an auto-scaling process that detects the workload automatically and changes the resources of the VMs accordingly. With the rapid change of workloads, various alerts are triggered and the VMs along with the resources are cloned for better throughput and performance. This makes the application user-friendly and very efficient.

The auto-scaling process used in this project ensures that the applications are running smoothly. This is experienced by the end-user directly, so they do not have any knowledge about the VM being scaled up or down in the background. This process is increasing the high availability of the VM in use resulting in better response time without any disruptions.

3.1 Techniques used in the Project

The techniques used are scaling up and scaling down the VM. This depends on the workload faced by the VM, with all the requests received. When the number of requests received is low and the CPU resources are not currently used i.e., the CPU resources are used below 20%, this triggers the process of scaling down, which happens automatically.

It avoids the use of resources that are not necessary for the requests in hand.

Scaling up is a process where new resources are required for the VM due to the large number of requests received. So, once the CPU usage goes beyond 80%, an alert is triggered and a new VM is created by cloning the old VM so that it can accommodate the workflow, that is experienced by the VM. The old VM along with its resources, such as network interface and disk are deleted to avoid any unwanted usage of the resources.

3.2 Equipment used in the Project

• Azure VM is the system that is used by the end-user.

• Azure monitoring metric alerts has all the conditions. This helps in monitoring all the processes that are happening under the VM.

• Traffic manager which is known as the load balancer in the backend logic. This has a unique URL. This URL is the place where the user views the application.

• This URL points to the current IP in use.

• The endpoint has the IP of the cloned VM.

• App services are the platform that hosts the algorithm.

• Image of the VM which has all the running applications, and other information of the VM, that is initially in use. This is the most important equipment as this is used for cloning the new VMs. This is similar to a copy of a VM.

4 Design Specification

When it comes to the workload of applications which is experienced by the end-users, dynamic changes are something that can never be avoided. During such situations, allocating the resources are done properly, so that the resources are used effectively. Allocating the right resources is the main strategy in using any machine to its full potential. When the workload of a particular application is increased, the instances of the system increase the resources. Managing the resources based on the workload is done by a process called scaling. Scaling can be differentiated as scaling up or scaling down. This shows if the resources are allocated to a higher degree or if the allocation of the resources is cut short according to the need.

When the changes in the workload are unpredictable in a web application, the workload cannot be checked at all times to allocate the right resources. Here, the process should be done automatically. This is done with the help of auto-scaling. In this method, managing resource allocation is done dynamically, based on the workload. This project uses the 'Makeshift Cross-Scale Algorithm', which constantly checks the workload being faced, and whenever a threshold is met, vertical scaling happens automatically. Once vertical scaling is initiated, cloning of the VM occurs subsequently and the resources are allocated accordingly.

4.1 Architecture Design : Scaling Alerts

The following is the architecture of the Azure platform. This helps in explaining the distribution of the objects used in the project. This architecture also explains how each object is being deployed in the cloud environment. Whenever a process is built on any platform, it consists of different objects to perform different functions. This is a common

feature of any process in any platform. The architectural design of this particular project is explained in detail as follows with an example:

A user is operating in the cloud environment and invokes a new request. This new request is then sent to the webserver. The request invoked alone requires 100% CPU for the process to run for 10 minutes. In this scenario, the Azure monitor is monitoring the VM. VM is being cloned where an alert is created and triggered. This alert is sent to the algorithm indicating that the CPU has reached 80% of its operational efficiency. After this step, the algorithm that is hosted in the form of a PaaS solution (App Services), is placed in the Azure data center. This algorithm is now fetched and is working in a particular way according to the request, which is received for scale-up or scale-down.

After this process is done, the current configuration of the CPU is checked. Once this step is completed, the next task of the algorithm is to fetch the next configuration, which is higher/lower from the configuration file. This is calculated according to the current demand for CPU. Once the configuration is fetched, the appropriate ARM API is called in a sequence, where it provisions or de-provisions the VM according to the CPU percentage in the Azure platform. All these processes happen in accordance with the algorithm.

In addition to this process, the Azure monitor continuously checks the requests that are received and communicate the right message to the end-users at all times. Throughout this process, the VM is placed under a load balancer which is the Azure Traffic Manager, which helps the algorithm to switch according to the requests received from the end-users. This in turn makes sure that there are no requests drops whenever there is a switch between the VMs. The VMs mentioned here are the present VM that is being used and the new VM which is being created. Once the new VM is cloned and created, the old VM and the resources associated with it are decommissioned.

