National College of Ireland

Optimal Resource Utilization Strategy for Cloud using MAPE-K model and Microservices on Kubernetes Federation

MSc Research Project
Cloud Computing

Apoorva Prasad Jagtap
Student ID: x18151957

School of Computing
National College of Ireland

Supervisor: Dr. Muhammad Iqbal

National College of Ireland
Project Submission Sheet
School of Computing

National College of Ireland

Student Name:	Apoorva Prasad Jagtap
Student ID:	x18151957
Programme:	Cloud Computing
Year:	2020
Module:	MSc Research Project
Supervisor:	Dr. Muhammad Iqbal
Submission Due Date:	23/04/2020
Project Title:	Optimal Resource Utilization Strategy for Cloud using MAPE-K model and Microservices on Kubernetes Federation
Word Count:	9595
Page Count:	23

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:	Apoorva Prasad Jagtap
Date:	26th May 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):	

Optimal Resource Utilization Strategy for Cloud using MAPE-K model and Microservices on Kubernetes Federation

Apoorva Prasad Jagtap
x18151957
MSc Cloud Computing

26th May 2020

Abstract

With an exponential rise in the demand of the cloud based services, it is mandatory to utilize the resources judiciously or efficiently. This is crucial as with the dynamic auto scaling process, sometimes the resources are either over used or not used completely to their potential which leads to wastage. This paper presents a classic strategy for optimal utilization of resources using the Monitoring, Analyzing, Planning and Execution to build service Knowledge (MAPE-K) model proposed by International Business Machines Corporation (IBM) along with the containerized microservices on Kubernetes Federation cloud. The proposed algorithm is designed to compete with the traditional Horizontal Pod Auto scaling (HPA) algorithm in terms of throughput.
Keywords: *HPA, MAPE-K, auto scaling, Microservices, Kubernetes Federation*

Contents

List of Figures

List of Tables

List of Algorithms

1 Introduction

With a rapid increase in the use of cloud based services for several domains, optimal resource allocation, management and utilization have become mandatory aspects at the computational level. A resource can be regarded as a very crucial dynamic entity required for accomplishing a particular task or job in a stipulated time frame. When an organization makes use of any specific cloud service or product, the dynamic auto scaling capability becomes one of the major factors in determining the Quality of Service (QoS) being offered by the respective Cloud Service Provider (CSP). The proposed research aims in developing an algorithm with the help of MAPE-K feedback loop proposed by IBM [2] and Microservices to provide a competitive edge to the traditional HPA (Horizontal Pod Auto scaling) algorithm in Cloud Computing.

It is hard to achieve high performance and throughput both at the same time for a system which requires auto scaling. With auto scaling in Cloud, there comes a need for resource allocation wherein sometimes unpredictable massive resources are allocated to a particular service which becomes directly proportional to the cost. In the (Figure 1) below the red line denotes the actual usage while the blue line denotes the traditional planned capacity. It simply implies that the user is paying for the resources which are underutilised or wasted and this can degrade the QoS leading to customer dissatisfaction.

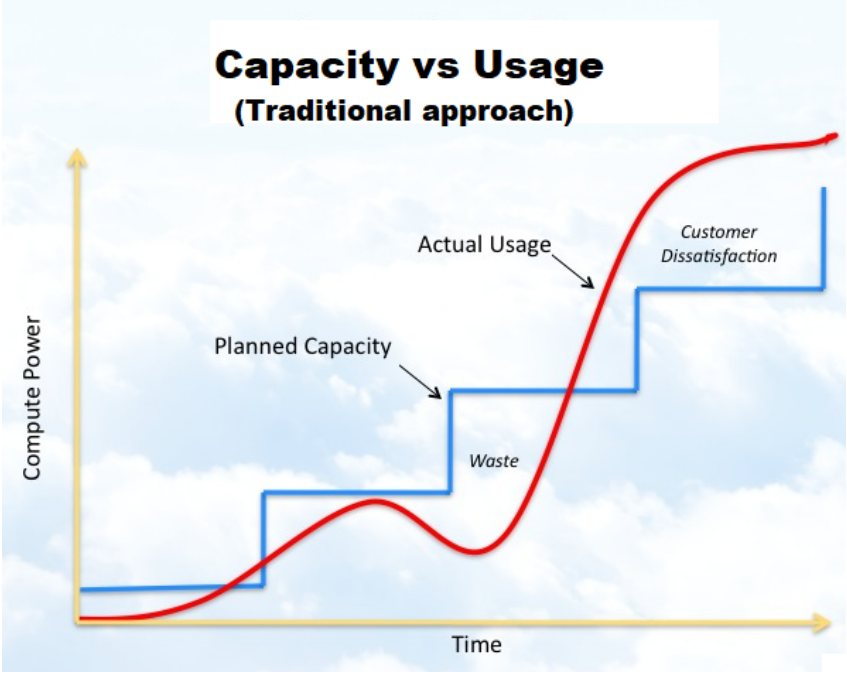

Figure 1: Traditional MapReduce architecture

This research is essentially based on two different research studies conducted by IBM and Netflix. IBM introduced a MAPE-K feedback loop for making autonomic computing versatile by breaking the procedure into phases: monitoring, analyzing, planning and executing the relevant actions on the basis of the knowledge being gathered [2]. Netflix is a classical example of microservice management. It implements the microservices on a very granular level to deliver several services at a time, making it one of the most reliable and successful application today. Cloud Computing serves as a backbone for several technologies which is why it is important for the CSP to deliver the QoS promised as per the Service Level Agreement (SLAs).

1.1 Research Question and Rational

1. Can a traditional Horizontal Pod Auto scaling (HPA) model be replaced by the Kubernetes Federation based approach using the MAPE-K feedback loop and Microservices?

In this, we will be making use of the Kubernetes Federation [3] concept which is an open-source platform originally developed by Google for managing a cluster or a group of clusters. The traditional Horizontal Pod Auto scaling (HPA) approach in Cloud fails to make optimal utilization of the resources which means that the state of the overall utilization of the resources to provide a particular service is either underutilized or overutilized. Both the states can prove to be a loss either in the way of cost or time.

The proposed paper is decomposed into various sections outlining the importance of the research under the suitable headings. Section 2 of this paper elaborates on the research work carried out by several research scholars on the relevant respective topics to analyze and find the best outcome. As this section proceeds further, comparison and similarities between some research topics having similar grounds are also highlighted.The end of this section, finds a major focus on why the proposed research study is necessary to find an alternative for the traditional HPA approach for optimal utilization of resources. The succeeding methodology Section 3 elaborates on the scientific and proposed approach mandatory for executing the study to achieve the desired aim, it also comprises of the algorithm in brief. Further Section 4 will lay details of the mandatory design specifications required to implement the project along with the underlying architecture. Section 5 will be briefing of the practical implementation of the proposed research. Further, section 6 will be the evaluations and analysis of the results obtained after implementation phase. The report will be concluded by stating the discussion and possible future work on the proposed topic in Conclusion Section 7.

2 Literature Survey

This section will lay out the significance of the quality research work done by the research scholars to contribute a better and efficient solution to the respective underlying problem in the computing domain. All the research papers highlighted in this column are related to the original proposed research in someway or the other. This section is vital as it will help in understanding the flow of the proposed project by contrasting and striking the similarities, dissimilarities of the relevant studies carried so far for rooting out an optimal solution.

2.1 Autonomic Computing

With the advent of autonomic computing, various organizations experienced a noticeable cut down in the cost and time parameters due to the automatic system updates and maintenance over the traditional time as well as cost consuming manual method. Autonomic computing is all about getting the work done automatically in a timely fashion without or less human intervention. Today, in the digital computing era it is necessary to have a reliable and accurate result oriented automated systems. The concept of Autonomic Computing Initiative (ACI) was put forward by IBM [4] which was inspired from the autonomic nervous system of the human body. According to IBM, an ideal automated system must essentially posses following characteristics: automation, adaptivity and awareness [4].

Today, the main reason why cloud industry is flourishing at the exponential pace is because of the integration of the autonomic computing with cloud computing domain. According to Flores et al. [5] decision making is the most proactive and important element in any automatic computing model. Flores et al. proposed a mathematical model using Nash Equilibrium [6] to

enhance the decision making component in the automated virtual cloud environment [5]. This research is significant as the decision making loop is a crucial module in MAPE-K feedback loop [2].

2.2 MAPE-K Adaptive Feedback Loop

In the architectural blueprint for autonomic computing originally published by IBM [1], has defined four major areas of autonomic computing. They are: Self-Configuration, Self-Healing (Error Correction), Self-Optimization and Self-Protection. Self-Optimization is a very important function here which implies the automatic resource control for optimal functioning [4]. For a versatile, accurate and reliable automated system, it is essential to have MAPE-K feedback loop. The block diagram (2) below illustrates how the request can be passed on through the mandatory four different phases in MAPE-K.

