IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 24 June 2024, accepted 2 August 2024, date of publication 6 August 2024, date of current version 19 August 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3439617

RESEARCH ARTICLE

Intelligent SLA Selection Through the Validation Cloud Broker System

IHAB SEKH AND KÁROLY NEHÉZ
Institute of Information Technology, University of Miskolc, 3515 Miskolc-Egyetemváros, Hungary
Corresponding author: Ihab Sekh (razzaq.razzaq.ihab@student.uni-miskolc.hu)

ABSTRACT Cloud computing has transformed digital service delivery by providing scalable, flexible access to computing resources, including servers, storage, and applications, under a pay-per-use model. This model utilizes geographically distributed data centers to enhance service delivery and dynamically adjust Service Level Agreement (SLA) pricing based on demand. However, effective resource allocation strategies remain challenging, especially for ensuring low latency and fast execution in real-time applications and interactive services. Increased data center load can degrade performance, impacting cost and productivity. To address these challenges, we developed the Intelligent Validation Cloud Broker System (IVCBS). Uses an algorithm to classify virtual machine (VM) resources and match them with users' request sizes, relying on a mathematical model aligned with the trapezoidal membership function in fuzzy logic. This reduces fuzzy rules and improves decision-making accuracy. We tested 11 types of AWS General Purpose EC2 specifications across 31 data centers in various regions. Implementing and comparing IVCBS with a traditional method through two policies—optimize response time and dynamically reconfigure load—showed that the IVCBS with optimized response time policy outperformed in terms of overall response time, data center processing, and total VM cost. IVCBS addresses scalability and performance challenges by efficiently assigning VMs, managing workload distribution, and preventing data center overload. Improving the average data center request servicing time maintains a high quality of service (QoS) and energy optimization.

INDEX TERMS Cloud computing broker, classification SLA, cloud analyst, network delay.

I. INTRODUCTION

Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction. This cloud model promotes availability and is composed of five essential characteristics, three service models, and four deployment models [1].

Cloud users can access the key elements of the underlying architecture, such as Broad network access, which allows services to be consumed from anywhere; on-demand self-service, which enables usage when desired; resource pooling and virtualization, which combine infrastructure, platforms, and applications; rapid elasticity, which allows for horizontal scalability with pooled resources; and measured service charges based on consumption [2]. The services of cloud computing are broadly divided into three categories: Infrastructure-as-a-Service (IaaS), which is the delivery of huge computing resources, such as the capacity of processing, storage, and network., Platform-as-a-Service (PaaS) supports a set of application program interfaces to cloud applications. Well-known examples are Amazon Web Services, Google App Engine, Microsoft's Azure Services Platform, and Software-as-a-Service (SaaS), which replace the applications running on PCs.

There is no need to install and run the special software on your computer if you use the SaaS [3]. The dynamic nature of cloud computing necessitates efficient resource allocation, which can be challenging due to potential resource shortages and conflicting interests between cloud service providers (CSPs) and cloud service users (CSUs). Service-level Agreement (SLA) negotiations can mitigate these issues, and

The associate editor coordinating the review of this manuscript and approving it for publication was Guangjie Han.

the proposed broker-based mediation framework optimizes these negotiations [4]. Cloud brokerage enhances service availability. Traditional brokers face limitations in ensuring service trust and outcomes. An intelligent cloud broker overcomes these limitations by validating and verifying service trust through factors like response time, sustainability, and accuracy. It also incorporates customer feedback and maps services from a service collection repository, outperforming traditional models in recommending services to cloud users [5].

Selecting the most suitable resources to meet diverse user demands is a significant research challenge. Non-functional Quality of Service (QoS) parameters play a crucial role in ranking these resources. This study proposes using fuzzy logic to handle uncertainties in QoS attribute weights and preclassify resources, reducing computational costs [6]. Fuzzy logic-based optimization algorithms present Fuzzy-RLVMrB and Fuzzy-MOVMrB, designed to balance horizontal and vertical loads across physical machines (PMs) by managing processor, bandwidth, and memory resources. Simulations demonstrate that these algorithms excel in load balancing and energy efficiency compared to other methods [7].

Performance and Resource-Aware Virtual Machine Selection using Fuzzy in Cloud Environment (PRSF) develops a virtual machine selection policy to optimize CPU resource utilization and minimize migration counts. Utilizing the Mamdani fuzzy controller, the PRSF policy enhances decision-making for VM selection, leading to decreased energy consumption and reduced migration events [8].

Furthermore, there are cloud simulators for creating and testing different cloud applications. These simulators are based on parameters like programming languages, availability, and SLA support. The analysis considers CloudSim to be the most effective and efficient simulator [9].

Simultaneously, Cloud Analyst is a simulation tool extended from CloudSim. Load balancing is a major challenge in the cloud, where resources have to be directed to their respective servers so that the whole system works efficiently by distributing the workload efficiently. Compare the average response times of the six load balancing algorithms, like Round-Robin, by using a cloud analyst tool to perform a thorough comparative study along with three service broker policies, like optimizing response time, to find out which is the best [10].

Resource stalemates can occur during resource allocation. The currently available algorithms, such as Min-Min and Min-Max, have issues with overhead, hunger, and deadlock. A solution to some of these problems has been proposed that decreases the amount of time required to respond while simultaneously increasing the cloud's overall efficiency [11]. The study introduces an ''Intelligent Cloud Broker Validation System'' designed to enhance cloud computing by optimizing Service Level Agreement (SLA) ratings based on AWS-EC2 specifications, such as VCPUs, RAM, storage, and bandwidth. These factors influence virtual machine (VM) costs, power usage, and processing times, ultimately affecting user confidence and decision-making regarding SLA selections that align with their budget and performance needs.

In this system, we used real data from various sources and analyzed 11 types of AWS-General Purpose EC2 Instances. We considered a scenario with one million customers entering the cloud environment, each with varying request sizes.Using MATLAB, we developed an algorithm to classify and arrange virtual mechanism resources (VCPU, RAM, Storage, and Bandwidth). We also classified user request sizes according to VM arrangements.

The classification was performed using a mathematical model similar to the Trapezoidal Membership Function, which provides classification results directly without the need for numerous fuzzy rules. We defined five linguistic variables: Poor, Fair, Good, Very Good, and Excellent.If the mathematical model's result is equal to (1), it indicates the effectiveness of a decision to validate Broker work, while a result of (0) means that the decision is excluded.

In expanding this research, user requests were distributed among data centers located across six geographic regions (North America(R0), South America(R1), Europe(R2), Asia pacific(R3), Africa(R4), And Australia(R5), comparing the traditional method to the proposed Intelligent Validation Cloud Broker System (IVCBS). Utilizing Cloud Analyst tools, we implemented two distinct broker policies: the Optimize Response Time Policy, which directs user requests to any data center across all geographical regions, and the Dynamic Reconfigure with Load Service Broker Policy, which specifically routes requests to data centers within the same regions as the users affiliated with them.

Both policies were applied using both the proposed IVCBS and the traditional method, across 11 scenario processes each involving one million users. The simulations were conducted using the Cloud Analyst Tool and spanned 31 AWS data centers worldwide.

The findings indicated that the IVCBS with the Optimize Response Time policy consistently outperformed the Dynamic Reconfiguration with Load policy. Moreover, the IVCBS generally surpassed the traditional method across several key metrics: overall response time, data center processing time, total VM cost, and Data Center Request Servicing Times. This demonstrates the effectiveness of the IVCBS approach in enhancing cloud computing efficiency across diverse global settings.

Contributions of This Study:

- *SLA Selection Adjustment: Classifies VM resources and user request sizes based on the proposed mathematical model to assign the appropriate SLA to execute user requests efficiently.*
- *Response Time Improvement: This enhancement significantly improves response times for users, even those located far from data centers, during both peak and off-peak times. This practical benefit underscores the value of the study's findings.*

- *Data Center Processing Reduction: This strategy shares VM resources and routes user requests among locally and globally distributed data centers. The result is a noticeable reduction in processing times, leading to a smoother user experience.*
- *Cost Reduction: This strategy decreases VM costs by matching user requests with suitable SLA specifications.*
- *Data Transfer Cost Reduction: This policy applies to the chosen Broker and reduces round-trip time by sharing VM resources across all geographic regions.*
- *Power Efficiency Improvement: Enhances device power efficiency by reducing data center request servicing times.*

The paper is structured into five parts: it begins with an introduction that briefly summarizes the problem and research question. The second section reviews crucial previous studies. The third section outlines the theoretical framework and modelling strategy and highlights the research gap. It includes an overview of cloud analyst tools, the proposed methodology, the mathematical modelling approach and the data sources and analysis methods. The fourth section on experimentation and analysis presents the research findings. The discussion section then examines the implications of these findings. Finally, the paper concludes by summarizing the research conducted.

II. RELATED WORKS

If Cloud computing delivers computing resources via a network as a service. With the fast adoption of this emerging technology in practical scenarios, understanding how to assess its performance and security challenges has grown increasingly significant. Nowadays, modelling and simulation technology is a valuable and potent resource among cloud computing researchers to tackle these issues [12].

Qazi et al. [13] examine SLA methodologies in cloud computing, detailing their taxonomy, challenges in QoS management, evaluation metrics, and design goals. It also highlights open research areas, guiding future development for enhanced service delivery and CSP-CSU accountability. Chauhan et al. [14] emphasized the role of cloud brokers within an interconnected cloud computing framework. Their study explored the advantages and limitations of cloud brokers, focusing on aspects like pricing, optimization, trust, and Quality of Service (QoS). Being a survey, the paper provides in-depth discussions to enhance the comprehension of cloud brokers in multi-cloud environments. Yao et al. and Ahmad et al. [15] introduce the Cost Optimization based on Task Deadline (COTD) algorithm for cloud and fog services, aiming to reduce costs by 35% without compromising response times. Tested with Cloud Analyst, COTD outperforms existing routing strategies, offering efficient real-time decision-making for service providers.

Reference [16] detailed the diverse roles played by cloud service brokers, including intermediation, aggregation, arbitration, integration, and customization. Therefore, the process of delivering services is a collaborative effort involving cloud service providers, cloud service brokers, and customers. Any issues arising within any of these parties will undoubtedly impact the broker's performance.