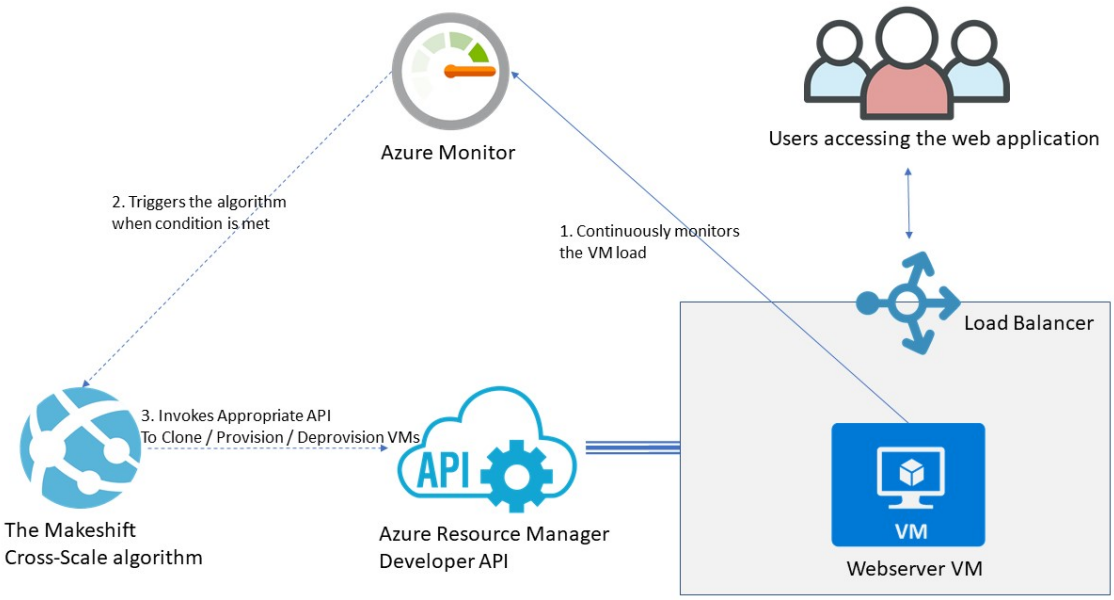

Figure 1: Architecture Design

4.2 Make Shift Cross Scale Algorithm

MakeShift Cross Scale Algorithm is implemented in this project. It detects the workload according to the new requests and triggers the alerts based on it. In order to scale up the machine, the old VM is cloned. Once the new VM is created, the old VM is decommissioned along with its resources.

Step 1 : In order to create a new VM, it is necessary to first create a network interface. This is a service that is necessary to start the VM every single time. The virtual network and network services are required to run the VM.

Step 2 : Once this network interface is created, the new cloned VM is created and it waits until the cloned VM starts working. This is done to check if all the necessary applications are loaded into the cloned VM and also if the VM is working. Only if the VM is working, the IP address of the VM is updated in the traffic manager. This process diverts the users and the requests to the cloned VM.

Step 3 : Upon the completion of all the above steps, the old VM and the resources are allocated to that particular VM are deleted. Some of such resources deleted are the old VM itself, its disk, and its network interface. The process gets completed after this stage and when a new alert is created the same process is executed again, for either scaling up/down the VM. The coding part describes the process that happens when the alert is created. Two static IPs are created manually. This is useful when the VM is cloned.

All the predefined information is stored in the coding part. Information like the subscription ID, resource group, virtual network, and network safety group that are related to the Azure subscription is available in the coding section. All the network information is set manually and then is configured for further use.

Instead of creating a new IP every time, the static IP is used for the cloned machine. Azure VM has different sizes. Only when different sizes are available, it is possible to scale up or scale down the VM. These different sizes are fixed and defined in the beginning and this acts as the master data. There is another file that has the properties of the current VM like its name, network interface, and the disk name along with its size and the IP address.

Creating alerts or the VM happens dynamically and that information is populated in the JSON file. Information stored such as VM being scaled up or down is needed for future reference. Once the new VM is cloned, the old VM and its resources are deleted to avoid unwanted usage of resources. When a new VM is cloned, the information of that particular VM is updated automatically to the JSON file. After this step, conditions to set up the parameters for scaling up and scaling down are given. A VM cannot be scaled down to a size lesser than the VM size (Basic A0) available to a user. This is the same for scaling up (Standard A1 v2).

A Boolean value ‘true’ is used in the next step to allow only one cloning process to happen at any given time. This is done to ensure that the cloning process is done, and completed without any conflicts even when new requests for scaling up or down are being received. Here, the JSON file is updated indicating that there is currently no-cloning happening and a new request is received. After this step is done, a unique name for the new VM that is cloned is allocated. The name is pre-defined along with a dynamic size for the VM. This is the current size of the VM that is in use plus 1 if the process is to scale up. The other pre-defined values that are required for creating new resources are the VM name, network interface, and the name for the new disk. This part also has the image of the VM in use. This is available in a file format and is used to create any

number of VMs. This is the most important part of having an image of the VM that allows new VMs to be cloned.