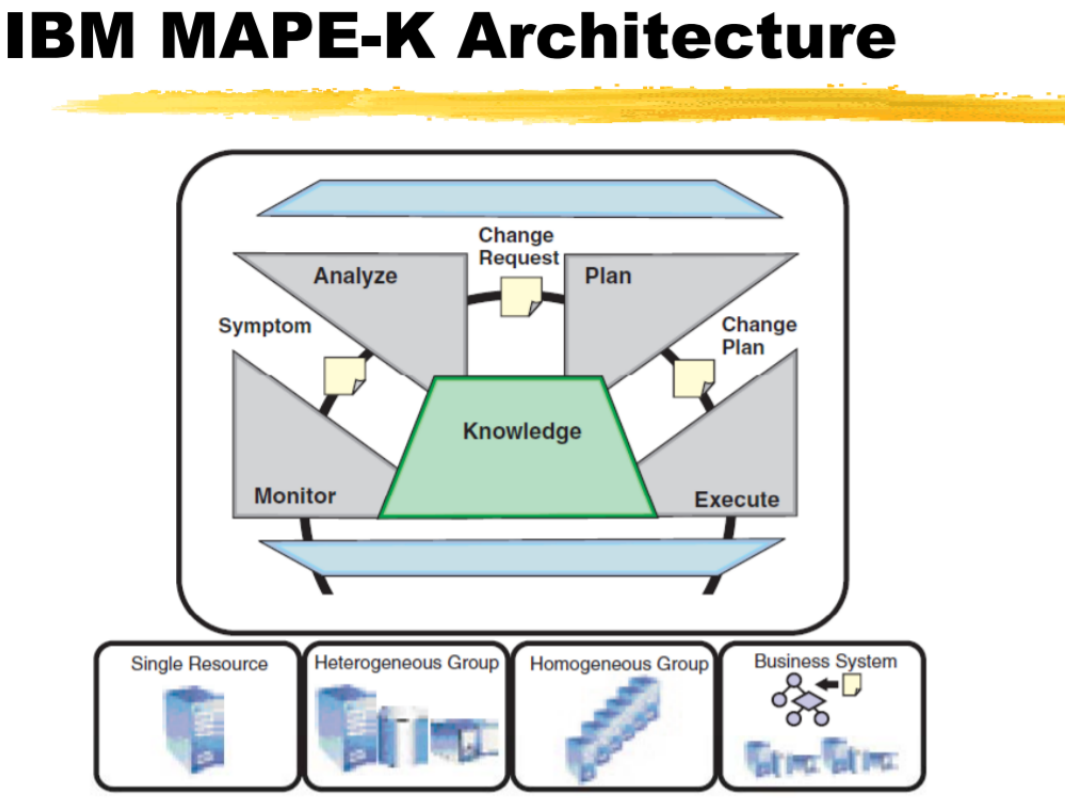

Figure 2: MAPE-K architecture by IBM [1]

Explanation and functioning of the MAPE-K loop in autonomic computing is described below.

1. Monitor: This phase is responsible for monitoring or keeping track of the system parameters from the primary stage of execution. It also ensures to keep the necessary system parameters running in background. Once this is done, the control is passed on to the analysis phase.

2. Analyze: In this phase deep analysis of the scenario is done with the help of binary conditions, scenario checks or control looping depending upon the nature of the platform being utilized. The analyze phase runs in coordination with the active state of the autonomic manager which is a managing component in autonomic computing.

3. Planning: This is extremely crucial phase of the feedback loop as it starts building the prototype of the solution as per the scenario on the basis of the input supplied by the analysis and monitor phases. It also works in integration with the knowledge base which is a repository or collection of the knowledge gathered from previous scenarios. After successful planning the control is forwarded to the final execution phase.

4. Execution: This is the last phase of the loop which implements the suitable actions with the help of the inputs served from the previous phases. This is the most active element which is responsible for executing the planned task or job.

5. Knowledge : This can be regarded as the central element of the MAPE-K feedback loop as it works in integration with all the respective phases to deliver reliable results.

Therefore, this elaborates the underlying architecture of the MAPE-K model proposed by IBM.

2.2.1 Self-Adaptive System

Cloud technology makes use of diverse dynamic platforms for offering various services pertaining to Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and Software as a Service (SaaS). It is vital to have an automated system which can capably adapt itself to the changing platforms and changing requirements to produce the desirable outcome. For example auto scaling, wherein the system is adaptive to the changing needs of the user and it scales up the cloud whenever the demand for resources is more and it automatically scales down if there is no demand of the required service as per the threshold. Although, manual intervention in this scenario can generate reliable outcome, but it will cost time and money leading to inefficient system. Many contributions were made in designing of the self-adaptive systems using the MAPE-K feedback loop. Klös et al. in 2015 came up with an idea of innovating the knowledge bases in the self-adaptive MAPE-K feedback loop [2]. They made the knowledge base of the loop more adaptive by gathering and storing vast outcomes recorded from several scenarios. This research [2] suffered from the time complexity drawback due to complex mathematical modelling. However, [7] came up with a solution that battled the drawback of [2]. Arcaini et al. [7] developed a theoretical computing model namely the multi-agent Abstract State Machines (ASM). The study [7] majorly revolved around modelling and improvising the analysis phase to attain Self-Configuration and automatic resource control for optimal functioning [4] of MAPE-K. The year 2015 saw implementations related to various versatile self adaptive systems and [8] was one of them. Eryilmaz et al. made an attempt to integrate a sensor that would function with the help of MAPE-K loop using opportunistic sensing framework. The reason why this research is authentic because, it enhanced the adaptability of the MAPE-K loop for autonomic system. Eryilmaz et al. [8] proposed a 'Dynamico model' which made use of multiple MAPE-K loops. Although it enhanced the dynamic adaptability of the system, it costed space complexity for restoring the results in the knowledge base.

With the advancing years, research scholars worked up on optimizing the performance of the MAPE-K loop. Klöes et al. developed a parameterized optimal pattern for MAPE-K feedback loop [9]. The research paper [9] states that the time complexity of the MAPE-K loop can be reduced by reducing the interaction of the loop with the unnecessary non-functional constraints in the given scenario. Klöes et al. implemented this by narrowing or controlling the input data set by fetching the mandatory parameters only to the system. Although, the model presented in [9] delivers great precision and resource utilization, it works only for small-scaled systems. Ouareth et al. realized that the generalized MAPE-K model lacked the control component which is why it seems less precise. So,Ouareth et al. proposed the 'Component-Based MAPE-K Control Loop' in [10]. They made use of the Fractal Component Model [10] to develop the improvised loop. Ouareth et al. constructed a flexible loop which reuses the

suitable components as per the scenario which adds a cost cutting edge towards maintenance and resource utilization when compared to the original MAPE-K loop.

2.3 Service Management using Microservices

When it comes to delivering optimal throughput or performance, managing the services in efficient manner becomes very important at service-level. The services today that the computing industries are dealing with are bulky and massive, which makes it complex and this is where the microservices can make an optimal impact. Just to understand how service management works on a basic level, let us have a glance at the traditional MapReduce architecture. From the graphical flow diagram 3 we observe that the big data is first split into chunks in the format of key, value pair and then it is processed from the map phase after which a new key,value pair is obtained as an intermediate output which is then further combined to produce a new reduced key,value pair. This technique works best in terms of throughput for small and moderately fair sized data. It gets complex at service level when the size of the data rises exponentially. This is where the microservices can prove to be the best and effective alternative for managing the services judiciously.

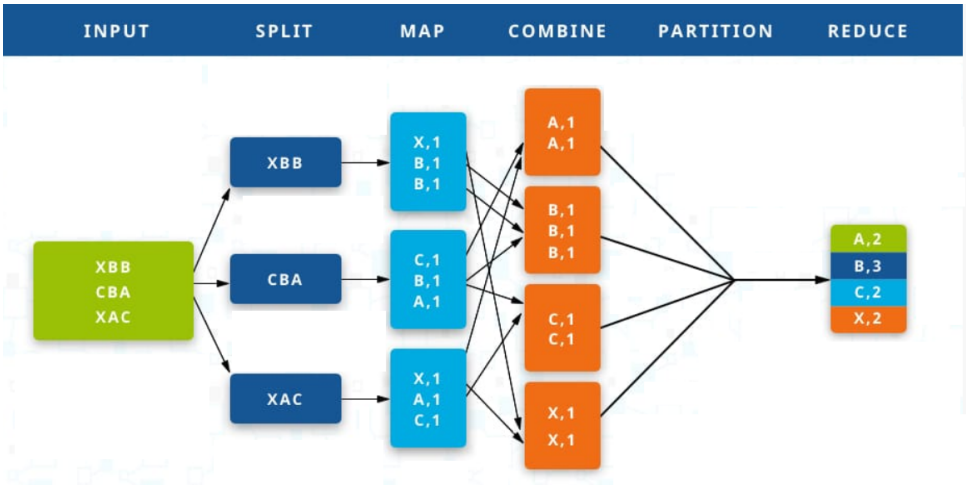

Figure 3: Managing services with Hadoop MapReduce

The block diagram 4 however, illustrates a similar yet fundamentally different approach of the service management process. Here in (figure 4), the task or job is fragmented into requirements with the help of request API. The reason why this approach is superior over MapReduce is because the service management is carried out on the execution level instead of the requirement level. Then the request goes through the developed business logic and then it is split into various small or micro service fragments.