Cinar [17] aim to bolster security and compliance in multi-cloud environments by leveraging sophisticated encryption and IAM strategies and legal insights. They underscore the role of cloud service brokers in applying best practices to overcome challenges posed by technology adoption and regulatory intricacies. Petcu [18] tackled the interoperability issue among cloud services, highlighting the challenge posed by vendor lock-in and the necessity to integrate different clouds to meet user needs.

Despite the existence of hybrid clouds, linking multiple cloud services is crucial for enhancing performance and user satisfaction. The authors suggested a strategy to enable portability and interoperability across various cloud providers. However, this proposal lacks a detailed practical method for addressing the interoperability challenges among cloud service providers.

Chafai et al. [19] This paper proposes a performance evaluation model for federated clouds using an open Jackson network, focusing on service diversity and user demand to improve system design. Calheiros et al. [20] explored the constraints a solitary cloud provider faces in service delivery. They noted that with the rising demand for services, current methods fell short regarding Service Level Agreements (SLA) and Quality of Service (QoS). The authors introduced an inter-cloud framework that leverages agents to address these issues. These agents publish, discover, and deliver services to cloud users under agreed-upon SLAs.

Nonetheless, the paper does not cover the decision-making strategies for purchasing and selling services. Al-E'mari et al. [21] This article evaluates Cloud Service Broker policies for Cloud Datacenter selection, highlighting their role in enhancing cloud computing efficiency and addressing challenges to improve Quality-of-Service standards and decision-making. El Karadawy et al. [22] conducted a detailed examination of the cloud analyst simulator, focusing on different load balancing (LB) algorithms and service broker policies. They specifically evaluated three unique LB algorithms: Round Robin (RR), throttled, and Equally Spread Current Execution (ESCE). Achhra et al. [23] examined various service broker policies and load balancing (LB) algorithms. They compared these LB algorithms across different service broker policies and conducted simulations using cloud analysts to evaluate the performance of existing algorithms. This comparison was based on various metrics to assess their effectiveness.

III. PROPOSED SYSTEM ARCHITECTURE

The proposed study focuses on the intelligent identification of cloud services through a meticulous validation process. This validation process ensures that a value of 1 is consistently achieved across all results of the classification algorithm, which applies to both resource allocation and the size of user requests. By maintaining this uniform value, the study

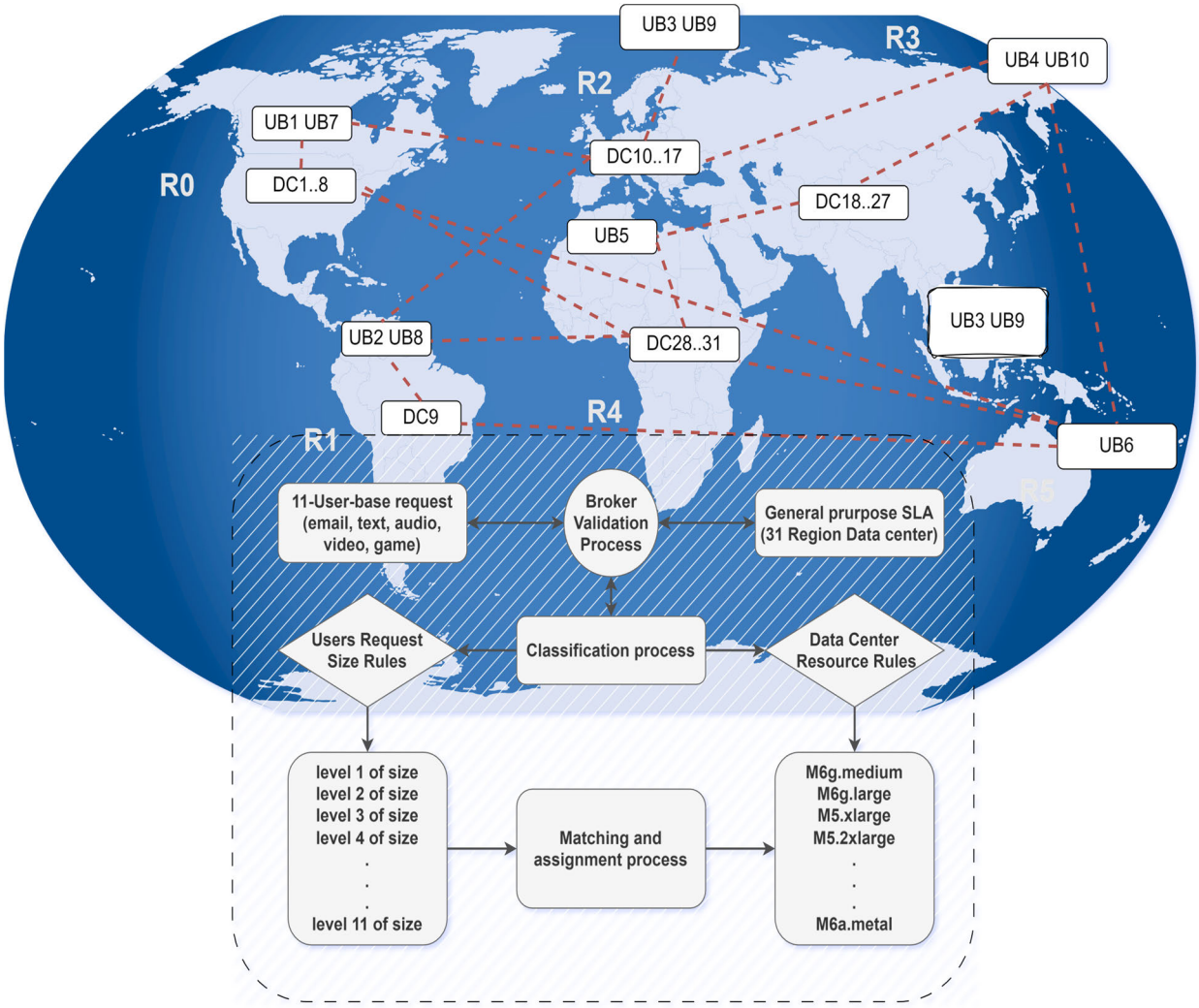

FIGURE 1. Intelligent validation cloud broker system framework.

guarantees that the classification algorithm's outcomes are reliable and accurate, leading to optimized resource management and enhanced service efficiency in cloud computing environments.

This approach underscores the importance of a systematic and consistent validation process to achieve high-quality cloud service identification. Figure 1. Illustrated the proposal system.

A. EXTRACTION OF SERVICE TRUST FACTORS FROM AWS-CLOUD ENVIRONMENT

Within the AWS cloud environment, users have access to a variety of service instance types, including General Purpose (https://aws.amazon.com/ec2/instance-types/),Compute Optimized, Memory-Optimized, Accelerated Computing, and Storage-Optimized, all falling under the broad category of 'XaaS' (Anything as a Service). This study will concentrate on general-purpose EC2 instance types tailored to meet user requirements. General-purpose EC2 instances are strategically deployed across 31 AWS data centers in six geographic regions(https://aws.amazon.com/about-aws/global-infrastructure/regions_az/), ensuring robust global infrastructure and service availability.

1) IDENTIFICATION OF AWS-GENERAL PURPOSE INSTANCE TYPE

Amazon Web Services (AWS) boasts 212 types of EC2 general-purpose instances, meticulously designed to balance computing, memory, and networking resources. These versatile instances excel at diverse workloads, making them ideal for applications requiring equal resource distribution, such as web servers and code repositories [24]. By sharing certain standardized features, these EC2 instances are grouped into 11 categories based on similarities in their specifications. Tables 1 and 2 highlight the adopted AWS-EC2 families' specifications.

while Table 3 lists the actual on-demand cost of each EC2 device, as indicated on AWS's official pricing page

TABLE 1. AWS-general purpose instance features.

EC2-families	AWS-General-Purpose Instance -features		
	Resource efficiency	Instance Storage	Enhance Security
M6g	AWS Nitro system	EBS or Nonvolatile Memory express (NVMe) based solid-state drive (SSD) storage NVMe SSDs	256-bit DRAM encryption
M5	AWS Nitro system	EBS or NVMe SSDs	XTS-AES-256 Cipher
M6i	AWS Nitro system	EBS or NVMe SSDs	Total Memory Encryption (TME)
M6a	AWS Nitro system	Elastic Block Store (EBS)	AMD Transparent Single key Memory Encryption (TSME)

(https://aws.amazon.com/ec2/pricing/on-demand/). Table 4 displays the number of customers entering the cloud for each scenario and the sizes of their requests.

2) THEORETICAL FRAMEWORKAND METHODOLOGY

a: INTELLIGENT SLA SELECTION CONTEXT

In cloud computing, ''intelligence'' signifies the deployment of sophisticated algorithms and decision-making techniques that emulate human cognitive abilities like learning, reasoning, and problem-solving [25]. Within the Intelligent Validation Cloud Broker System (IVCBS), this intelligence is harnessed through optimization algorithms based on a mathematical model inspired by the trapezoidal membership function. This approach enhances service level agreement (SLA) selection and boosts overall system efficiency.

Our method provides adaptability and utility, making it a valuable tool for scientists and researchers facing decision-making in ambiguous situations that require precise and comprehensive insights. It facilitates the assessment of a value's impact on the environment in connection with the decision-making process.

Figure 2 demonstrates the mathematical approach closely aligning with the behavior of a trapezoidal membership function. The following equations and ideas serve as the foundation for the results of the proposed algorithms in Table 5.

The behavior of a trapezoidal membership function can be closely aligned with the equation of a straight line:

$$y = mx + c \tag{1}$$

Here, 'm' represents the slope of the line, and 'c' stands for the y-intercept. This is the most used equation form for a straight line in geometry. However, the straight-line equation can be presented in various forms, including point-slope.

The equation of a straight line with a slope 'm' that passes through a specific point (x1, y1) is derived using the point-slope form, which is expressed as:

$$y - y1 = m(x - x1) \tag{2}$$

In this equation, (x, y) denotes an arbitrary point on the line [26], [27]. The mathematical model employed in the IVCBS is classifies and arranges virtual machine (VM) resources (e.g., VCPU, RAM, Storage, Bandwidth) and user request sizes. This model defines five linguistic variables: Poor, Fair, Good, Very Good, and Excellent, to evaluate the suitability of SLA selections. The classification results directly influence the decision-making process for validating broker work. A result equal to (1) indicates an effective decision, while a result of (0) suggests exclusion.