The next step in the coding part is to denote which endpoint is currently in use when a VM is scaled up/down. The conditions are set as 'true' for scale-up and 'false' for scale down. The size of the new VM is defined based on this value. This helps in fetching the right size for the new VM from the array of sizes that are predefined in the beginning. In the two IP addresses declared earlier, one IP is already in use by the existing VM. This information is available in the VM config along with all the properties of the current VM. All the above steps are used to provide the information that is necessary to create a new VM. The next step is authentication which connects the process to the Azure platform. Once the authentication is checked the next step is to delete the old alert. If the deletion is not done in this stage, numerous alerts are sent to the Azure portal consecutively. In order to avoid this, the old alert is always deleted once the request is received and the process of cloning is initiated.

The following steps are done when actual cloning takes place. After this, a network interface for the new VM is created. This name is also predefined and mentioned in the coding during the first few stages. For a VM to be created in an Azure portal, network interface, network security group and virtual network are essential. Since the virtual network and network security group are predefined, only the creation of a network interface takes place here. The virtual network and the network security group remains the same for any number of VMs that are cloned. Only the network interface gets changed. The new IP is given when the network interface is being created. This is the IP that is not allocated to the old VM at this stage. Here, an access token is passed along with the name of the network interface and the size of the VM, so that permission is received to use the MS portal.

This process of creating a new VM is asynchronous. Once the above API is called, the status changes to 'VM is being created'. The completion of the process is not known and changing the IP address before knowing if the VM is created, can cause problems in routing the new requests to the new VM that is being cloned. This is because the VM cannot be started. Here a while loop is used to check if the process of creating the VM is completed. This loop keeps running until the VM is created. This helps in knowing if the process of creating the VM is stopped in between because of some error. This process is continued until the new VM runs in the portal and the application in the old VM is running in the VM that is cloned.

The major goal of the above step is to know if the new VM is created. Once this is done, the new IP is updated in the traffic manager. After this, the applications are pointed to the new cloned VM. During this stage, both the VMs are running while the applications are being pointed to the new VM, and the web application has no downtime and is up and running at that point in time. The next step is to delete the properties of the old VM that are stored in the JSON file. This is also an asynchronous process. A while loop is used here to keep checking if the deletion is completed. This is important in order to delete the network interface and the other resources that are related to the old VM.

Once the old VM is deleted, the network interface and the VM disk are also deleted. Then the alerts to the new VM are created. There are two alerts to indicate that the CPU usage is below 20% and above 80%. Once all these steps are completed the Boolean value is set to 'false' informing that the VM has the capacity to take up the incoming requests. When the workload is increased or decreased beyond the alerts that are set,

the whole process is repeated. This completes the coding process of this project.

5 Implementation

According to the workload changes in the web application, a new request is triggered in the Azure platform. There is a change in the load on the CPU. This is because different applications and operations use the CPU differently. When there is a sudden increase in CPU usage, the speed of the VM goes down naturally. This is because the CPU allocates its resources for the completion of the application that are received. A problem does not arise until there is a huge change in the load given to the CPU. But, when the new request triggered, uses up a lot of CPU resources all of a sudden, the VM becomes slow. This is a problem as the end users experience a downtime in executing their applications. To avoid this, the machine is to cope up with the requests that are being sent. If the processes or requests just need only 20% of the CPU resources, the VM is scaled down and this happens automatically without any problem when the right alert is received. Scaling down is a rather easy process as the same system can just be used as it is.

When scaling up a VM is necessary, there are a few steps to be followed. Scaling up is done when a request needs 80% or more of the CPU resources for itself. During such instances, the new requests are routed to a different machine so that there is no lag in the performance of the system. This explains the need for scaling up a VM. Once a new request with a heavy workload is received, the alert for scaling up is sent out to the algorithm. There is an image that is a copy of the existing VM along with all the applications and the resources of that particular VM. Once the alert for scaling up is received by the algorithm, the next step is to use the image that is available already to create a new VM. This new VM has all the same applications and properties of the old VM with higher efficiency. This new VM also has a different static IP address. This IP address is updated in the traffic manager, once the process of scaling up is completed. After the process of scaling up is done as explained earlier, the new requests are sent to the VM that is cloned while all the other applications in the old VM are being transferred to the cloned one.

In the Azure portal, a person creates their own VM. This is also used to see various information like CPU percentage, network, request in, request out in the subscription. In this portal, there is a tab for size where a person can change the size of the VM according to the usage. The number of CPU, size of RAM, and Disc are different for different sizes of VM. Changing this size of the RAM according to VM's load using PaaS solutions (App Services) is part of the project. In Azure, there is a VM that has alerts that are configured in the portal itself. This alert is available in the load balancer. This is the place where the required alerts are created and depends on the load of the CPU. The alerts are also customized according to a specific need. Makeshift alerts can either denote scale up or scale down depending on how much load is assumed to be handled by the CPU without any downtime. Scope mentions the name of the VM and the condition has a clause with which the alert is created. Once these alerts are created they keep running in the background at all times in order to check the load of the CPU. There are usually two alerts for a VM where one can either upgrade or downgrade.