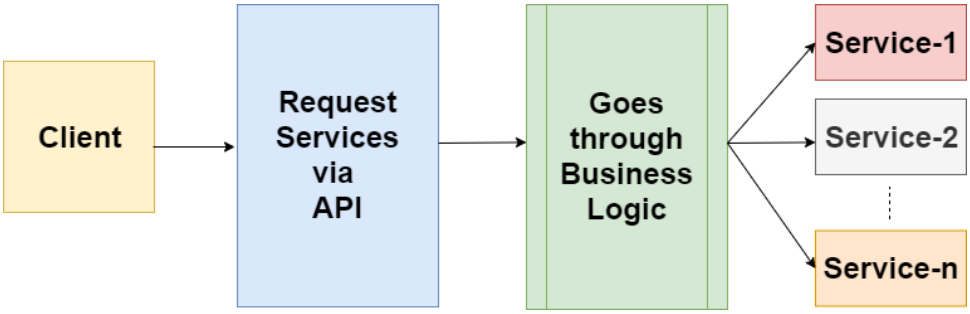

Figure 4: Service managing API in Microservices

2.3.1 Microservices for Real Time Scenarios and Resource Utilization

Cloud containers gel up smoothly as a channel or medium for running the microservices efficiently. In this, the services that are divided into chucks gets distributed amongst the different containers. Every container comprising of the service module is treated as an individual problem which later gets resolved and finally the solutions from all the containers are combined to the obtain the final solution. Lin et al. made an attempt to integrate a probabilistic population-based meta heuristic algorithm called the Ant Colony Algorithm [11] to the container-based microservices in order to optimize the resource scheduling and utilization [12]. The major aim of the research work [12] was to reduce the running overheads and ensure the throughput of the cluster as a whole by optimally making use of the given resources for running the services. Although, this work provides a crucial insights for developing this proposed research, it suffers from the time complexity with the expansion in the cluster topology. As stated earlier, it is not easy to achieve high performance and throughput at the same time in the auto scaling cloud environment. Taking this point as a motivation, Abdullah et al. conducted a deep study on predicting the auto scaling requirement for the cloud-hosted Microservices with the help of trace-driven mathematical modelling [13]. They implemented their strategy on the AWS Cloud. The research [13] succeeded in achieving the optimal response time, but the implementation setup went costly.

2.3.2 Fine grain execution patterns and Machine Learning in Microservices

Netflix, a worldwide famous entertainment application utilizes the containerized microservices as a backbone for catering various services to the user. The functioning of the entire application is divided into three horizontals as visible from figure 5. A request made by the client goes through the business logic which then further goes into the microservice layer. The reason why Netflix makes use of microservices is because it is a bulky application comprising of diverse shows categorized as per the genres and movies as well [14].

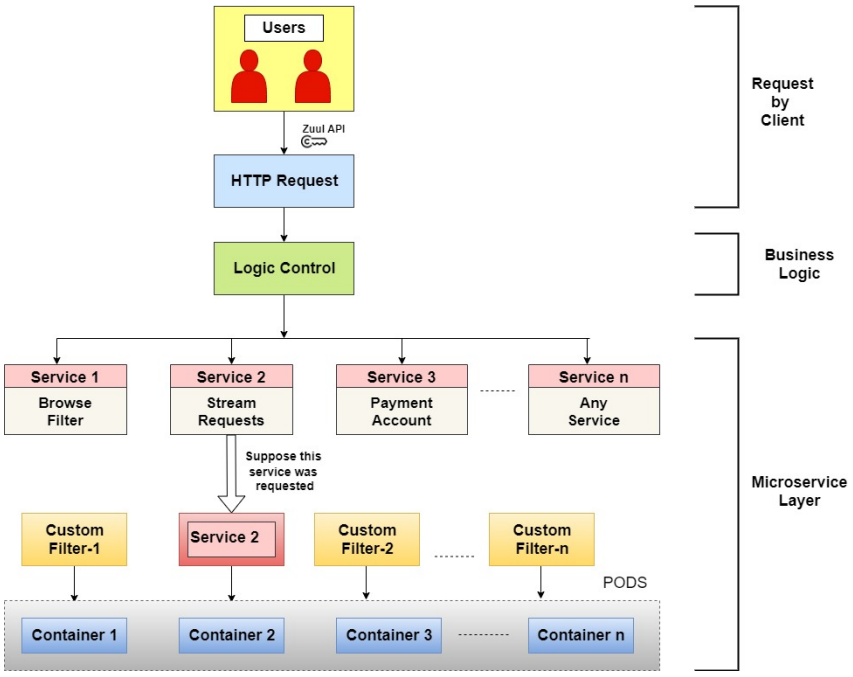

Figure 5: Container based Microservices approach adapted by Netflix

Parekh et al. made a fair attempt to configure a machine learning engine to the underlying Microservice architecture for monitoring resources [15]. What draws the similarity between this

and the proposed research is the usage of the Kubernetes framework. According to the study conducted in [15], monitoring the utilization of the resources is essential as the unused resources can be preserved for future use. It implements Cluster Resource Monitoring (CRM) algorithm and trains the system using supervised machine learning. Abdollahi et al. presented a work on enhancing the availability of the resources using the microservices architecture on the open source Kubernetes platform [16]. The work done in [16] is important as we will be implementing the containerized microservices on the developed kubernetes cloud federation.

2.4 Resource Utilization Scheme

In the world of computing, making the efficient use of resources to reduce the cost and time is crucial. In this part, we will be discussing how various research scholars made an attempt to optimize resource utilization and scheduling in Cloud or on some other platforms like virtual machines. Domanal et al. started the trend of working on efficient utilization of resources on virtual machine using the technique of load balancing in Cloud Computing [17]. In this [17], they compared the traditional resource utilization scheme on CloudSim [18] with the resource utilization on their Virtual Machine (VM) using their load balancing algorithm. Domanal et al. achieved optimal utilization of resources on VM by distributed the load efficiently across the nodes of the clusters. Sonkar et al. in 2016 made a similar attempt of maximizing the resource utilization on cloud by using the VM scheduling technique [19]. This research is superior over the attempt made by Domanal et al. in terms of space complexity. In [19], they successfully lowered the response time and achieved good throughput.

The work stated above were either for a single cloud cluster or for a VM. In real time systems, we deal with multiple group of clusters or multi-tier cloud systems. Khasnabish et al. came up with an innovation of working on the tier-centric resource allocation scheme for the multi-tier cloud systems [20]. Here in [20], they developed a Tier-centric Business Impact and Cost Analysis (T-BICA) optimal resource algorithm. They had setup a threshold for all the four tiers for checking on the optimal resource allocation and utilization. Although their algorithm delivered 65.19 % optimal utilization of the utilities on cloud, the Capital Expenditure (CapEx) cost involved was more. Hussain et al. followed a similar approach of setting up the adaptive threshold parameter in Cloud which would dynamically change as per the demand for efficient utilization of the resources [21]. In this research, more emphasis is laid on optimizing the thresholds of the underutilized resources which can be productive for future use. Ni et al. in 2019 proposed an optimal strategy for resource utilization in cloud data centers [22]. Just like the previous research works, Ni et al. also made an attempt to enhance the threshold parameter using a feed-forward neural network model for dynamic resource utilization as per the scenario. Even though the contribution of [22] was mandatory it however delivered almost same data center efficiency just like a traditional data center. Zhang et al. contributed towards fair allocation of multiple resources for a cloud federation [23]. In this paper, the research scholars worked on utilizing the idle or underutilized resources by sharing such resources with the other CSP in the Cloud Federation. This paper proved to be of vital help as in our proposed research work, we will be making use of the kubernetes cloud federation concept. Zhang et al. achieved good QoS in [23]. Buzato et al. chose different microservices deployment models to enhance the efficiency of the resource utilization on cloud [24]. This research also proved to be of fundamental help as we are making use of microservices. Buzato et al. achieved lower network overheads in [24] by making use of microservices to enhance the CPU Utilization. Therefore, in [24], Buzato et al. accomplished the reduction in network consumption up to 97.6 %.

2.5 Comparative study of the traditional HPA model vs proposed MAPE-K

model This section differentiates the comparative study between the traditional HPA and proposed microservices based MAPE-K loop in brief on the basis of few parameters.

Table 1: Summary of review of literature

Parameter	HPA	MAPE-K	Author/ Reference
Workload Scaling	HPA relies on delivering the solution by increasing the resources allocation to the workloads.	Proposed algorithm relies on developing a adaptive agile microservices pack to ensure more granular level solution to the problem rather than increasing resources unnecessarily..	[7], [8], [9]
Time Management	HPA has a unique way of managing time, it focuses on deducting the duration by enabling additional resources for processing. Although, the time factor is reduced but the resources remain underutilized.	The proposed algorithm works on breaking down the problem into small containerized microservices, which utilizes the time effectively to deliver the optimal use of resources.	[24], [15]
Latency	HPA shows latency in response if the auto scaling method is not properly initiated. This could happen due to overhead in network workloads or unexpected changes in workloads leading to wastage of resources which consume the time for processing, hence resulting in not only latency but also under-utilization of resources.	Addition and evaluation for microservices can sometimes consume extra time leading to a latency in certain extent, but this is still conceptually less as compared to HPA .	[12], [9]
Complexity	HPA adheres a simple resource scheduling approach which runs on prediction methodology, hence the evaluation process is simple and complexities are less.	Here the complexities may keep on arising if the service containers are not well defined from a granular level.	[23],[24], [25]

The review of literature is elaborated in the flow of the factors essential for developing the proposed research. In the proceeding section, we will see how the proposed algorithm is developed using the MAPE-K concept in integration with the microservices on the kubernetes cloud cluster to achieve optimal resource utilization.

3 Research Methodology

For any research problem, it is mandatory to have a chronological path or method before building the prototype or implementing it. Practical and theoretical procedure of developing the pseudo-code, flowchart or algorithm helps in refining the topic more clearly. This section is partitioned into two main subsections namely, 3.1 Scientific Methodology and 4.1 Proposed Approach.