We have demonstrated that similar to previous examples, the following steps outline the configuration of the trapezoidal membership function. This continuation ensures a comprehensive understanding of our approach.

- Data Import and Initialization:

This section initializes the fuzzy inference system to explore the intelligent features built into the Intelligent Validation Cloud Broker System (IVCBS), we looked into the complex sorting of VCPU sources, using them as a key example. This strict method is used the same way for all VM resources and user requests, as shown in Figure 3 of the MATLAB script. This makes sure that the SLA-level classification is correct and reliable. Moreover, to demonstrate the

TABLE 2. AWS-general-purpose series attributes and specs.

AWS-General-Purpose series Attributes and specs					
EC2-Series	VCPU	RAM GB	Storage GB	Bandwidth Gbps	VCPU-clock speed GHz
M6g.medium	1	4	1	2	2
M6g.Large	2	8	2	4	2
M6g.Xlarge	4	16	4	8	2.4
M5.2XLarge	8	32	8	10	2.5
M5.4XLarge	16	64	12	12	2.5
M6gd.8XLarge	32	128	16	14	2.5
M6gd.12XLarge	48	192	24	16	2.7
M6g.metal	64	256	32	18	2.7
M5d.metal	96	384	48	24	3.4
M6i.metal	128	512	64	30	3.4
M6a.metal	192	768	88	40	3.4

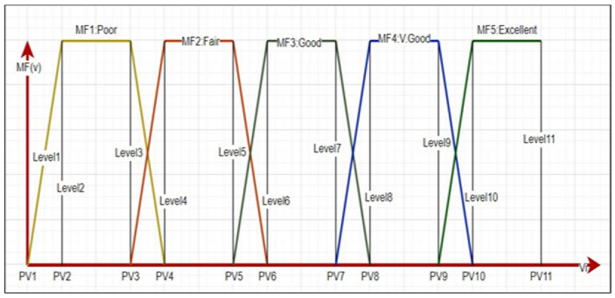

FIGURE 2. Symmetrical trapezoidal shape function.

TABLE 3. AWS data centers and general costs.

6-Geographical (31-Regions)	EC2-General purpose cost										
	M6g.meduim	M6g.large	M6g.xlarge	M5.2xlarge	M5.4xlarge	M6gd.8xlarge	M6gd.12xlarge	M6g.metal	M5d.metal	M6i.metal	M6a.metal
R0-N. virgina	$0.0385	$0.077	$0.154	$0.384	$0.768	$1.4464	$2.1696	$2.464	$5.424	$6.144	$8.2944
R0- Ohio	$0.0385	$0.077	$0.154	$0.384	$0.768	$1.4464	$2.1696	$2.464	$5.424	$6.144	$8.2944
R0- N. California	$0.0448	$0.0896	$0.1792	$0.448	$0.896	$1.696	$2.544	$2.8672	$6.384	$7.168	$9.6768
R0- Oregon	$0.0385	$0.077	$0.154	$0.384	$0.768	$1.4464	$2.1696	$2.464	$5.424	$6.144	$8.2944
R0- Canada Central	$0.0428	$0.0856	$0.1712	$0.428	$0.856	$1.6128	$2.4192	$2.7392	$6.048	$6.848	$9.2448
R0- Canada west (Calgary)	$0.0428	$0.0856	$0.1712	$0.428	$0.856	$1.6128	$2.4192	$2.7392	$6.048	$6.848	$8.3922
R0- AWS GovCloud (US-East)	$0.0484	$0.0968	$0.1936	$0.484	$0.968	$1.7168	$2.5644	$3.0976	$6.864	$7.744	$9.1428
R0- AWS GovCloud (US-West)	$0.0484	$0.0968	$0.1936	$0.484	$0.968	$1.7168	$2.5644	$3.0976	$6.864	$7.744	$9.3648
R1- São Paulo	$0.0612	$0.1224	$0.2448	$0.612	$1.224	$2.304	$3.456	$3.9168	$8.64	$9.792	$13.2192
R2- Frankfurt	$0.046	$0.092	$0.184	$0.46	$0.92	$1.744	$2.616	$2.944	$6.528	$7.36	$9.936
R2- Ireland	$0.043	$0.086	$0.172	$0.428	$0.856	$1.6128	$2.4192	$2.752	$6.048	$6.848	$9.2448
R2- London	$0.0444	$0.0888	$0.1776	$0.444	$0.888	$1.6768	$2.5152	$2.8416	$6.288	$7.104	$9.5904
R2- Milan	$0.0448	$0.0896	$0.1792	$0.448	$0.896	$1.6896	$2.5344	$2.8672	$6.336	$7.168	$9.6768

TABLE 3. *(Continued.)* **AWS data centers and general costs.**

R2- Paris	$0.045	$0.09	$0.18	$0.448	$0.896	$1.6896	$2.5344	$2.88	$6.336	$7.168	$9.6768
R2- Spain	$0.043	$0.086	$0.172	$0.428	$0.856	$1.6128	$2.4192	$2.752	$6.048	$7.172	$9.6865
R2- Stockholm	$0.041	$0.082	$0.164	$0.408	$0.816	$1.536	$2.304	$2.624	$5.76	$6.528	$9.5980
R2- Zurich	$0.0506	$0.1012	$0.2024	$0.506	$1.012	$1.9184	$2.8776	$3.2384	$7.181	$8.096	$9.6878
R3- Hong Kong	$0.053	$0.106	$0.212	$0.528	$1.056	$1.984	$2.976	$3.392	$7.44	$8.448	$9.7136
R3- Hyderabad	$0.0253	$0.0506	$0.1012	$0.404	$0.808	$0.9664	$1.4496	$1.6192	$5.856	$6.464	$5.3328
R3-Jakarta	$0.048	$0.096	$0.192	$0.48	$0.96	$1.808	$2.712	$3.072	$6.768	$7.68	$5.3328
R3- Melbourne	$0.048	$0.096	$0.192	$0.48	$0.96	$1.824	$2.736	$3.072	$6.816	$7.68	$10.368
R3- Mumbai	$0.0253	$0.0506	$0.1012	$0.404	$0.808	$0.9664	$1.4496	$1.6192	$5.856	$6.464	$5.3328
R3- Osaka	$0.0496	$0.0992	$0.1984	$0.496	$0.992	$1.8688	$2.8032	$3.1744	$7.008	$7.936	$10.7136
R3- Seoul	$0.047	$0.094	$0.188	$0.472	$0.944	$1.7792	$2.6688	$3.008	$6.672	$7.552	$10.8944
R3- Singapore	$0.048	$0.096	$0.192	$0.48	$0.96	$1.808	$2.712	$3.072	$6.768	$7.68	$10.368
R3- Sydney	$0.048	$0.096	$0.192	$0.48	$0.96	$1.824	$2.736	$3.072	$6.816	$7.68	$10.368
R3- Tokyo	$0.0495	$0.099	$0.198	$0.496	$0.992	$1.872	$2.808	$3.168	$7.008	$7.936	$10.7136
R4- Cape town	$0.0508	$0.1016	$0.2032	$0.508	$1.016	$1.92	$2.88	$3.2512	$7.20	$8.128	$9.513
R4- Bahrain	$0.047	$0.094	$0.188	$0.471	$0.942	$1.7741	$2.6611	$3.008	$6.653	$8.448	$9.510

TABLE 3. *(Continued.)* **AWS data centers and general costs.**

R4- Israel	$0.0452	$0.0903	$0.1806	$0.449	$0.899	$1.6934	$2.5402	$2.8896	$6.35	$7.1904	$8.2512
R4- UAE	$0.0473	$0.0946	$0.1892	$0.471	$0.942	$1.7728	$2.6592	$3.0272	$6.653	$7.5328	$8.4117

```
TotalNo=length(Input-Value);A=zeros(TotalN,5);%Five columns for MFs:(MF1:Poor,MF2:Fair,MF3:Good,MF4:VGood and MF5:Excellent)
%MF represents the membership function.
for i=1:TotalNo %Total Number of Resources(ex:VCPU) for classification.
    if Input-Value(i) >= pV1 && Input-Value(i)<=pV2 %Classify level1.
       MF1=(((-1/(pV1-pV2))*((Input-Value(i)-pV2)))+1); A(i,1)=MF1;A(i,2)=0;A(i,3)=0;A(i,4)=0;A(i,5)=0;
    end
    if Input-Value(i)>pV2 && Input-Value(i)<=pV3;MF1=1;A(i,1)=MF1;A(i,2)=0;A(i,3)=0; A(i,4)=0; A(i,5)=0;%Classify level2.
    end
      if Input-Value(i)>pV3 &&Input-Value(i)<=pV4 %Classify level3.
       MF1=(((-1/(p4-p3))*((Input-Value(i)-p3)))+1);A(i,1)=MF1;A(i,3)=0;A(i,4)=0; A(i,5)=0;
       MF2=(((-1/(pV3-pV4))*((Input-Value(i)-pV4)))+1);A(i,2)=MF2;A(i,3)=0; A(i,4)=0; A(i,5)=0;
      end
   if Input-Value(i)>pV4 && Input-Value(i)<=pV5; MF2=1;A(i,2)=MF2;A(i,1)=0; A(i,3)=0;A(i,4)=0;A(i,5)=0;%Classify level4.
    end
    if Input-Value(i)>pV5 &&Input-Value(i)<=pV6 %Classify level5
       MF2=(((-1/(pV6-pV5))*((Input-Value(i)-pV5)))+1);A(i,2)=MF2;A(i,1)=0; A(i,4)=0;A(i,5)=0;
      MF3=(((-1/(pV5-pV6))*((Input-Value(i)-pV6)))+1);A(i,3)=MF3;A(i,1)=0;A(i,4)=0;A(i,5)=0;
    end
    if Input-Value(i)>pV6 && Input-Value(i)<=pV7;MF3=1; A(i,3)=MF3; A(i,1)=0;A(i,2)=0; A(i,4)=0; A(i,5)=0; %Classify level6.
    end
    if Input-Value(i)>pV7 &&Input-Value(i)<=pV8 %Classify level7
       MF3=(((-1/(pV8-pV7))*((Input-Value(i)-pV7)))+1);A(i,3)=MF3;A(i,1)=0;A(i,2)=0;A(i,5)=0;
        MF4=(((-1/(pV7-pV8))*((Input-Value(i)-pV8)))+1); A(i,4)=MF4;A(i,1)=0;A(i,2)=0;A(i,5)=0;
    end
    if In(i)>=pV8 &&Input-Value(i)<=pV9 %Classify level8
         MF4=1; A(i,4)=MF4; A(i,1)=0;A(i,2)=0; A(i,3)=0; A(i,5)=0;
    end
      if Input-Value(i)>pV9 && Input-Value(i)<=pV10 %Classify level9
       MF4=(((-1/(pV10-pV9))*((Input-Value(i)-pV9)))+1);A(i,4)=MF4; A(i,1)=0;A(i,2)=0; A(i,3)=0;
        MF5=(((-1/(pv9-pv10))*((Input-Value(i)-pv10)))+1);A(i,5)=MF5;A(i,1)=0;A(i,2)=0;A(i,3)=0;
      end
       if In(i)>pv10 && In(i)<=pv11; MF5=1;A(i,5)=MF5;A(i,1)=0;A(i,2)=0;A(i,3)=0;A(i,4)=0;%Classify level10&11.
        end
end
```