In this process, the CPU stability for scale-up stays for 1 minute, and scale down it stays for 15 minutes. In the vertical scaling method, the alert is triggered, when the CPU percentage goes up or down. When the system is upgraded or downgraded according

to the needs, the efficiency is increased along with balancing the load. This is necessary when there is more number of users or applications requiring more resources as the system has to be automatically upgraded. To avoid the downtime here, two alerts are used for the same system. When an IAAS service is purchased from a service provider, it is always better to have the right configuration of the system, that a person wants, so they do not have to pay unnecessarily.

In this project, a sample application is hosted inside the VM that is used, so that availability of the machine is known. This application is open at all times and is hosted inside the VM. The IP address of the VM is viewed. This helps in denoting the VM which is currently in use.

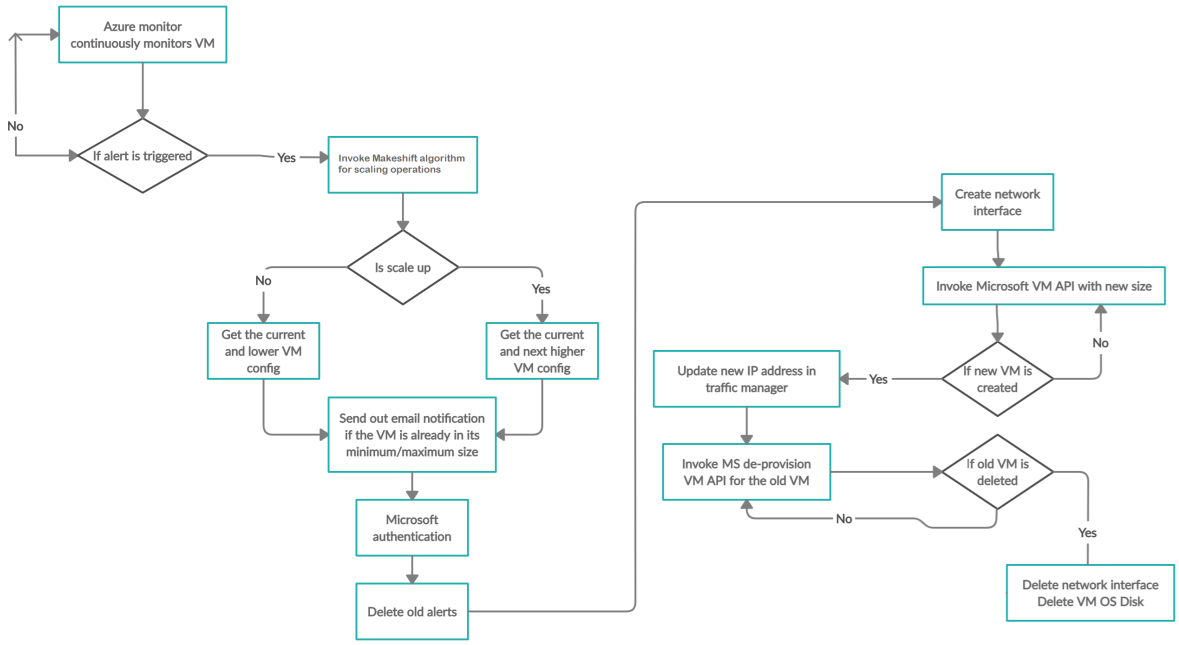

Figure 2: Workflow Diagram

There is also a load balancer named traffic manager which helps in managing the traffic that is routed to the new VM. The traffic manager has one endpoint. The endpoint is nothing but the IP address of the VM which is currently in use. The endpoint inside a traffic manager helps in routing the processes to the desired VM. This endpoint can also be an IP or a URL based on the user's need. Having a single endpoint calls for a change in the IP address when a new VM is being created by cloning the old VM. The URL of the traffic manager is default and received when a VM is created. This URL checks for the available endpoint when an application is being initiated and routes the process to the preferred VM.

When the URL that is given, automatically reaches the IP that is mentioned in the endpoint by the traffic manager. In this project, there is a logic similar to an API and that is hosted in the Azure platform. Based on the alerts that are set previously, if the CPU usage exceeds or goes below the mentioned level, it triggers an URL. This URL is generally known as an HTTP request or a webhook. When the URL is triggered, the algorithm or the back end logic is called. This is a major part of the whole project. There are two endpoints in the API depending on the alerts that are created.