3.1 Scientific Methodology

As seen in flowchart 6, the first step towards this approach is to start the Ubuntu virtual OS. For implementing this project, we have made use of the python 3.6.7 framework to create the kubernetes federation which runs on the combination of microservices and MAPE-K feedback loop. So the next step is to import the coded kubernetes.py zip file and unzip it. After extracting all the contents of the zip file, monitoring phase of the MAPE-K feedback will start which will monitor all the necessary system parameters for executing the aim. The workloads.csv format file will be loaded and running in background after which the traditional HPA algorithm will be executed first for comparison purpose. The total resource utilization while running the HPA algorithm will be recorded and displayed on the screen. It is mandatory to note that, only monitoring and analysis phase will be active for HPA algorithm for recording the proof of its actual performance. MAPE-K loop is not applied entirely to the HPA algorithm, this is because the traditional HPA algorithm does not make use of the MAPE-K loop.

Now for executing the proposed algorithm, all the initial system parameters will be same as that for the HPA. After the monitor and analysis phase, the actual necessary condition check will occur. In this, the total utilization will be rectified whether it is in the range of 65-80 %. Once this condition is evaluated in the active planning phase in the presence of the recorded

knowledge base, a decision is made whether to execute the proposed algorithm for optimal resource utilization or whether to stop if the resource utilization falls under the criteria of below 60% or beyond 80 %. Note that the proposed algorithm makes use of float datatype which implies that if the threshold value goes even a decimal point ahead like (for example) 80.01% , the system will be declared of having overutilized state and if it is like (for example) 59.09% it will be declared as underutilized state.

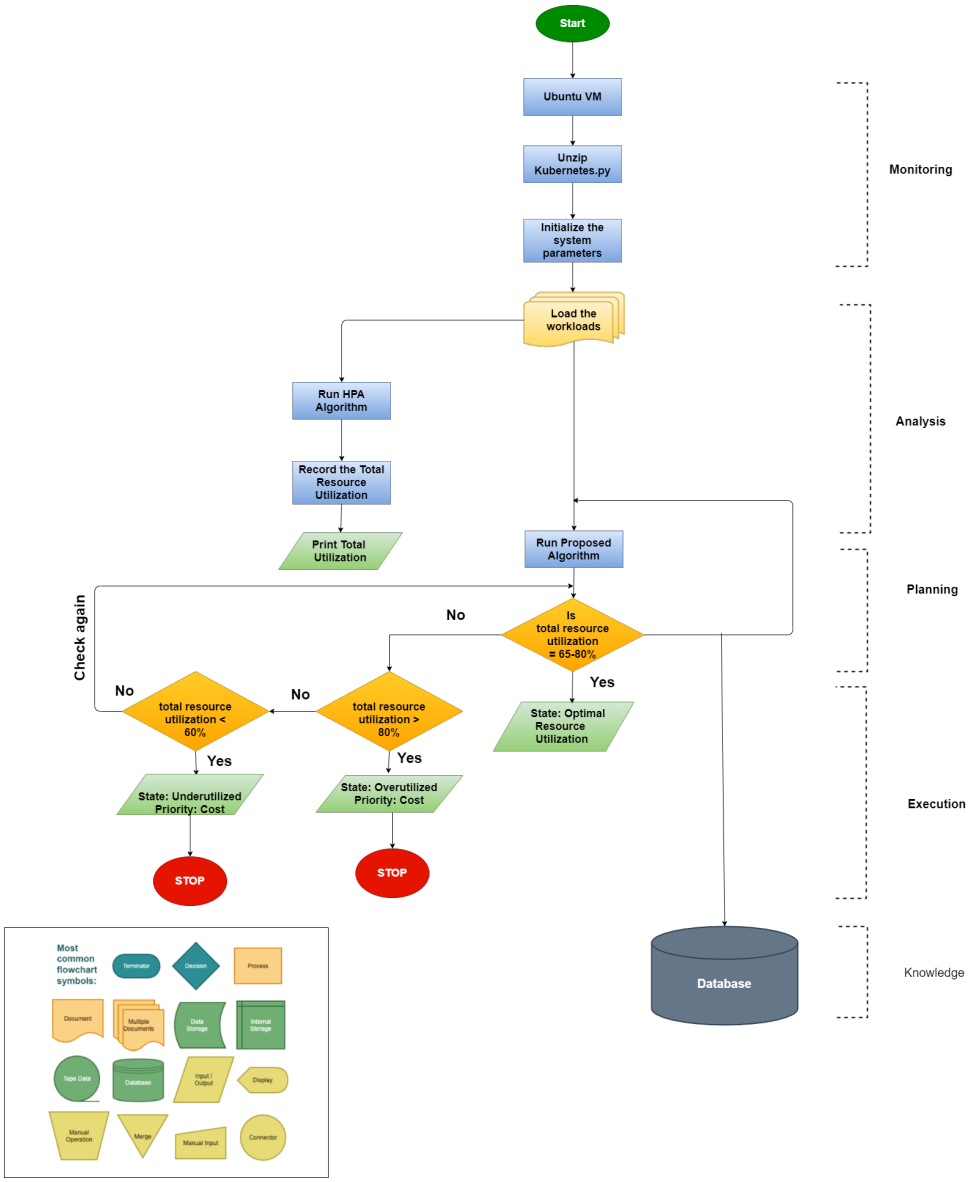

Figure 6: Research Methodology Flowchart

In order to understand the purpose of the different elements involved in making of the flowchart, a small box in the extreme left corner of the flowchart signifies the purpose of the elements being used for building the methodological flowchart.

3.2 Proposed Approach

This section will reveal the algorithmic approach or the pseudo code designed to conduct the proposed experiment.

3.2.1 MAPE-K Algorithmic Approach

As discussed in previous sections, MAPE-K approach is an acronym for Monitoring, Analyzing, Planning, and Execution in the presence of the Knowledge base. The containerized microservice packs run within the Kubernetes federation using the knowledge base on the granular service level. The algorithm or the pseudo code for MAPE-K elaborated below in the respective different phases, for that it is necessary to understand the parameters defined in the below table 2.

Table 2: Variables used in MAPE-K Algorithm

Sr.no	Parameter	Definition
0	$\mathbf{C_t}$	CPU utilization time
1	$\mathbf{C_{up}}$	Overutilization threshold, greater than 80 %
2	$\mathbf{C_o}$	Optimal threshold, 65-80 %
3	$\mathbf{C_{undr}}$	Underutilization threshold, less than 60%
4	$\mathbf{t_i}$	Time iteration
5	$\mathbf{L_w}$	Load workloads
6	$\mathbf{T_{min}}$	Minimum time to deploy
7	$\mathbf{W_t}$	Client input or Workload taken at time when $\mathbf{C_t}$ is registered
8	$\mathbf{W_{t2}}$	Predicted workload at time when $\mathbf{C_{t2}}$ is measured
9	**state**	Overall usage of the system
10	$\mathbf{T_u}$	Total utilization of the system

3.2.2 Algorithm for Monitoring

As stated earlier in section 2, the primary phase of MAPE-K is responsible for monitoring the initial system parameters like: CPU Utilization, start time, state, etc.

Algorithm 1: Monitoring phase

Result: Monitor the values of $\mathbf{C_t}$, state and $\mathbf{W_t}$
Begin
Input: $\mathbf{W_t}$,
Check: CPU utilization State $\mathbf{C_t}$
Store: $\mathbf{W_t}$ and $\mathbf{C_t}$

3.2.3 Algorithm for Analysis Phase

After monitoring phase, analysis is made over the dynamically changing values of the system variables while executing the workloads.

Algorithm 2: Analysis phase

Result: Keep a check on the execution of the HPA algorithm and record the same initial system parameters $\mathbf{C_t}$, state and $\mathbf{W_t}$ for executing the proposed algorithm
input: $\mathbf{W_t}$, $\mathbf{C_t}$,
$\mathbf{t_i}$=$\mathbf{t_i}$+1;
if *algorithm == hpa* **then**
  store $\mathbf{C_t}$, state, $\mathbf{W_t}$ print $\mathbf{T_u}$
else
  Run the proposed algorithm
  State= $\mathbf{W_t}$
end

3.2.4 Algorithm for Planning Phase

This phase is crucial as it can be regarded as a core which is responsible for making decisions to execute a particular plan.

Algorithm 3: Planning phase in MAPE-K

```
Result: Check C_o, C_up, C_undr and decide whether to continue or stop
input: C_t, C_up,C_lo,W_t2;
while time < 180 seconds do
    state=normal;
    priority=normal;
    if C_p > C_up then
        Allocate C_p=P;
        if C_o == 65-80% then
            C_t=Value obtained in Algorithm 2
            go to execute () phase and continue
        else
        end
        if C_p > 80 % then
            state= Overutilization go to execute () phase and stop
        else
            if C_p < 60 % then
                state= Underutilization go to execute () phase and stop
            else
            end
        end
    end
    Go to Analysis Phase;
    if T_u == 65-80 % then
        state= Optimal Resource Utilization
    else
        repeat the loop;
    end
end
```

3.2.5 Algorithm for Execution phase

This phase is highly dependent on the planning phase of the MAPE-K model. It simply executes the actions approved by the other previous phases. Now this is the reason why containerized microservices must be implemented on the granular level. More granular the code, precision is higher and the outcome is reliable.