FIGURE 3. **VCPU classification code.**

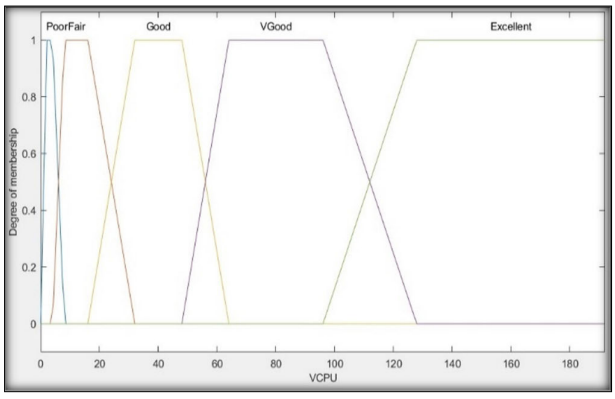

FIGURE 4. **Trapezoidal of CPU levels.**

TABLE 4. **Cloud users and sizes of their requests.**

Cloud users		User request	
Scenario number	Total number of users	SaaS	Size
1	1000,000	App1	3 MB
2	1000,000	App2	5 MB
3	1000,000	App3	10 MB
4	1000,000	App4	35 MB
5	1000,000	App5	70 MB
6	1000,000	App6	105 MB
7	1000,000	App7	140 MB
8	1000,000	App8	750 MB
9	1000,000	App9	1500 MB
10	1000,000	App10	2250 MB
11	1000,000	App11	3000 MB

alignment of our mathematical model with the trapezoidal membership function, we referenced this approach in the discussion on initializing and depicting the membership function, as shown in Figure 5. This MATLAB code is crucial,

serving as a foundational tool for the analysis and enhancement of cloud resource allocation.

```
clear; close all; CLC; warning off fis =
newfis('Classification'); d = xlsread('VCPU.xlsx'); Input-
Value = d(:,1); MAX = max(Input-Value);
```

(FIS) and reads input data from an Excel file ('VCPU.xlsx'), extracting the 'Input-Value' column and determining the maximum value for normalization.

- Defining Membership Functions

```
pV1 = 1; pV2 = 2; pV3 = 4; pV4 = 8; pV5 = 16;

pV6 = 32; pV7 = 48; pV8 = 64; pV9 = 96; pV10 = 128; pV11 = 192;

fis = addvar(fis, 'input', 'VCPU', [0 MAX]);

fis = addmf(fis, 'input', 1, 'Poor', 'trapmf', [pV1 pV2 pV3 pV4]);

fis = addmf(fis, 'input', 1, 'Fair', 'trapmf', [pV3 pV4 pV5 pV6]);

fis = addmf(fis, 'input', 1, 'Good', 'trapmf', [pV5 pV6 pV7 pV8]);

fis = addmf(fis, 'input', 1, 'VGood', 'trapmf', [pV7 pV8 pV9 pV10]);

fis = addmf(fis, 'input', 1, 'Excellent', 'trapmf', [pV9 pV10 pV11 pV11]);

fis = addvar(fis, 'output', 'VCPU Level', [0 MAX]);

fis = addmf(fis, 'output', 1, 'Poor', 'trapmf', [pV1 pV2 pV3 pV4]);

fis = addmf(fis, 'output', 1, 'Fair', 'trapmf', [pV3 pV4 pV5 pV6]);

fis = addmf(fis, 'output', 1, 'Good', 'trapmf', [pV5 pV6 pV7 pV8]);

fis = addmf(fis, 'output', 1, 'VGood', 'trapmf', [pV7 pV8 pV9 pV10]);

fis = addmf(fis, 'output', 1, 'Excellent', 'trapmf', [pV9 pV10 pV11
pV11]);
```

Membership functions (MFs) for the input and output variables are defined using trapezoidal membership functions (trapmf). These functions categorize the VCPU values into linguistic variables: Poor, Fair, Good, Very Good, and Excellent.

- Visualization

```
figure

plotmf(fis, 'input', 1);
```

This visualizes the trapezoidal membership functions. Finally, the specific MATLAB software and libraries, along with the parameters and functions examined in the Intelligent Cloud Broker Validation System, were represented.

After the broker finalizes the classification of user requests and SLA resources using the classification algorithm, it then performs precise matching of the validation results, ensuring that all outcomes equate to 1. This is accomplished through a specialized matching algorithm. This section delves into both algorithms, showcasing their crucial role in guaranteeing intelligent SLA selection for executing corresponding user requests. The following context in this section illustrates both algorithms.

Classification Algorithm

```
Inputs: Parameter Value (PV)set= {PV1, PV2,,,PV11}
Output=Classification with order Parameter Values
Compute the level for each input parameters
1.For each input value (V) from input parameter value set
2.IF (V >=PV1 and V <=PV2)
3.MF1 ← (((-1/PV1-PV2)) *((V-PV2))) +1)
//MF: Membership Functions
4.Output ← (Poor, MF1)
5.Output ← ((Fair, Good, V. Good, Excellent),0)
6.End
7.IF(V>PV2 and V<=PV3)
8.MF1 ← 1
9.Output← (Poor, MF1)
10.Output← ((Fair, Good, V. Good, Excellent),0)
11.End
12.IF (V>PV3 and V<=PV4)
13.MF1← (((-1/(PV4-PV3)) *((V-PV3))) +1)
14.Output← (Poor, MF1)
15.Output← ((Good, V. Good, Excellent),0)
16.MF2← (((-1/PV3-PV4)) *((V-PV4))) +1)
17.Output← (Fair, MF2)
18.Output← ((Good, V. Good, Excellent),0)
19.End
20.IF(V>PV4 and V<=PV5)
21.MF2←1
22.Output← (Fair, MF2)
23.Output← ((Poor, Good, V. Good, Excellent),0)
24.End
25.IF(V>PV5 and V<=PV6)
26.MF2← (((-1/(PV6-PV5)) *((V-PV5))) +1)
27.Outputç←Fair, MF2)
28.Output← ((Poor, V. Good, Excellent),0)
29.MF3← (((-1/PV5-PV6)) *((V-PV6))) +1)
30.Output← (Good, MF3)
31.Output← ((Poor, V. Good, Excellent),0)
32.End
33.IF (V>PV6 and V<=PV7)
34.MF3←1
35.Output← (Good, MF3)
36.Output← ((Poor, Fair, V.Good, Excellent),0)
37.End
38.IF (V>PV7 and V <=PV8)
39.MF3← (((-1/(PV8-PV7)) *((V-PV7))) +1)
40.Output← (Good, MF3)
41.Output← (Poor, Fair, Excellent),0)
42.MF4← (((-1/(PV7-PV8)) *((V-PV8))) +1)
43.Output← (V. Good, MF4)
44.Output←(Poor, Fair, Excellent,0)
45.End
46. IF (V>PV8 and V<=PV9)
47. MF4←1
48.Output←(V. Good, MF4)
49.Output← ((Poor, Fair, Good, Excellent),0)
50.End
51.IF (V>PV9 and V<=PV10)
52.MF4← (((-1/(PV10-PV9)) *((V-PV9))) +1)
53.Output← (V. Good, MF4)
54.Output← ((Poor, Fair, Good),0)
55.MF5← (((-1/(PV9-PV10)) *((V-PV10))) +1)
56.Output← (Excellent, MF5)
57.Output← ((Poor, Fair, Good),0)
58.End
59.IF (V>PV10 and V<=PV11)
60.MF5←1
61.Output← (Excellent, MF5)
62.Output← (Poor, Fair, Good, V.Good),0)
63.End
64.End
```

Matching Algorithm	
1.IF Output (Poor, PV1)	18.End 19.IF Output (Good, PV7)
2.Assign: User base Request (App1)←M6g.medium	20.Assign: User base request (App7)←M6gd.12XLarge
3.End	21.End
4.IF Output (Poor, PV2)	22.IF Output (V. Good, PV8)
5.Assign: User base request (App2)←M6g.large	23.Assign: User base request (App8)←M6g.metal
6.End	24.End
7.IF Output (Poor, PV3)	25.IF Output (V. Good, PV9)
8.Assign: User base request (App3)←M6g.XLarge	26.Assign: User base request (App9)←M5d.metal
9.End	27.End
10.IF Output (Fair, PV4)	28.IF Output (Excellent, PV10)
11.Assign: User base request (App4)←M5.2XLarge	29.Assign: User base request (App10)←M6i.metal
12.End	30.End
13.IF Output (Fair, PV5)	31.IF Output (Excellent, PV11)
14.Assign: User base request (App5)←M5.4XLarge	31.Assign: User base request (App11)←M6a.metal
15.End	32.End
16. IF Output (Good, PV6)	
17.Assign: User base request (App6)←M6gd.8XLarge	

b: MODELING STRATEGY FOR CLOUD COMPUTING APPLICATIONS

This section addresses the handling of ten user-base requests, employing the round-robin algorithm to evenly distribute workloads across virtual machine clusters. It introduces a set of equations that form the mathematical basis for estimating the time required to process a given task. As previously discussed, our framework utilizes 31 individual VMs linked to 31 data centers, spread across six geographical areas and categorized based on 11 clustering factors. The rationale for using a single VM from each AWS-supported data center is to harness suitable computing resources that align with the demand of user requests. This strategy aims to achieve cost efficiency, enhance processing speed, reduce energy consumption, and ensure the availability of additional computing resources to handle other users' requests consistently. To operationalize this concept, we applied the CloudAnalyst tool under a designated service broker policy in two distinct scenarios (optimizing response time and dynamically reconfiguring based on load).