The back end logic comes into play once the alert is triggered, depending on the

list of VM sizes available. There are two IPs by default which works for upgrading or downgrading the machine. An image is created at the beginning which is a copy of the VM. With the help of this image, any number of VM can be created. The image is kept ready along with two static IP addresses right from the start. The VM which is running automatically takes up an IP and there is another IP that becomes free at the moment. Scaling down and Scaling up are done depending on the size of the VM. Before each cloning process, the system checks for the next available size of the VM. The end-user is sent an email notification if the existing VM is already in the minimum or maximum configuration. This informs the user that the system cannot be scaled down or scaled up respectively, as it is already in its lowest or highest configuration.

The information that is required here is available in a form that is similar to an array and it is saved in a configuration file. This helps to know the configuration of the VM that is currently in use. It becomes easier to scale up or scale down once this configuration is known. Each VM size has a different array of elements and when a certain VM size is called, that particular array of elements are taken into account. In order to find the second element, a simple formula of VM size + 1 is used. This determines if the machine that is already in use has to be scaled-up or scaled-down. After finding this out, a network interface is created along with a new VM which is then being created with the help of the image of the old VM available. The new VM that is created has all the applications and the data that are available in the VM that is being scaled up or down. Once this process of cloning is done, the new VM has all the existing applications along with the new IP address created for it. Once this IP allocation is done the same gets automatically updated in the traffic manager. By doing this the existing VM and its applications are retained in the new VM which is scaled up/down.

Once this is done, the new application that is received by the traffic manager is automatically routed to the new VM. This enables the application to be running at all times thus eliminating downtime. Since the VM is being cloned, the application hosted on the user end keeps running in the new VM with the same configuration as before. Once the whole process is completed, the resources of the old VM are deleted since the old VM is not used at all after the new VM is hosted. This has only a single VM at the end of the day. In this whole process, only the alert is set manually.

6 Evaluation

The research project aimed at reducing the downtime that was faced because of varying workloads on a cloud platform. When horizontal scaling was used, there was no problem with the availability and throughput. This was because numerous resources were used if one machine was already operating to its full efficiency. But, in vertical scaling, there was only one machine and the scaling up or scaling down was needed to be done in that machine to accommodate the workload. This was the reason for the presence of downtime in the method. This problem was rectified by cloning a VM and configuring it into a machine with a higher-level or a lower level configuration. The cloning process used the image of the existing VM that had all the properties and applications of the existing VM with a different configuration. This was done by creating alerts that informed if the machine had reached its maximum efficiency or was being operated with a very little workload. Once the alert was created, it was used to call the algorithm that was used in the project.

The algorithm that was used here is the 'Makeshift Cross Scale Algorithm'. Once the algorithm was called, it decided if the VM had to be scaled-up or scaled-down. Depending upon the CPU usage, the VM was scaled down if the CPU percentage was less than 20%. Similarly, when the CPU reached 80% or more of its capacity, an alert for the cloning process was called and a new machine was cloned. This new machine was given the static IP address that was unused. Once the cloning process was completely over, the same IP address was updated in the traffic manager. The old VM along with its resources was deleted once the cloning process was over.

6.1 Test Case 1: Scale-up process

This process happened when the CPU reached more than 80% of its efficiency. After a minute of operating at its full efficiency, a scale-up alert was triggered, and the cloning process started happening in the background. In the below-given image, the CPU reached its maximum efficiency of 87.85% at 01:39 pm, and the scaling up alert was triggered. This began the cloning process of the existing VM. The process of cloning took 9 minutes to complete and the new VM was created. Once the new VM was created, the old VM along with its resources were deleted and this deletion process took 4 minutes.