Algorithm 4: Execution phase in MAPE-K

```
Result: T_u and state
input: Current state of the system (state) and service allocation order based on the
 container microservice manager;
if T_u=C_o then
    print state go to analysis() repeatedly until the user stops the system
else
    if T_u > 80% then
        print State: Overutilized
        stop;
    else
    end
    if T_u < 60% then
        print State: Underutilized
        stop;
    end
end
```

So, this is how the MAPE-K feedback loop is fragmented and executed in chunks as an individual microservice that runs in the specific containers to deliver an output as a whole on granular level. There is no specific algorithm for knowledge base in MAPE-K as it acts as a database repository which gathers and stores the output obtained for the respective scenario with specified set of input. As seen in Section 2, the knowledge base works in integration with all the phases.

4 Design Specification

Every successful research work makes use of some specific utilities like platform, frameworks, tools, systems with special configuration etc for implementing the work in the provided budget. To understand how the modules in the project work together in integrated fashion to deliver the anticipated results, it is mandatory to list down the overall specifications with their respective versions. The table 3 below represents the compulsory requirements needed for implementing the proposed research.

Table 3: Specifications for the proposed model

Purpose	**Tool/ Platform**	**Specifications**
Operating System	Ubuntu 18.04.2 LTS (Bionic Beaver)	64-bit
Virtual Machine	Oracle VirtualBox	-
Programming paradigm	Python 3.6.7	Version 3
Container-Orchestration System	Kubernetes	Open-source
Statistical Evaluation	R	Version 3.6.1

4.1 Proposed Design and Architecture

For the optimal resource utilization strategy using MAPE-K loop and microservices, we have developed a kubernetes federation which in the form of master-slave cluster configuration using the python framework. We will be seeing the architecture of the coded kubernetes cluster in the upcoming sub-section 4.2. The general design description of the model is graphically illustrated in figure 7 below. First, the tasks which in this case are the random workloads generated on the

virtually running Ubuntu environment. After running the proposed algorithm, the output will be scanned through the three proposed scenarios to declare the state. Briefing of each scenario is given below.

Scenario 1- **State: Underutilization:** This scenario arises when the Total Utilization ($\mathbf{T_u}$) of the system is below 60%. A penalty will be charged in terms of cost as the underutilized state of the resources results in wastage of the worthy utilities which can be used in future for a purpose.

Scenario 2- **State: Optimal Utilization:** In this scenario, if the Total Utilization ($\mathbf{T_u}$) of the system is in the range of 65 to 80%, then the system has achieved optimal resource or normal utilization state. Note that for experimental purpose, we have chosen this threshold which is inspired from the recent research works done by [21], [22] and [23]. Also it is substantial to note that, the datatype used while coding the algorithm is float. Therefore, if the value increases even by a single decimal place for instance 80.01%, this condition will fall under the overutilized state.

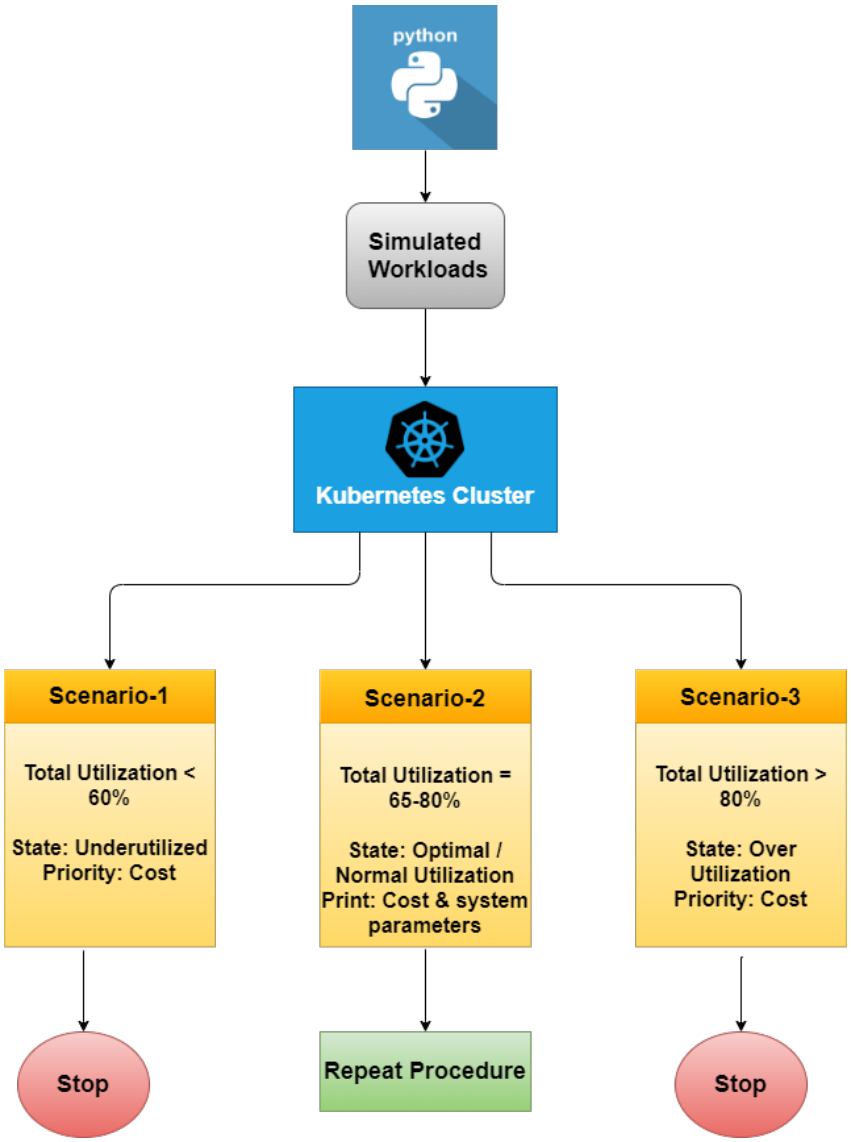

Figure 7: Proposed Design

Scenario 3- State: Underutilization: If the Total Utilization ($\mathbf{T_u}$) of the system is above 80%, the state of the system will be declared as overutilized and penalty will be charged either in terms of cost or time.

Therefore, the above mentioned scenarios are the core part of the research methodology which strongly help in determining the state of the system. Only for normal or optimal utilization state the proposed algorithm will run recursively in the planning phase of MAPE-K. For underutilized and overutilized states it will terminate as explained in 6.

4.2 Cluster Architecture

As stated earlier, we will be making use of the master-slave configuration wherein, the kubernetes federation will be the master under which the AWS datacenter and the Google datacenter will be the slaves. Along with the datacenters, the container packs possessing the respective microservices will be running. The MAPE-K feedback loop will be running in integration with the cluster configuration. The database repository plays a crucial role in gathering the data and providing the data for planning phase majorly for making the decision of executing the proposed algorithm in optimal resource utilization state.

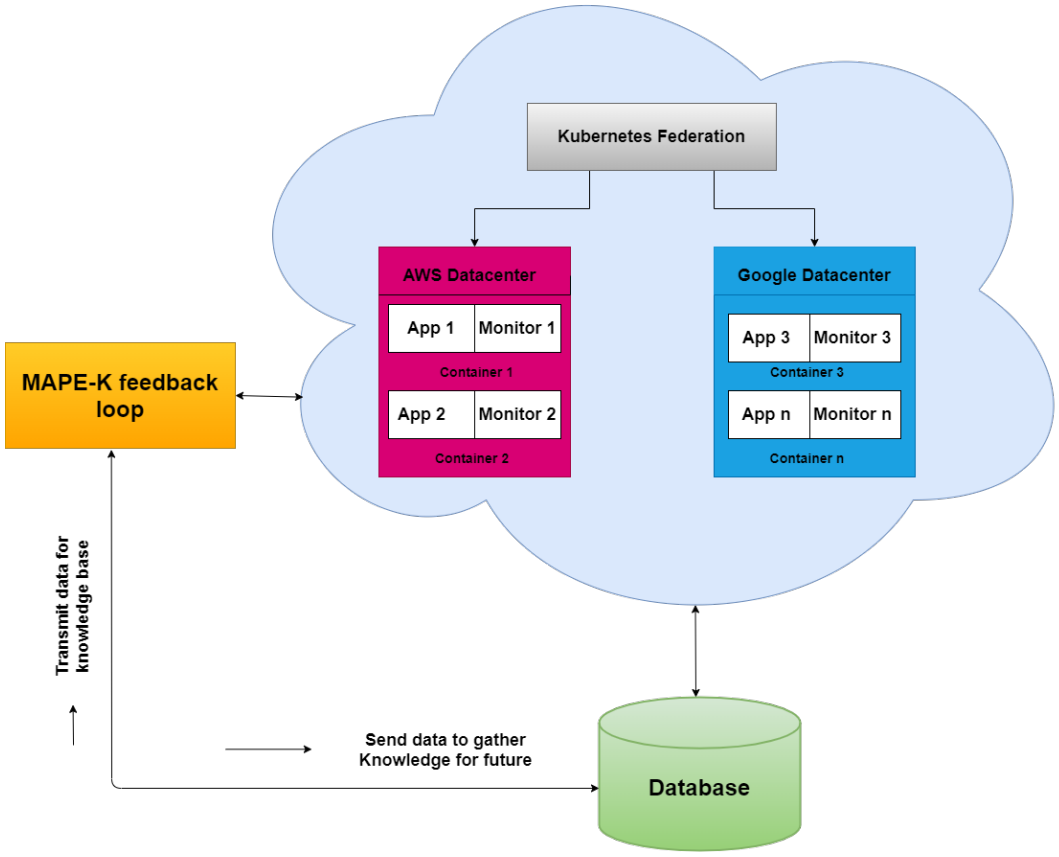

Figure 8: Cluster Architecture

Therefore, this is the proposed design and sequential approach which will be used for implementing the architecture in the next section 5.