Eq. (3) is given by n as the number of sets for the load (L) or requests that need to be scheduled to servers.

$$L = \{L_1, L_2, L_3, \ldots, L_n\} \tag{3}$$

Eq. (4) is given by k as the number of data centers (dc) In particular data center (DC).

$$\mathrm{DC} = \{\mathrm{dc}_1, \mathrm{dc}_2, \mathrm{dc}_3, \ldots., \mathrm{dc}_k\} \tag{4}$$

The following equation (5) describes the fact that for each Data center dc, there is only one virtual machine VM. Therefore, i is the number of virtual machines in particular data center dc.

$$\mathrm{dc_i} = \{\mathrm{VM_i}\} \tag{5}$$

Eq. (6) is given by DCs_L as the current data centers load.

$$\mathrm{DCs_L} = \{VM_1\mathrm{L}, \mathrm{VM_2L}, \mathrm{VM_3L}, \ldots, \mathrm{VM_kL}\} \tag{6}$$

Eq. (7) describes the load VM_iL of each virtual machine VM_i to be essentially equal.

$$\mathrm{VM_1L} \approx \mathrm{VM_2L} \approx \mathrm{VM_3L}, \ldots, \mathrm{VM_kL} \tag{7}$$

Eq. (8) shows to calculate the time needed to allocate all tasks L to each virtual machine Vmi, take $\tau 0$ as the time to execute the task L.

$$\mathrm{t_0} = \sum\nolimits_{0 \in \mathrm{f(L)}} \tau_0 \tag{8}$$

When there are several virtual machines in a particular data center, which means there are more virtual machines available, all available tasks can be allocated (shared equally) to multiple servers.

$$\mathrm{VM} = \{\mathrm{VM_1}, \mathrm{VM_2}, \mathrm{VM_3}, \ldots, \mathrm{VM_k}\} \tag{9}$$

And k is given as the number of virtual machines in the data center. Eq. (10) shows the execution time will be the summation of all time which can be calculated as Ti of each available task will be executed on that total number of virtual machines in the particular data center.

$$\mathrm{T_0} = \sum \mathrm{T_i}(\mathrm{i} = 1, .., \mathrm{n}) \tag{10}$$

c: CLOUD ANALYST

This framework extends the CloudSim simulator with new capabilities, allowing for the analysis of performance and costs associated with large, geographically dispersed cloud systems under extensive user workloads and various parameters. It offers a user-friendly graphical interface and the ability to customize settings for any geographically distributed system, including hardware configurations like storage, CPU, main memory, and bandwidth. The results of simulations are provided in charts and tables, detailing aspects such as cost, response time, data center processing time, and data center load, among others [28]. Figure 3 depicts the cloud analyst model.

d: ROUND ROBIN ALGORITHM

The round-robin algorithm, known for its simplicity, is popular among load-balancing mechanisms. It evenly distributes the workload by cyclically rotating through each server in sequence. This method effectively manages the queues within load-balancing systems by assigning turns to each virtual server, ensuring a systematic distribution cycle. The process operates on a fixed time allocation known as the time quantum, the designated duration for a process's execution within the system or for processing queued data. This approach is notably equitable, as it does not prioritize any process over others; each receives an equal time allotment, calculated as

TABLE 5. Outcomes of the suggested algorithms.

	1	2	3	4	5	6	7	8	9	10	11
User Base Request Size	3 MB	5 MB	10 MB	350 MB	700 MB	105 MB	140 MB	750 MB	1500 MB	2250 MB	3000 MB
Poor	1	1	1	0	0	0	0	0	0	0	0
Fair	0	0	0	1	1	0	0	0	0	0	0
Good	0	0	0	0	0	1	1	0	0	0	0
V.Good	0	0	0	0	0	0	0	1	1	0	0
Excellent	0	0	0	0	0	0	0	0	0	1	1
EC2 (VCPU)	1	2	4	8	16	32	48	64	96	128	192
Poor	1	1	1	0	0	0	0	0	0	0	0
Fair	0	0	0	1	1	0	0	0	0	0	0
Good	0	0	0	0	0	1	1	0	0	0	0
V.Good	0	0	0	0	0	0	0	1	1	0	0
Excellent	0	0	0	0	0	0	0	0	0	1	1
EC2 (RAM)	4	8	16	32	64	128	192	256	384	512	768
Poor	1	1	1	0	0	0	0	0	0	0	0
Fair	0	0	0	1	1	0	0	0	0	0	0
Good	0	0	0	0	0	1	1	0	0	0	0
V.Good	0	0	0	0	0	0	0	1	1	0	0
Excellent	0	0	0	0	0	0	0	0	0	1	1
EC2 (Storage)	1	2	4	8	12	16	24	32	48	64	88
Poor	1	1	1	0	0	0	0	0	0	0	0
Fair	0	0	0	1	1	0	0	0	0	0	0
Good	0	0	0	0	0	1	1	0	0	0	0
V.Good	0	0	0	0	0	0	0	1	1	0	0
Excellent	0	0	0	0	0	0	0	0	0	1	1
EC2(BW)	2	4	8	10	12	14	16	18	24	30	40
Poor	1	1	1	0	0	0	0	0	0	0	0
Fair	0	0	0	1	1	0	0	0	0	0	0
Good	0	0	0	0	0	1	1	0	0	0	0
V.Good	0	0	0	0	0	0	0	1	1	0	0
Excellent	0	0	0	0	0	0	0	0	0	1	1
Assignment	M6g.medium	M6g.large	M6g.xlarge	M5.2xlarge	M5.4xlarge	M6gd.8xlarg	M6gd.12xlarge	M6g.metal	M5d.metal	M6i.metal	M6a.metal

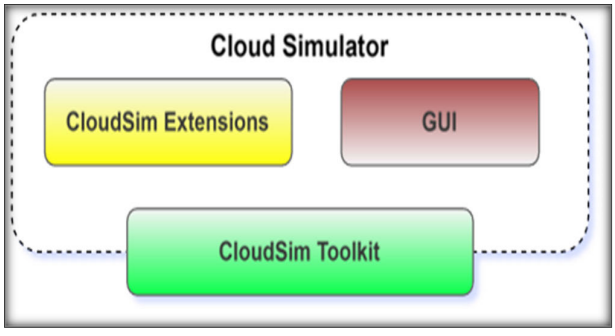

FIGURE 5. Cloud analyst model.

$(1/n)$, where n represents the number of processes in the queue. Thus, the wait time for any process is limited to $(n\text{-}1)$ times the quantum length, q, ensuring a fair and efficient distribution of processing time [29] [30].

e: STRATEGIES FOR SERVICE BROKERING

The role of a service broker is essential for determining the appropriate data center to satisfy customer needs and for orchestrating the data exchange between consumers and data centers [31]. This intermediary position enhances the connection between customers and cloud service providers [32]. Through the Service Broker Policy (SBP), services are dynamically distributed between the cloud's infrastructure and its service providers [33], effectively guiding the selection of data centers [31]. The assignment of virtual machines to physical hardware in data centers, a process critical to the data center broker known as virtual machine deployment,

underscores the importance of the SBP [34]. It is crucial to grasp the operational context of the SBP, particularly how it mediates between specific data centers and user demands. The SBP plays a pivotal role in identifying the most fitting data center to meet service expectations based on customer requests [31]. Our analysis involved adopting two foundational broker strategies and examining and contrasting their effectiveness [35]. The primary policy focuses on optimizing response time, where the service broker evaluates essential attributes of data centers to gauge their performance [22]. This approach ensures the quickest possible response times for end-users during queries [36]. In this routing strategy, the efficiency of data centers is continuously monitored, with preference given to directing traffic to the data center that offers the best response time, effectively managing direct bottlenecks [37]. Virtual machines are utilized to handle customer requests swiftly, enhancing point-to-point communication [38]. This policy assumes uniform processing requirements and execution times for all requests [39]. The secondary policy involves dynamic reconfiguration based on load, where the service broker manages scalability for cloud applications [22]. This involves the service broker dynamically reconfiguring and altering the virtual machines within data centers to match demand [36]. A cloud analyst facilitates the redistribution of loads across different data centers when the performance of the initial data center falls below a certain threshold [9]. This method calculates retention times to achieve the longest cycle time recorded, addressing both cost and performance expectations of users [39] and adjusting the number of virtual machines as needed [40].

IV. EXPERIMENTATION AND ANALYSIS

A. SIMULATION

To test our proposed policy, we deployed Cloud-Analyst with the optimize response time policy as part of an intelligent cloud broker validation process. This involved handling 1,000,000 user requests, allocated across ten user bases, and leveraging 31 individual AWS data centers spread across six geographic regions. Each data center operated with a single virtual machine, with configurations based on 11 real-life EC2 attributes as previously described. This setup allowed us to benchmark the performance against existing routing policies, notably the Reconfigure Dynamically with Load broker policy. Before initiating the simulations, we standardized the network delay metrics from AWS latency monitoring(https://www.cloudping.co/grid) (shown in Table 6) and set advanced data center configurations for all tests, as detailed below.

Table 7 displays data related to a Single User Base, which becomes pertinent in Table 8 as our research encompasses 11 analogous instances derived from this single-user base, varying according to the magnitude of user requests.

We employed Peak Hours (GMT) to depict the timing of user activity on AWS-Cloud. The number 60 is used to denote the number of requests per user within a one-hour simulation,

TABLE 6. Delay matrix.