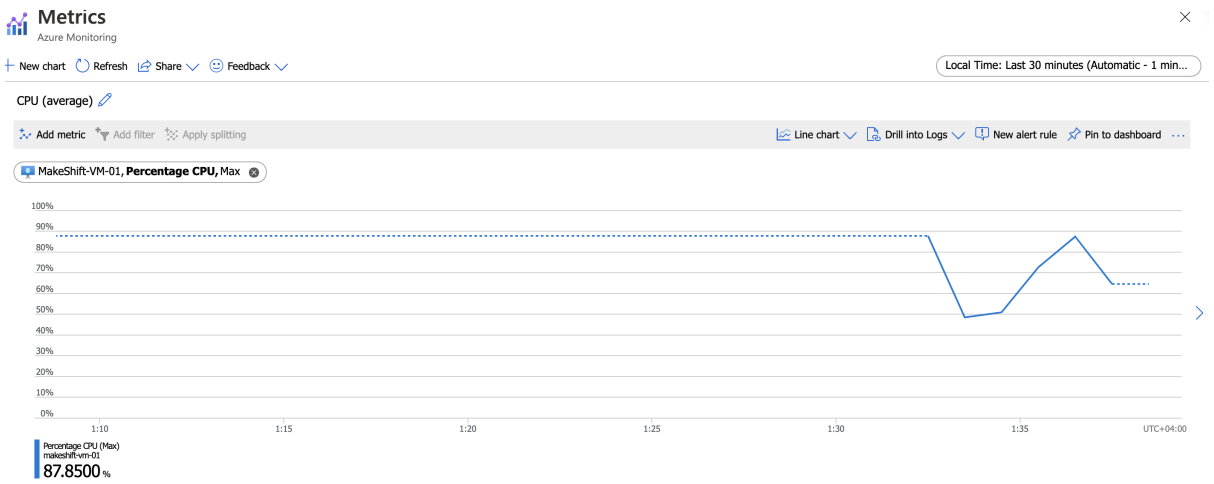

Figure 3: Scale-up process

6.2 Test Case 2: Scale-down process

Scaling down is a process that happened when the usage of CPU and its resources fell below 20%. Scaling down was necessary for this situation so that the unwanted resources will not have been in use until needed. In the graph shown below, the CPU usage went down below 20% (2.54%) during 01:49 pm and it stayed for 15 minutes. After that scale down alert was triggered and the scaling down part of the algorithm was called. It took a whole of 8 minutes for the process to be completed. Once this was done, the old VM and its resources were deleted which took about 5 minutes.

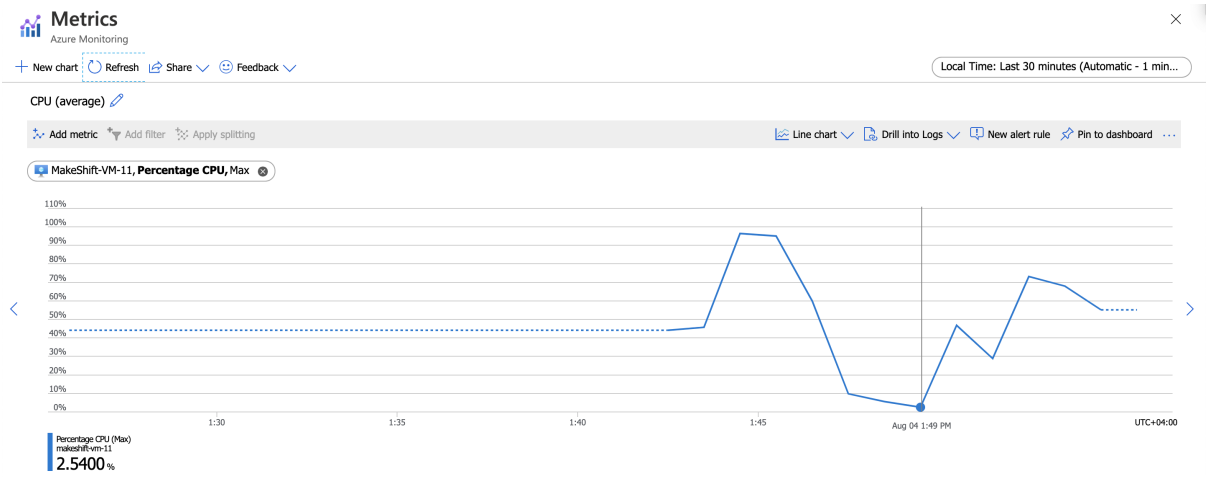

Figure 4: Scale-down process

6.3 Test Case 3: No scale-up or scale-down

This scenario can also be mentioned as the basic flow in this project. This is a place where the CPU usage remained constant without any drastic changes in the workload. When the CPU usage was at 73.15% there was no requirement of scaling up or scaling down the VM. The process remained in a stable state during 01:52 pm. During such instances, the algorithm was not triggered and the VM in use remained to be in use until there was a change in the workload. The same is mentioned in the figure below:

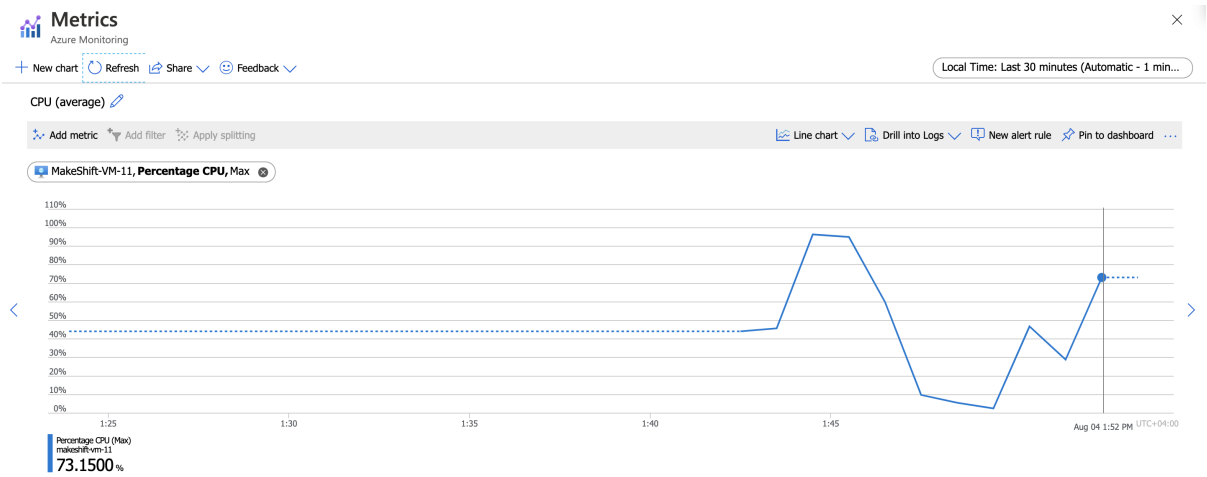

Figure 5: No scale-up or scale-down

6.4 Timeline Graph

The below image is the result that was obtained from testing the project. The first scale-up occurred at 5 minutes where the CPU usage reached 80% and stayed that way for 1 minute. Microsoft Azure API triggered the Scaling up alert and the cloning process and the deletion of the old VM along with its resources were performed. After that, the workload was changed to normal and the graph stayed stable for a while. The first scale

down happened at 20 minutes where the CPU usage reached below 20% and stayed the same for 15 minutes. The API then triggered the Scaling down alert and the configuration of the VM was scaled down. The next process was a scale-up that happened at 40 minutes when the CPU was more than 80% and the final scale down occurred at 55 minutes where the usage of the CPU went below 20% again. This image shows the scaling up and scaling down process that took place in a VM for 1 hour.

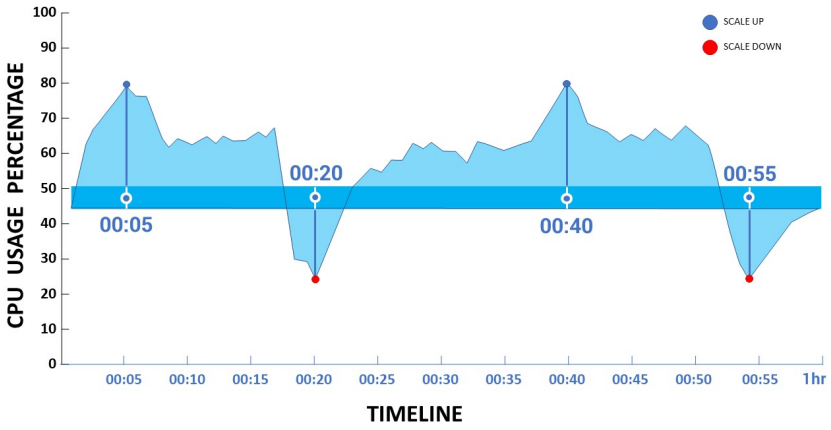

Figure 6: Timeline Graph

6.5 Discussion

The research project efficiently used vertical scaling. This was done by eliminating the downtime that occurred due to varying workloads. This downtime was a result of the CPU usage going up and down dynamically. During such changes, the CPU usage was monitored by the Azure monitor continuously. 'Makeshift Cross Scale Algorithm' was then called, which triggered the alert. The Azure platform was used to host this API. After the algorithm called the API, the process of cloning and deletion of the VMs was carried out at the backend by Microsoft. As the new VM was cloned, with the help of the existing VM, the downtime was eliminated here. Thus, the availability of the application was increased along with efficiency. The results from the above processes were:

The application was up and running during the cloning and deletion process of the VMs without any downtime. The average cloning time was approximately 8.8 minutes and the average time is taken to delete the old VM and its resources were 4.5 minutes.

7 Conclusion and Future Work

This research was successful in answering the research question in hand, where the downtime was eliminated during the process of vertical auto-scaling. The major objective of this project was to eliminate the downtime that was faced during the switching process between the VMs. This was successfully achieved by using vertical scaling in the Azure platform, where the VM in use was cloned with the help of the alerts generated. The written algorithm 'Makeshift Cross Scale' scaled up or down depending upon the workload that was generated by the new requests received. The results obtained from the above processes were:

The application had no downtime during the cloning and deletion process of the VM. Time taken for the cloning process was approximately 8.8 minutes and it took 4.5 minutes for the deletion of old VM along with its resources.

Even though the downtime is successfully reduced with the help of this project, there are still some places where the process can be improved. The whole alerting mechanism of the project that is taking place inside the Azure monitor can be improved in terms of its efficiency. As a part of improving this process, we can use third party solutions or custom coding in the future, which will be placed in the window service of a VM, and that will monitor the CPU of that specific VM. Another way to improve this whole project is to use services inside the VM itself, to take care of the alerting mechanism. This will improve the monitoring speed greatly, and the alerts will be sent out to the algorithm much quicker with this method. It can be evaluated in other cloud service providers and can extend the algorithm to support multi-cloud.