5 Proposed Implementation

In this segment, the execution of the proposed algorithm will be done conceptually by applying the theory mentioned in the previous sections 3 and 4. As mentioned earlier in specification (4) part , we will be making use of Python (version 3.6.7) framework extensively for almost everything. For creating the entire kubernetes federation cloud cluster including AWS and google datacenter. The container based microservices MAPE-K model is coded using python as well. The backbone for running the proposed project is the Ubuntu 18.04.2 virtual operating system. After getting the output, deep analysis will be made using the statistical visualization tool R which will be explained in succeeding section 6.

An illustrative class diagram is created in order to understand the flow and connections among all the active entities involved in the research study is shown below (Figure 9). The direction of the arrow heads are extremely crucial in the class diagram for understanding the sequence of the logic.

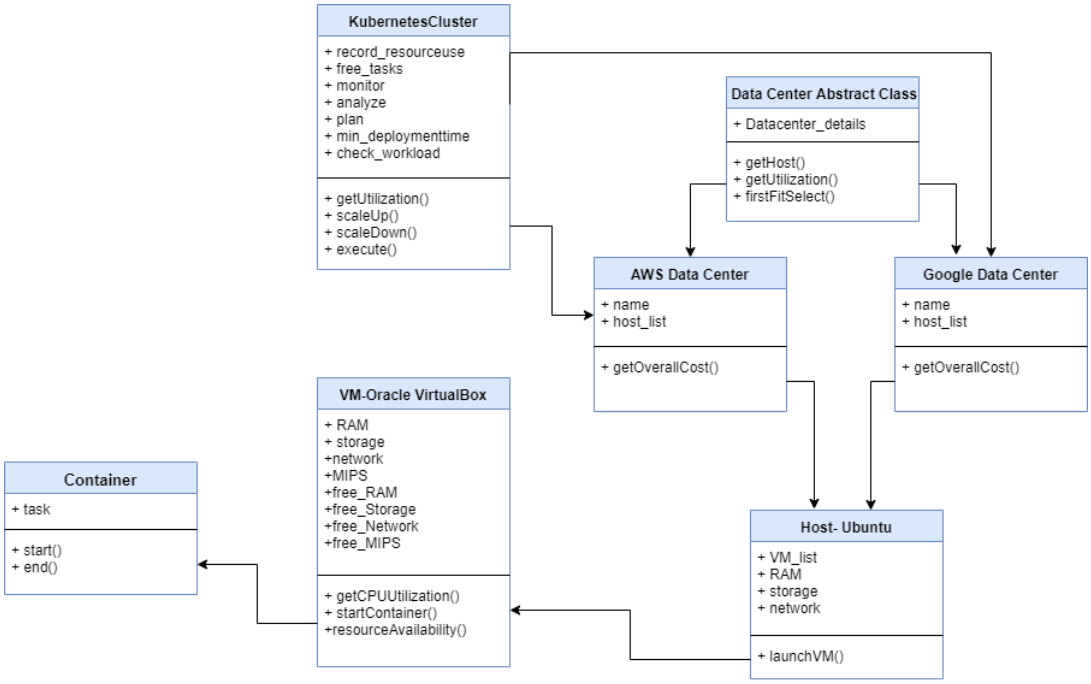

Figure 9: Class diagram for the visual illustration of implementation

A class diagram is mandatory for describing the technical aspects of the code like: classes, methods, packages, attributes and functions. It becomes simple to understand which class is extended or which method is overridden etc. Here, in figure 9 we can see that the super class datacenter is extending the functionalities to the AWS datacenter subclass and Google datacenter sub class. Both of the datacenter sub classes are directly connected to the main class Kubernetes Cluster as shown in figure. The entire cluster federation runs on the virtual Ubuntu host machine. The container runs parallelly with the cluster architecture.

In this research work, we have made use of the threshold concept which is inspired from the recent studies carried out by Ni et al. [22], Hussain et al. [21] and Khanabish et al. [20]. Work done by [22] is quite relevant as Hussain et al. also worked on the total utilization parameter. However, the parameters being considered to study the optimal resource utilization was limited in [22]. In this paper, we are dealing with following parameters: RAM, MIPS, storage, network, CPU and end time. All these parameters are mandatory for getting the Total Utilization ($\mathbf{T_u}$) value. After obtaining the value of the ($\mathbf{T_u}$) a scenario check is done to

evaluate the overall state of the system. As discussed in the preceding sections, for this research study we have considered three scenarios unlike the studies [20] and [21] which only considered one scenario. As the proposed algorithm involves heavy coding and mathematical functions, we decided to choose Python framework. For executing the code successfully, we made use of the Numpy library which is a very powerful package for performing scientific computations and other mathematical functions. Another open source library is the Sklearn python library which is used mostly for training the system (un-supervised or supervised learning), data mining and analysis on Cloud.

The proposed research problem is significant as the optimal resource utilization is necessary to save on the cost and time parameter. The research work is unique as we are making use of three different concepts: MAPE-K feedback loop, microservices and Kubernetes cloud federation in an integrated fashion to come up with an Optimal strategy for efficient resource utilization. Therefore, we were successful in implementing the proposed research work in the stipulated time frame. However, some potential improvements can be carried out in future work to achieve high throughput.

6 Mathematical Evaluations and Analysis

It is extremely important to evaluate and analyze the output delivered by the proposed system mathematically. Evaluation procedure depends upon the nature of the research work. Here, the outputs obtained from the traditional HPA algorithm will be compared with the outputs obtained by the proposed algorithm.

6.1 Analysis of Results

Analysis of the overall results is done after obtaining the final successful outputs. Here, a deep analysis will be done keeping the following system parameters in consideration: RAM, MIPS, storage, network, CPU and end time. These parameters strongly determine the overall resource utilization rate and the state of the system.

6.1.1 Result obtained from the traditional HPA algorithm

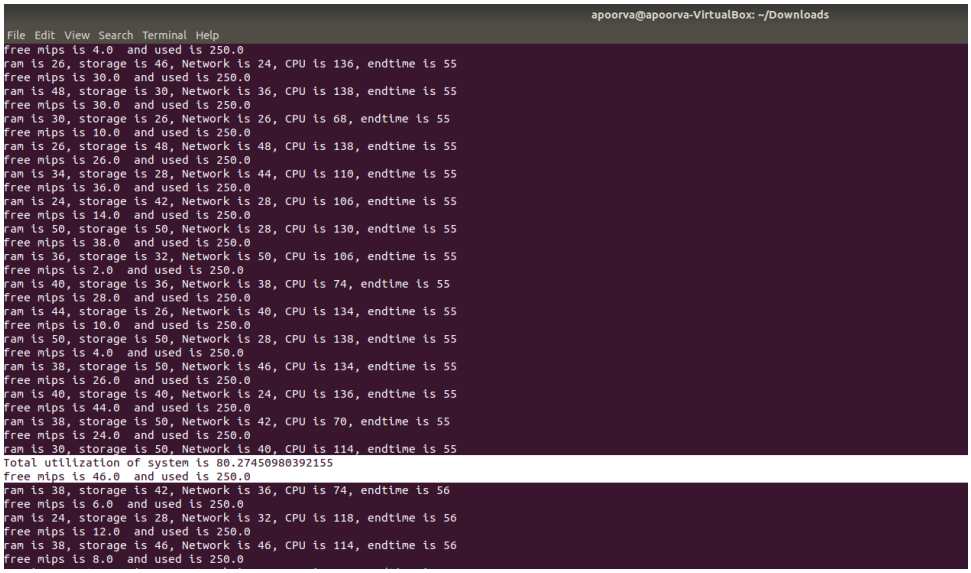

Figure 10: Result delivered by HPA

After unzipping the kubernetes.py file and executing the HPA code in the presence of the running workloads, following is the screenshot of the obtained output. Because it is a traditional algorithm taken from GitHub [26], it only delivers the total utilization of the system and there is no indication of the state. As evident from the screenshot, the total utilization for this is $\mathbf{T_u = 80.27\%}$ which conceptually is the overutilized state. However, it is still not optimal or normal utilization of resources as the resources are utilized over the buffer limit.

6.1.2 Results obtained from the proposed algorithm

For the proposed algorithm as stated in the previous sections, we have three scenarios. After running the traditional HPA algorithm, we execute the code of the proposed algorithm with same system parameters like initial. This will help in making a fair comparison between the two to decide the reliability. Following are the screenshots of the obtained outputs from the proposed algorithm.

1. **State: Underutilization:** From the screenshot 11, we can see that the value of total utilization is $\mathbf{T_u = 57.51\%}$ which clearly falls under **60%** lower threshold criteria hence, state is underutilized. A cost penalty is also charged for it.