Geographic -Regions	R0	R1	R2	R3	R4	R5
R0	3,27 ms	117,23 ms	94,24 ms	190,95 ms	227,74 ms	199,16 ms
R1	117,23 ms	2,63 ms	205,77 ms	299,86 ms	341,07 ms	312,32 ms
R2	94,24 ms	205,77 ms	4,99 ms	128,66 ms	155,91 ms	248,86 ms
R3	190,95 ms	299,86 ms	128,66 ms	3,51 ms	270,64 ms	153,24 ms
R4	227,74 ms	341,07 ms	155,91 ms	270,64 ms	8,1 ms	415 ms
R5	199,16 ms	312,32 ms	248,86 ms	153,24 ms	415 ms	4,42 ms

measured hourly (60.0). It's posited that the upper limit of users from each user base cluster during peak times is 100,000 average peak users, while the lower limit during off-peak periods is 10,000 average users. This is established using the following mathematical formula:

$$\text{Avgpeak users} = \frac{\textit{Total User Count}}{10\ \textit{UB}} \tag{11}$$

$$\text{Avg Off}-\text{peak users} = \frac{\textit{Avg Peak users}}{10} \tag{12}$$

The data size per request (in bytes) and the instruction length per request (in bytes) were determined by applying mathematical formulas No. 12 and No. 13, respectively. The "Grouping factor in data centers" refers to the capacity of a single application server instance to handle multiple requests concurrently. Similarly, the "User grouping factor in user bases" denotes the maximum number of users accessing services from a single user base simultaneously.

Additionally, a round-robin load-balancing strategy is employed to manage the distribution of workloads across virtual machines within a single data centre.

$$\text{Data size per request} = \frac{\text{Total UB request}}{\text{Avg peak users}} \tag{13}$$

$$\text{Executable length} = \frac{\textit{Total UB request}}{10\textit{UBs}} \tag{14}$$

Table 9 displays the foundational configuration for each of the 31 data centres featured in our research, which were deployed in 11 different scenarios adhering to the specifications of AWS General Purpose EC2 instances, as indicated in Table 10. The pricing below is based on data transferred "in" to and "out" of Amazon EC2. https://aws.amazon.com/ec2/pricing/on-demand/.

In our study, we contrasted the proposed Intelligent Validation Cloud Broker System (IVCBS) with traditional random allocation methods within the context of cloud resource management. Both approaches were evaluated under two distinct policies: optimizing response times and dynamically reconfiguring loads based on demand.

Traditional methods of allocating virtual machine (VM) resources typically distribute these resources to customer requests indiscriminately, using a random approach that does not account for the specific needs of the requests. Our

TABLE 7. Single-user base clusters.

Single-User Base Clusters	Geographic Regions	Requests per user per Hour	Peak Hours (GMT)		Avg peak users	Avg Off-peak users
			Start	End		
UB1	R0	60	12	15	100000	10000
UB2	R1	60	14	17	100000	10000
UB3	R2	60	19	22	100000	10000
UB4	R3	60	0	3	100000	10000
UB5	R4	60	20	23	100000	10000
UB6	R5	60	8	11	100000	10000
UB7	R0	60	12	15	100000	10000
UB8	R1	60	14	17	100000	10000
UB9	R2	60	19	22	100000	10000
UB10	R3	60	0	3	100000	10000

TABLE 8. (11-User base instances).

11-User Base Instances		Data size per request (Byte)	User grouping factor in User bases	Request Grouping factor in data centers	Executable Instruction length per request (byte)
EC2 instances	Count of User Base Clusters				
M6g.medium	10-UBs	30	100000	100000	300000
M6g.large	10-UBs	50	100000	100000	500000
M6g.xlarge	10-UBs	100	100000	100000	1000000
M5.2xlarge	10-UBs	350	100000	100000	3500000
M5.4xlarge	10-UBs	700	100000	100000	7000000
M6gd.8xlarg	10-UBs	1050	100000	100000	10500000
M6gd.12xlarge	10-UBs	1400	100000	100000	14000000
M6g.metal	10-UBs	7500	100000	100000	75000000
M5d.metal	10-UBs	15000	100000	100000	150000000
M6i.metal	10-UBs	22500	100000	100000	225000000
M6a.metal	10-UBs	30000	100000	100000	300000000

study provides a comprehensive description of these traditional allocation strategies in Table 11. It is critical to note that the specifications of the EC2 instances utilized in these traditional methods are identical to those employed in the Intelligent Validation Cloud Broker System (IVCBS) method, as detailed in previous tables and sections of our study. This strategic allocation is further illustrated by the general distribution of EC2 across 31 data centers, as depicted in our study. We apply this distribution in 11 different scenarios, tailored according to the number of user request sizes identified in this study.

B. RESULTS

1) IMPLEMENTATION OF IVCBS METHOD

Our analysis reveals that the Optimized Response Time Policy yields better outcomes than the Dynamic Reconfiguration with Load Policy in several key performance metrics: Average Overall Response Time, Average Data Center Processing Time, and Total Virtual Machine Cost. This suggests that the optimized policy more efficiently handles these aspects of cloud service management. However, the scenario shifts when examining Data Center Request Servicing Times, where the optimized policy either matches or slightly exceeds the times achieved by the dynamic reconfiguration policy. This indicates a nuanced trade-off between the two approaches in handling specific service demands.

TABLE 9. Fundamental data center.

31-AWS (DC- single instance)	Geographic Regions	Arch	OS	VMM	Data transfer cost	Physical HW-units
DC1	R0-N.virgina	X86	Linux	Xen	0,02	1
DC2	R0- Ohio	X86	Linux	Xen	0,02	1
DC3	R0-N.California	X86	Linux	Xen	0,02	1
DC4	R0- Oregon	X86	Linux	Xen	0,02	1
DC5	R0- Canada Central	X86	Linux	Xen	0,02	1
DC6	R0-Canada west(Calgary)	X86	Linux	Xen	0,02	1
DC7	R0-AWS GovCloud(US-East)	X86	Linux	Xen	0,02	1
DC8	R0-AWS GovCloud(US-West)	X86	Linux	Xen	0,02	1
DC9	R1- São Paulo	X86	Linux	Xen	0,02	1
DC10	R2- Frankfurt	X86	Linux	Xen	0,02	1
DC11	R2- Ireland	X86	Linux	Xen	0,02	1
DC12	R2- London	X86	Linux	Xen	0,02	1
DC13	R2- Milan	X86	Linux	Xen	0,02	1
DC14	R2- Paris	X86	Linux	Xen	0,02	1
DC15	R2- Spain	X86	Linux	Xen	0,02	1
DC16	R2- Stockholm	X86	Linux	Xen	0,02	1
DC17	R2- Zurich	X86	Linux	Xen	0,02	1
DC18	R3- HongKong	X86	Linux	Xen	0,02	1
DC19	R3- Hyderabad	X86	Linux	Xen	0,02	1
DC20	R3-Jakarta	X86	Linux	Xen	0,02	1
DC21	R3- Melbourne	X86	Linux	Xen	0,02	1
DC22	R3- Mumbai	X86	Linux	Xen	0,02	1
DC23	R3- Osaka	X86	Linux	Xen	0,02	1
DC24	R3- Seoul	X86	Linux	Xen	0,02	1
DC25	R3- Singapore	X86	Linux	Xen	0,02	1
DC26	R3- Sydney	X86	Linux	Xen	0,02	1
DC27	R3- Tokyo	X86	Linux	Xen	0,02	1
DC28	R4- Cape town	X86	Linux	Xen	0,02	1
DC29	R4- Bahrain	X86	Linux	Xen	0,02	1
DC30	R4- Israel	X86	Linux	Xen	0,02	1
DC31	R4- UAE	X86	Linux	Xen	0,02	1

TABLE 10. Data centers configurations according to EC2 class specifications.

11-AWS-EC2 Instances	Data Centers Utilized for Execution within the EC2 Class Specification		
	# of DCs	# of VM	VM policy
M6g.medium	31	1	Time-Shared
M6g.large	31	1	Time-Shared
M6g.xlarge	31	1	Time-Shared
M5.2xlarge	31	1	Time-Shared
M5.4xlarge	31	1	Time-Shared
M6gd.8xlarg	31	1	Time-Shared
M6gd.12xlarge	31	1	Time-Shared
M6g.metal	31	1	Time-Shared
M5d.metal	31	1	Time-Shared
M6i.metal	31	1	Time-Shared
M6a.metal	31	1	Time-Shared

TABLE 11. Arrangement of the EC2 instances in traditional methods.

31-AWS (DC- single instance)	Geographic Regions	EC2	Cost ($)	Physical HW-units
DC1	R0-N.virgina	M6g.medium	0.0385	1
DC2	R0- Ohio	M6g.xlarge	0.154	1
DC3	R0- N.California	M5.4xlarge	0.896	1
DC4	R0- Oregon	M6gd.12xlarge	2.1696	1
DC5	R0- Canada Central	M5d.metal	6.048	1
DC6	R0-Canada west(Calgary)	M6a.metal	8.3922	1
DC7	R0-AWS GovCloud(US-East)	M6g.large	0.0968	1
DC8	R0-AWS GovCloud(US-West)	M5.2xlarge	0.484	1
DC9	R1- São Paulo	M6gd.8xlarg	2.304	1
DC10	R2- Frankfurt	M6g.metal	2.944	1
DC11	R2- Ireland	M6i.metal	6.848	1
DC12	R2- London	M6g.medium	0.0444	1
DC13	R2- Milan	M6g.xlarge	0.1792	1
DC14	R2- Paris	M5.4xlarge	0.896	1
DC15	R2- Spain	M6gd.12xlarge	2.4192	1
DC16	R2- Stockholm	M5d.metal	5.76	1
DC17	R2- Zurich	M6a.metal	9.6878	1
DC18	R3- Hong Kong	M6g.large	0.106	1
DC19	R3- Hyderabad	M5.2xlarge	0.404	1
DC20	R3-Jakarta	M6gd.8xlarg	1.808	1
DC21	R3- Melbourne	M6g.metal	3.072	1
DC22	R3- Mumbai	M6i.metal	6.464	1
DC23	R3- Osaka	M6g.medium	0.0496	1
DC24	R3- Seoul	M6g.xlarge	0.188	1
DC25	R3- Singapore	M5.4xlarge	0.96	1
DC26	R3- Sydney	M6gd.12xlarge	2.736	1
DC27	R3- Tokyo	M5d.metal	7.008	1
DC28	R4- Cape town	M6a.metal	9.513	1
DC29	R4- Bahrain	M6g.large	0.094	1
DC30	R4- Israel	M5.2xlarge	0.449	1
DC31	R4- UAE	M6gd.8xlarg	1.7728	1

To provide a clear comparison, Table 12 showcases the results of implementing the IVCBS method with the Optimized Response Time Service Broker Policy, while Table 13 details the outcomes when applying the Dynamic Reconfiguration with Load Service Broker Policy.