References

Ahmad, A. A., Brereton, P. and Andras, P. (2017). A systematic mapping study of empirical studies on software cloud testing methods, *2017 IEEE International Conference on Software Quality, Reliability and Security Companion (QRS-C)*, pp. 555–562.

Al-Said Ahmad, A. and Andras, P. (2019). Scalability analysis comparisons of cloud-based software services, *Journal of Cloud Computing* **8**.

Ali-Eldin, A., Tordsson, J. and Elmroth, E. (2012). An adaptive hybrid elasticity controller for cloud infrastructures, *2012 IEEE Network Operations and Management Symposium*, pp. 204–212.

Bauer, A., Herbst, N., Spinner, S., Ali-Eldin, A. and Kounev, S. (2019). Chameleon: A hybrid, proactive auto-scaling mechanism on a level-playing field, *IEEE Transactions on Parallel and Distributed Systems* **30**(4): 800–813.

Casalicchio, E. and Silvestri, L. (2011). Architectures for autonomic service management in cloud-based systems, *2011 IEEE Symposium on Computers and Communications (ISCC)*, pp. 161–166.

Feng, D., Wu, Z., Zuo, D. and Zhang, Z. (2019). Erp: An elastic resource provisioning approach for cloud applications, *PLoS ONE* **14**.

Han, R., Guo, L., Ghanem, M. M. and Guo, Y. (2012). Lightweight resource scaling for cloud applications, *2012 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (ccgrid 2012)*, pp. 644–651.

Hasan, M. Z., Magana, E., Clemm, A., Tucker, L. and Gudreddi, S. L. D. (2012). Integrated and autonomic cloud resource scaling, *2012 IEEE Network Operations and Management Symposium*, pp. 1327–1334.

Herbst, N. R., Kounev, S., Weber, A. and Groenda, H. (2015). Bungee: An elasticity benchmark for self-adaptive iaas cloud environments, *2015 IEEE/ACM 10th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*, pp. 46–56.

Hwang, K., Bai, X., Shi, Y., Li, M., Chen, W. and Wu, Y. (2016). Cloud performance modeling with benchmark evaluation of elastic scaling strategies, *IEEE Transactions on Parallel and Distributed Systems* **27**(1): 130–143.

Hwang, K., Shi, Y. and Bai, X. (2014). Scale-out vs. scale-up techniques for cloud performance and productivity, *2014 IEEE 6th International Conference on Cloud Computing Technology and Science*, pp. 763–768.

Jayasinghe, D., Malkowski, S., Li, J., Wang, Q., Wang, Z. and Pu, C. (2014). Variations in performance and scalability: An experimental study in iaas clouds using multi-tier workloads, *IEEE Transactions on Services Computing* **7**(2): 293–306.

Lorido-Botran, T., Miguel-Alonso, J. and Lozano, J. A. (2014). A review of auto-scaling techniques for elastic applications in cloud environments, *Journal of Grid Computing* **12**: 559–592.

Nikravesh, A. Y., Ajila, S. A. and Lung, C. (2015). Towards an autonomic auto-scaling prediction system for cloud resource provisioning, *2015 IEEE/ACM 10th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*, pp. 35–45.

Podolskiy, V., Jindal, A. and Gerndt, M. (2018). Iaas reactive autoscaling performance challenges, *2018 IEEE 11th International Conference on Cloud Computing (CLOUD)*, pp. 954–957.

Qu, C., Calheiros, R. N. and Buyya, R. (2018). Auto-scaling web applications in clouds: A taxonomy and survey, *ACM Comput. Surv.* **51**(4).
URL: *https://doi.org/10.1145/3148149*

Rodrigues, G., Calheiros, R., Guimaraes, V., Santos, G., De Carvalho, M., Granville, L., Tarouco, L. and Buyya, R. (2016). Monitoring of cloud computing environments: concepts, solutions, trends, and future directions, pp. 378–383.

Shelar, M., Sane, S. and Kharat, V. S. (2016). Enhancing performance of applications in cloud using hybrid scaling technique, *International Journal of Computer Applications* **143**: 43–48.

Singh, P., Gupta, P., Jyoti, K. and Nayyar, A. (2019). Research on auto-scaling of web applications in cloud: Survey, trends and future directions, *Scalable Comput. Pract. Exp.* **20**: 399–432.

Sotiriadis, S., Bessis, N., Amza, C. and Buyya, R. (2019). Elastic load balancing for dynamic virtual machine reconfiguration based on vertical and horizontal scaling, *IEEE Transactions on Services Computing* **12**(2): 319–334.