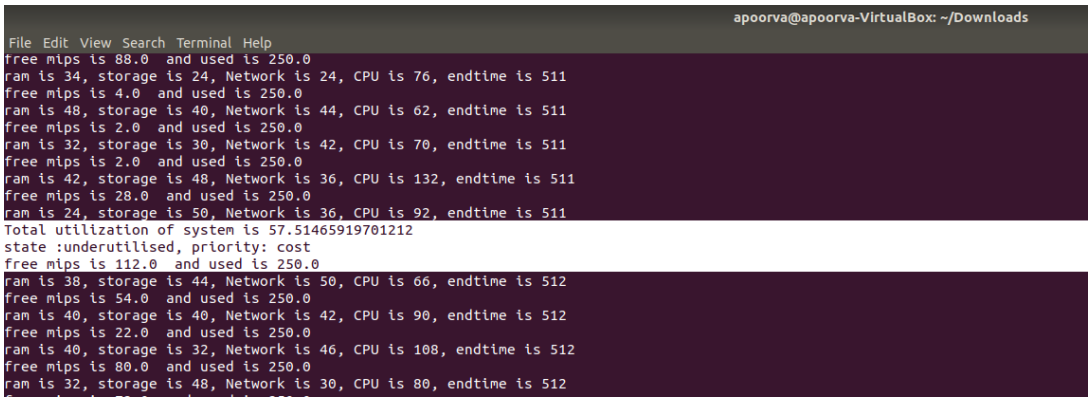

Figure 11: Underutilized State ($\mathbf{T_{undr}}$)

2. **State: Normal or Optimal Utilization:** From the screenshot 12, we can see that the value of total utilization is $\mathbf{T_u = 74.64\%}$ which evidently lies in the range of **65-80%** threshold. Hence, the state is Normal or Optimal and as visible no cost penalty is associated with it.

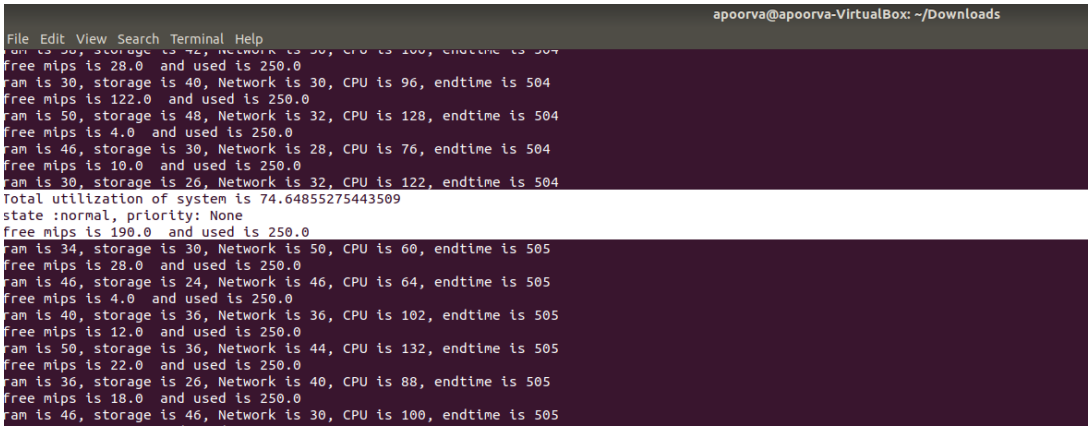

Figure 12: Normal or Optimal Utilization State ($\mathbf{C_o}$)

3. **State: Overutilization:** From the screenshot 13, we can see that the value of total utilization is $\mathbf{T_u} = \mathbf{90.0}\%$ which evidently beyond the upper threshold **80%**. Hence, the state declared is Overutilized and as visible cost penalty is associated with it.

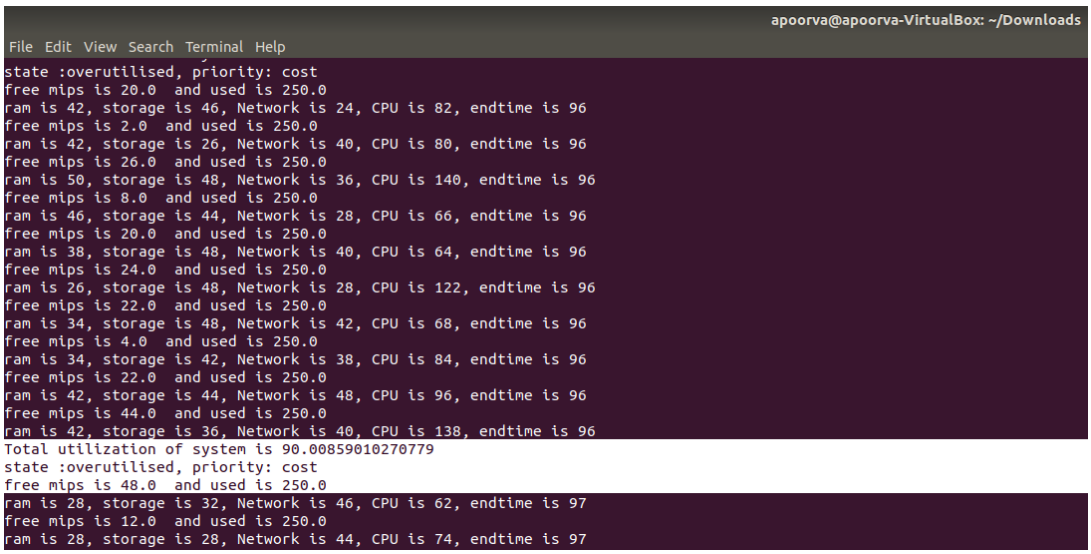

Figure 13: Overutilization State ($\mathbf{C_{up}}$)

After running both the HPA and MAPE-K algorithm several times, their values are noted down and a line graph is plotted for analyzing the throughput statistically. In the visual bar graph, the blue line denotes the output obtained by the traditional HPA algorithm and the orange line denotes the outcome of the proposed algorithm.

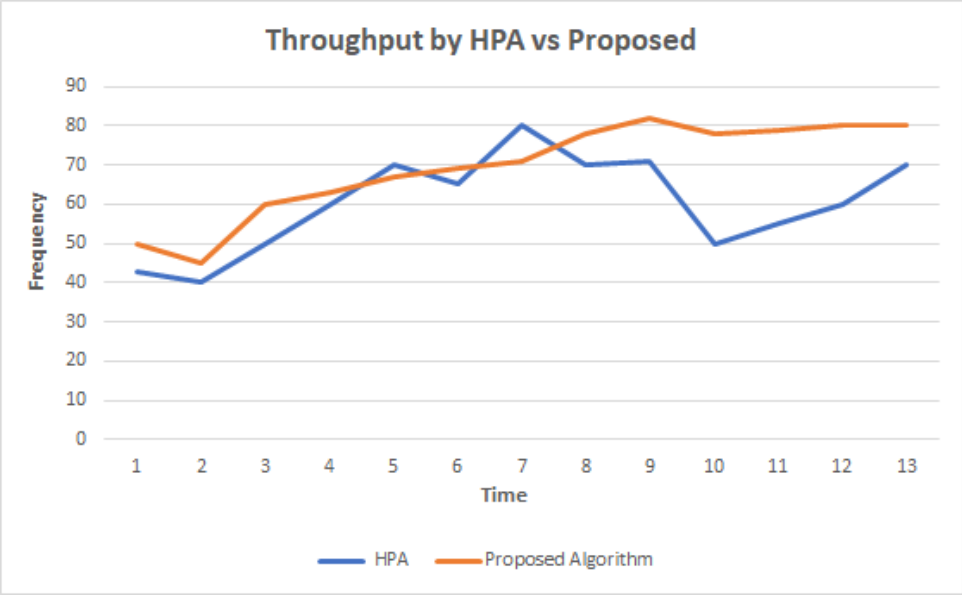

Figure 14: Graphical comparative illustration of the individual performances delivered by HPA and proposed MAPE-K

6.2 Statistical Analysis

In order to perform a statistical analysis, we make use of a very famous tool for visualization purpose. So in this, we are using R-Tool with version 3.6.1. For clear visual analysis we propose

a normalization test. Figure 15 below is the recorded screenshot which displays a fair gap or difference between the mean values of both the algorithms for the overall utilization constraint. There is also a significant gap in the medians of both the algorithms which implies that there is a potential difference in grouping of the values.

Evaluation Statistic	Time	HPA	Proposed
Min	1.00	0.00	0.00
First Qu.	139.50	71.83	83.52
Median	275.00	76.25	89.35
Mean	275.00	70.42	83.57
3rd qu.	425.50	79.24	90.19
Max	570.00	89.85	91.94

Figure 15: Mean and Median values of the HPA vs MAPE-K Models

With the help of R-tool we obtained the visualization in the form of a pictorial histogram illustrated in figure 16. From this we can infer that for most of the runs (in multiple cases), the overall throughput falls in the range of **80 to 100%** for both HPA and proposed algorithm. However, it is evident that for the proposed algorithm the delivery of the throughput rate is higher or it is almost approaching **99%** and is not highly fluctuating like HPA.