The experiments were carried out across 31 Amazon data centers spanning 6 geographic regions. To capture data accurately during both peak and off-peak periods, 11 scenarios were implemented across 11 EC2 levels based on hourly intervals. Figure 6 displays the regional average response times to the 10 user bases, showcasing the effectiveness of IVCBS's Optimized Response Time Policy.

This approach uniformly distributes user requests across all data centers, regardless of geographic location, and consistently achieves lower response times compared to the Dynamic Reconfiguration Policy. Conversely, the outcomes of the Dynamic Reconfiguration Policy are detailed in Figure 7, which shows the regional average response times to the 10 user bases under the IVCBS method.

TABLE 12. Implementing IVCBS with optimize response time policy.

AWS-EC2	Overall Response Time (ms)	Data Center Processing (ms)	Total VM Cost ($)	Total Data Transfer Cost ($)
M6g.medium	2475,8	2373,38	83,29	$298,59
M6g.Large	3853,10	3740,25	167,24	497,65
M6g.Xlarge	14325,08	10798,69	334,48	1255,96
M5.2XLarge	140667,03	137632,98	853,50	3483,46
M5.4XLarge	1010570,86	1031103,10	1707,06	6963,47
M6gd.8XLarge	2151917,72	1947568,70	3140,37	9966,88
M6gd.12XLarge	3684599,83	3335444,58	4709,26	13114,84
M6g.metal	38334990,80	38234416,58	5351,62	25236,98
M5d.metal	79337433,27	79315311,43	12090,55	14482,63
M6i.metal	93529270,35	93372293,67	13730,36	6863,40
M6a.metal	94549552,26	94331238,90	17150,67	3320,20

This strategy routes user requests to data centers located in the same geographic region as the users, aiming to reduce latency. Despite the intuitive logic behind this approach, response times were generally higher than those achieved by the Optimized Response Time Policy, highlighting a key area where the latter excels.

The Average Data Center Request Servicing Time significantly influences energy consumption within cloud computing environments. Extended servicing times often reflect inefficient utilization of computing resources like processors and memory, which in turn can increase the energy load of operations.

This inefficiency not only affects the Power Usage Effectiveness (PUE) of data centers but also demands more extensive cooling solutions, a major contributor to energy consumption in these facilities. Additionally, the need to scale up resources to reduce servicing times can lead to over-provisioning, further elevating overall energy usage.

Enhancing the efficiency of request servicing times not only promotes more responsive cloud services but also helps in cutting down energy costs, thus supporting the broader goal of making cloud computing more energy-efficient and eco-friendly [41], [42].

Our observations highlight that the results of applying the Intelligent Validation Cloud Broker System (IVCBS) with an optimized response time policy significantly outperform those obtained through the dynamic reconfiguration policy.

This is evident from the comparative analysis of Figure 8 and Figure 9, which illustrate the superior performance of the optimized response time policy in managing Average Data Center Request Servicing Time, which leads to enhanced energy efficiency. Previously, the results demonstrated that systems using IVCBS with a dynamically reconfigured load-balancing broker policy, as shown in Figure 10, differ in performance from those using the Intelligent Validation Cloud Broker System (IVCBS) optimized for response times.

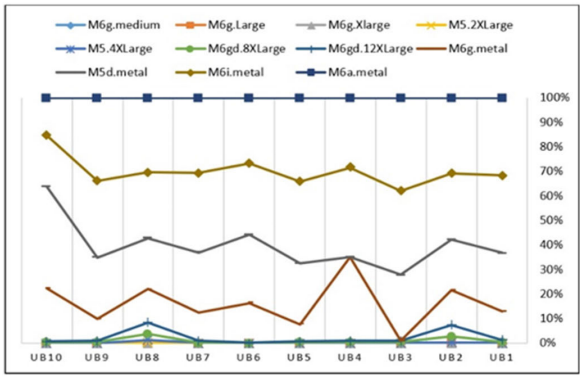

FIGURE 6. IVCBS-Response time by region (optimize response time policy).

TABLE 13. Implementing IVCBS with dynamic reconfiguration with load service broker policy.

AWS-EC2	Overall Response Time (ms)	Data Center Processing (ms)	Total VM Cost ($)	Total Data Transfer Cost ($)
M6g.medium	6353,58	6324,05	166,32	$298,59
M6g.Large	55390,42	55364	667,5	497,65
M6g.Xlarge	275390,88	270714,32	2666,54	1255,83
M5.2XLarge	2556092	2556270,05	8502,06	3483,45
M5.4XLarge	3252254,20	3255057,05	20401,48	6234,76
M6gd.8XLarge	3915809,21	3921022,05	43758,17	8915,92
M6gd.12XLarge	3573677,62	3584236,77	74944,34	11618,91
M6g.metal	37016372,94	37016688,54	95138,79	25828,65
M5d.metal	81818244,66	81883142,21	273382,89	14705,94
M6i.metal	93919067,50	93689019,40	379237,75	6796,75
M6a.metal	96334126,87	96128434,12	607000,72	3341,66

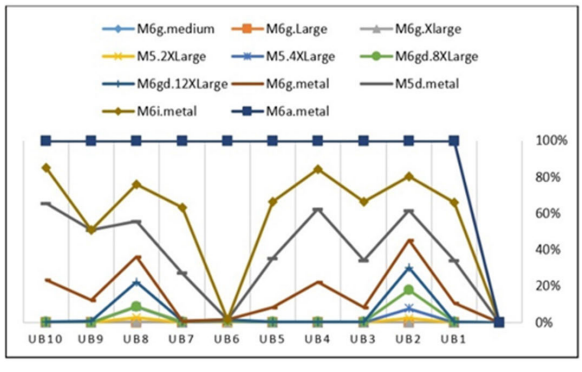

FIGURE 7. IVCBS-Response time by region (reconfigure dynamically policy).

As shown in Figure 11, This variance primarily stems from the dynamics of reconfiguration itself. The dynamic reconfiguration strategy routes user requests to data centers within the same geographic area as the users, often leading to increased processing delays. This occurs as requests queue up, awaiting

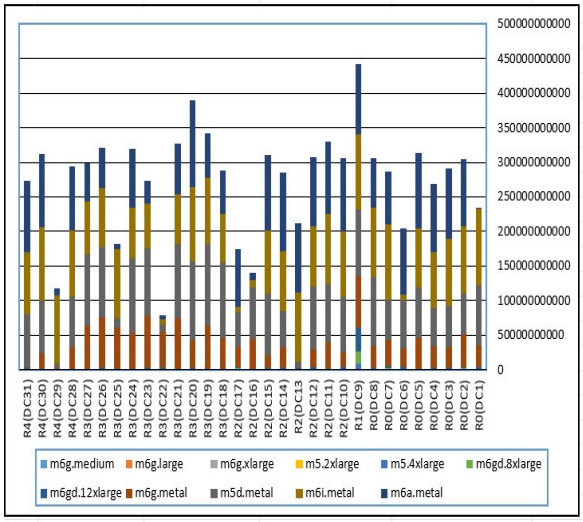

FIGURE 8. IVCBS DC- Request Servicing Time (optimize response time policy).

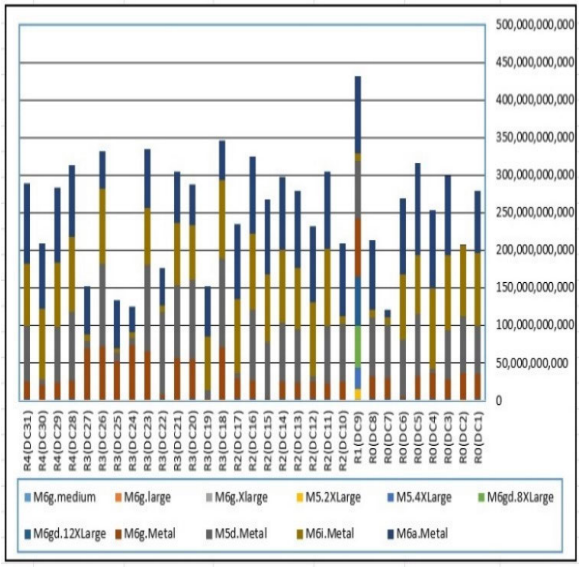

FIGURE 9. IVCBS DC- Request Servicing Time (dynamic reconfiguration policy).

available virtual machines for reconfiguration. Additionally, in some regions, having only one data center acts as a bottleneck, exacerbating delays during peak demand periods.

In contrast, the optimized response time policy excels by delivering superior round-trip times and more efficient processing. Moreover, our analysis is grounded in Amazon's real-world distribution of data center locations globally, utilizing eight virtual machines (VMs) in North America, one in South America, eight in Europe, ten in the Asia Pacific and Australia, and four in Africa and the Middle East.

This strategic distribution facilitates the IVCBS's ability to redirect user requests to data centers with appropriate VMs, optimized both for the characteristics of the user requests and for reduced processing times, energy consumption, and costs. For example, small user requests, defined in our study as 3 MB, are routed to VMs like the M6g.medium, while

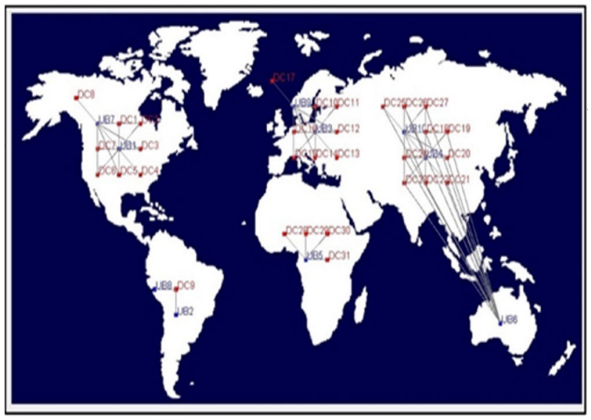

FIGURE 10. Routing strategy by the dynamic reconfigurations policy.