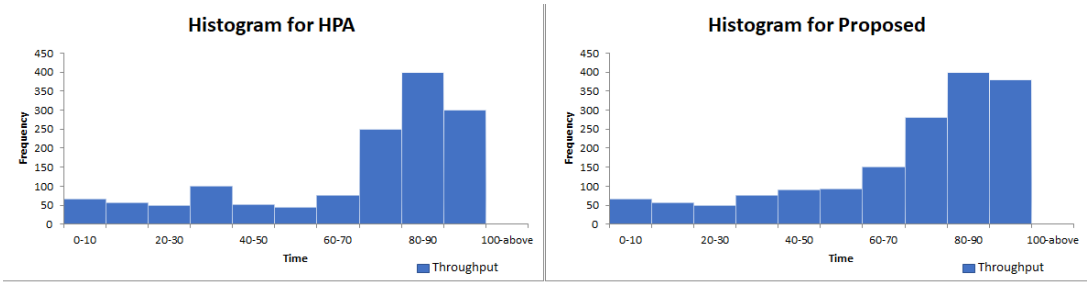

Figure 16: Histogram of HPA vs MAPE-K Model

So this was the generalized statistical evaluation for both the algorithms wherein, the proposed one proved to be superior. Now, in order to confirm the authenticity and degree of accuracy of the results obtained, we will try Wilcoxon Test [27] on the available data.

6.2.1 Non-parametric Statistical Wilcoxon Test

For any de-normalized data such as in this case, Wilcoxon Test [27] which is a statistical non-parametric test is proposed or chosen. Another reason on selecting this particular test is because, it will compare the two paired groups and will find a difference between them. This comparison will help in confirming the authenticity of the delivered result in the Statistical Analysis section 6.2. The null hypothesis for this test can be stated as: Null hypothesis for this research states that the two outputs are significantly varying and based on what the results will be, this hypothesis can be true or false. Outcome of the successful test run is shown in Figure 17 below.

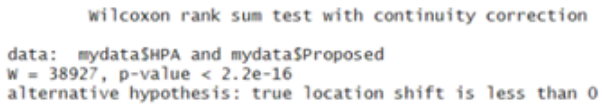

Figure 17: Wilcoxon Test result

From the above obtained outcome, it is visible that the p-value is less than 2.2e-16 and also the sum of the ranks 'W' is a positive number, W=38927. This concludes that there is a non-significant difference between the traditional HPA and the proposed solution just by the factor of 6%, and the confidence of null hypothesis is a staggering 94%.

7 Conclusion and Future Work

This section will brief the essence of the results obtained and what can be rectified in future to make the proposed work ideal from implementation perspective.

7.1 Conclusion

While reviewing the literature work for the proposed research, we came across some research papers that covered the similar concept of innovating a strategy for optimal utilization of resources using unique perspectives. This served as a major motivation for carrying out this research successfully. Most of the recent works done by the scholars involved the concept of threshold for certain parameters. So, we also decided to implement the threshold concept on the Total Utilization $\mathbf{T_u}$ parameter. The central idea of the work is to optimize the resource utilization on a developed cloud federation using the combined concept of MAPE-K feedback loop and containerized microservices. In this, we are comparing the original HPA algorithm with the developed proposed algorithm in terms of Total Utilization $'\mathbf{T'_u}$ which determines the overall throughput of the system. From the Mathematical Evaluations section 6 we can clearly conclude that the resource utilization is judicious or optimal in case of the proposed scheme wherein $\mathbf{T_u} = \mathbf{74.64}\%$ which falls in the range of **65-80%**, whereas for HPA the $\mathbf{T_u} = \mathbf{80.27}\%$ implying the exploitation or overutilization of resources.

As per the statistical analysis conducted in segment 6.2, we can infer from the conducted test (16) that there is some significant difference in the overall throughputs delivered by both the models. However, the difference is not that huge but what makes the proposed scheme more remarkable is that, it delivers a uniform consistent throughput and is not fluctuating visibly like the HPA scheme (figure: 16). This proves that we can readily implement the proposed model in real-time scenario to achieve maximum efficiency. Hence, the proposed research is implemented successfully in the stipulated time frame.

7.2 Future Work

In future, we can try to elevate the optimal utilization threshold range to **80-90%** for which we will have to consider a wider scenario like multiple cloud cluster environment and heavy real-time workloads. Also, the proposed algorithm can be made more potential by reducing or compressing the lines of coding which will add onto a winning edge of good time complexity of algorithm.

References

[1] IBM, "An architectural blueprint for autonomic computing," 2005.

[2] V. Klös, T. Göthel, and S. Glesner, "Adaptive knowledge bases in self-adaptive system design," in *2015 41st Euromicro Conference on Software Engineering and Advanced Applications*, pp. 472–478, 2015.

[3] Kubernetes, "Kubernetes," 2020.

[4] B. OpenMind, "What is autonomic computing-ibm," 2016.

[5] A. A. Flores, R. Mendes, C. Westphall, G. Brascher, and M. Villarreal, "Decision-theoretic model to support autonomic cloud computing," 04 2015.

[6] J. CHEN, "Nash equilibrium," 2020.

[7] P. Arcaini, E. Riccobene, and P. Scandurra, "Modeling and analyzing mape-k feedback loops for self-adaptation," in *2015 IEEE/ACM 10th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*, pp. 13–23, 2015.

[8] E. Eryilmaz, F. Trollmann, and S. Albayrak, "Conceptual application of the mape-k feedback loop to opportunistic sensing," in *2015 Sensor Data Fusion: Trends, Solutions, Applications (SDF)*, pp. 1–6, 2015.

[9] V. Klöes, T. Goethel, and S. Glesner, "Parameterisation and optimisation patterns for mape-k feedback loops," in *2017 IEEE 2nd International Workshops on Foundations and Applications of Self* Systems (FAS*W)*, pp. 13–18, 2017.

[10] S. Ouareth, S. Boulehouache, and S. Mazouzi, "A component-based mape-k control loop model for self-adaptation," in *2018 3rd International Conference on Pattern Analysis and Intelligent Systems (PAIS)*, pp. 1–7, 2018.

[11] E. Weisstein, "Ant colony algorithm," 2018.

[12] M. Lin, J. Xi, W. Bai, and J. Wu, "Ant colony algorithm for multi-objective optimization of container-based microservice scheduling in cloud," *IEEE Access*, vol. 7, pp. 83088–83100, 2019.

[13] M. Abdullah, W. Iqbal, A. Erradi, and F. Bukhari, "Learning predictive autoscaling policies for cloud-hosted microservices using trace-driven modeling," in *2019 IEEE International Conference on Cloud Computing Technology and Science (CloudCom)*, pp. 119–126, 2019.

[14] Netflix, "Microservices in netflix," 2020.

[15] N. Parekh, S. Kurunji, and A. Beck, "Monitoring resources of machine learning engine in microservices architecture," in *2018 IEEE 9th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*, pp. 486–492, 2018.

[16] L. Abdollahi Vayghan, M. A. Saied, M. Toeroe, and F. Khendek, "Microservice based architecture: Towards high-availability for stateful applications with kubernetes," in *2019 IEEE 19th International Conference on Software Quality, Reliability and Security (QRS)*, pp. 176–185, July 2019.

[17] S. G. Domanal and G. R. M. Reddy, "Optimal load balancing in cloud computing by efficient utilization of virtual machines," in *2015 Sixth International Conference on Communication Systems and Networks (COMSNETS)*, pp. 1–4, 2014.

[18] OpenSourceForum, "The cloudsim framework: Modelling and simulating the cloud environment," 2020.

[19] S. K. Sonkar and M. U. Kharat, "A review on resource allocation and vm scheduling techniques and a model for efficient resource management in cloud computing environment," in *2016 International Conference on ICT in Business Industry Government (ICTBIG)*, pp. 1–7, 2016.

[20] J. N. Khasnabish, M. F. Mithani, and S. Rao, "Tier-centric resource allocation in multi-tier cloud systems," *IEEE Transactions on Cloud Computing*, vol. 5, no. 3, pp. 576–589, 2017.

[21] A. J. Hussain, L. Chunlin, and Q. Hammad-Ur-Rehman, "Adaptive threshold detection based on current demand for efficient utilization of cloud resources," in *2019 IEEE 4th International Conference on Computer and Communication Systems (ICCCS)*, pp. 341–346, 2019.

[22] W. Ni, Y. Zhang, and W. W. Li, "An optimal strategy for resource utilization in cloud data centers," *IEEE Access*, vol. 7, pp. 158095–158112, 2019.

[23] G. Zhang, R. Lu, and W. Wu, "Multi-resource fair allocation for cloud federation," in *2019 IEEE 21st International Conference on High Performance Computing and Communications; IEEE 17th International Conference on Smart City; IEEE 5th International Conference on Data Science and Systems (HPCC/SmartCity/DSS)*, pp. 2189–2194, 2019.

[24] F. H. L. Buzato, A. Goldman, and D. Batista, "Efficient resources utilization by different microservices deployment models," in *2018 IEEE 17th International Symposium on Network Computing and Applications (NCA)*, pp. 1–4, 2018.

[25] S. Newman, "Building microservices-designing fine-grained systems," vol. 3, pp. 17–23, 02 2015.

[26] GitHub, "Mape-k," 2019.

[27] A. HAYES, "Wilcoxon test," 2020.

8 List of Acronyms

Acronyms

ACI Autonomic Computing Initiative. 2

API Application Protocol Interface. 2, 5

ASM Abstract State Machines. 4

CSP Cloud Service Provider. 1, 7

HPA Horizontal Pod Auto scaling. 1, 2, 8, 10, 16–20

IaaS Infrastructure as a Service. 4

IBM International Business Machines Corporation. 1–4

MAPE-K Monitoring, Analyzing, Planning and Execution to build service Knowledge. 1–5, 8, 10–12, 14–16, 18–20

PaaS Platform as a Service. 4

QoS Quality of Service. 1, 7

SaaS Software as a Service. 4

SLAs Service Level Agreement. 1

VM Virtual Machine. 7