FIGURE 11. Routing strategy by the optimized response time policy.

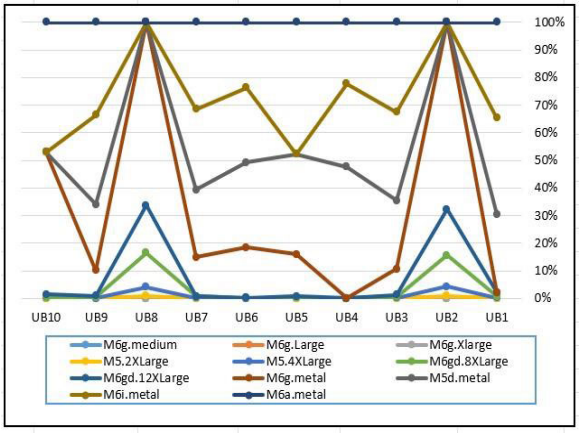

FIGURE 12. Traditional-Response time by region (optimize response time policy).

larger requests of 3 GB are directed to more robust machines like the M6a.metal.

2) IMPLEMENTATION OF TRADITIONAL METHODS

In this case, we applied the same 11 scenarios described earlier, using the EC2 distribution and allocation framework outlined in Table 9, which employs a traditional method. The outcomes of employing the optimized response time policy resulted in a higher average Overall Response Time, average

TABLE 14. Implementing traditional with optimize response time policy.

AWS-EC2	Overall Response Time (ms)	Data Center Processing (ms)	Total VM Cost ($)	Total Data Transfer Cost ($)
M6g.medium	2648,32	2544,20	5039,17	298,59
M6g.Large	3979,79	3866,43	5039,17	497,65
M6g.Xlarge	16565,20	16507,91	5039,17	995,31
M5.2XLarge	200877,44	206148,60	5039,17	3483,25
M5.4XLarge	1012024,16	1045751,95	5039,17	6965,51
M6gd.8XLarge	2784038,22	2523254,74	5039,17	9907,33
M6gd.12XLarge	4246474,38	3977103,11	5039,17	13054,04
M6g.metal	44420610,74	43609256,19	5039,17	17375,69
M5d.metal	80927473,71	80639117,03	5039,17	7093,73
M6i.metal	95412416,34	95769447,44	5039,17	3711,87
M6a.metal	97606171,17	98736234,17	5039,17	1686,10

TABLE 15. Implementing traditional with dynamic reconfiguration policy.

AWS-EC2	Overall Response Time (ms)	Data Center Processing (ms)	Total VM Cost ($)	Total Data Transfer Cost ($)
M6g.medium	2950,74	2918.84	137867,12	298,59
M6g.Large	4501,42	4481,36	137962,28	497,65
M6g.Xlarge	49465,79	49405,39	137677,42	995,31
M5.2XLarge	1275803,03	1276385,26	137762,59	3483,52
M5.4XLarge	3599233,17	3600108,32	137634,08	6234,08
M6gd.8XLarge	5282197,57	5322005,63	137742,44	8914,56
M6gd.12XLarge	7432190,15	7473084,39	137624,85	11566,42
M6g.metal	48005803,13	47769425,91	136059,33	14250,25
M5d.metal	84937790,73	85306107,68	134039,80	5810,42
M6i.metal	93010845,72	93028448,77	131046,97	3042,69
M6a.metal	91124687,42	90537061,27	124762,54	1462,37

Data Center Processing Time, and Total Virtual Machine Cost compared to those achieved with our proposed IVCBS method. However, we observed that the Total Data Transfer Cost was either less than or equal to that of the proposed IVCBS method. These findings are detailed in Table 14.

In evaluating the results from applying the dynamic reconfiguration policy with traditional methods, as detailed in Table 15, it is noted that the overall response time is broader than that achieved by the proposed IVCBS method in specific EC2 allocations (M5.4xlarge, m6gd.8xlarge, m6gd.12xlarge, m6g.metal, and m5d.metal). However, in all scenarios concerning the Total Data Transfer Cost, the traditional methods demonstrate lower costs compared to the IVCBS approach. As well as, Figure 12 displays the regional average response times for the 10 user bases, showcasing the performance of the traditional Optimized Response Time Policy. Meanwhile, Figure 13 provides a visualization of the regional average

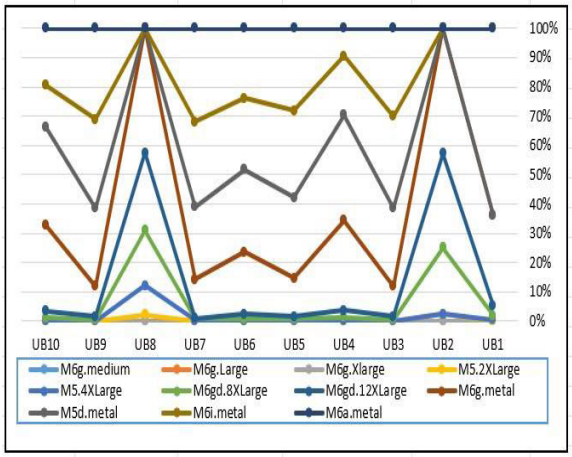

FIGURE 13. Traditional-Response time by region (reconfigure dynamically policy).

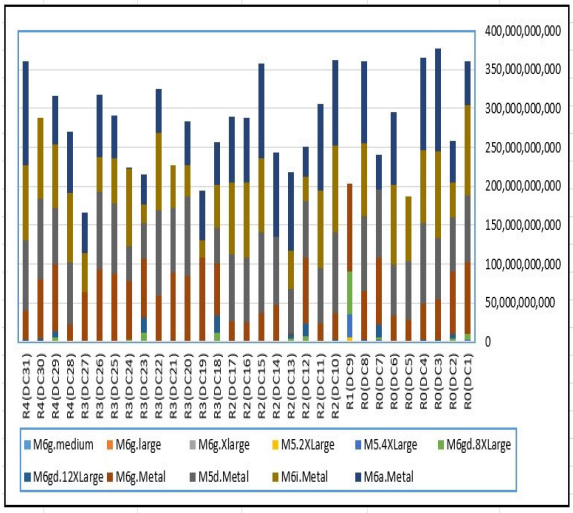

FIGURE 14. Traditional DC- Request Servicing Time (optimize response time policy).

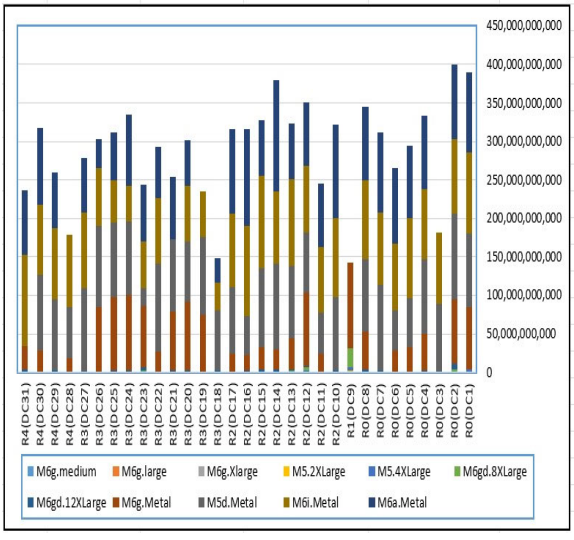

FIGURE 15. Traditional DC- Request Servicing Time (dynamic reconfiguration policy).

response times under the dynamic reconfiguration with load policy. Both figures highlight that these traditional methods were less effective compared to the results achieved by the proposed IVCBS method. Additionally, Figure 15 illustrates the outcomes when the traditional method incorporates the Dynamic Reconfiguration Policy. By comparing these findings with those from the proposed IVCBS method, it is evident that the IVCBS generally provides better Data Center Request Servicing

Times. This improvement significantly impacts energy efficiency in the computing environment, showcasing the advantages of the proposed method over conventional strategies. enhance the IVCBS's effectiveness, showcasing its potential to accommodate future growth in cloud systems while ensuring efficient and cost-effective user request processing within the cloud computing environment. Simultaneously, Figure 14 displays the results of the average Data Center Request Servicing Time across the 31 data centers in our study, applied in 11 different scenarios using the traditional Optimized Response Time Policy.

V. CONCLUSION

This research delves into crucial cloud computing aspects such as optimizing resource use during peak and off-peak periods, minimizing data processing and transfer times and costs and reducing the average response time from different geographical regions. A novel simulation was developed to improve cloud computing's response times by adjusting virtual machine (VM) attributes to match user request sizes and evenly distributing workloads as per Service Level Agreement (SLA) standards. This approach considers the current and future workloads and the available resources on each AWS-EC2 instance, aiming to distribute workloads across VM uniformly to ensure balanced system utilization and avoid over- or underutilization. A significant part of the study introduces the Intelligent Validation Cloud Broker System (IVCBS). Which enhances the proximity routing policy for data center selection by considering both VM attributes and the size of user requests. This modification allows for more efficient handling of variable request sizes, optimizing network delay, VM, and data transfer costs, and selecting data centers with minimal delay while considering real-time bandwidth, EC2 attribute diversity, and expected processing times. This refined approach improves upon traditional performance-optimized routing policies by including job size in its considerations, thereby achieving better response and processing times. Using the Cloud Analyst simulator for evaluation, the IVCBS demonstrated significant improvements over existing response and processing times policies. Adopting a Throttled load balancing policy could enhance the IVCBS's effectiveness, showcasing its potential to accommodate future growth in cloud systems while ensuring efficient and cost-effective user request processing within the cloud computing environment.

REFERENCES

[1] N. B. Ruparelia, *Cloud Computing*. Cambridge, MA, USA: MIT Press, 2023.
[2] M. N. Rao, *Cloud Computing*. New Delhi, India: PHI Learning Pvt. Ltd., 2015.

[3] C. Gong, J. Liu, Q. Zhang, H. Chen, and Z. Gong, "The characteristics of cloud computing," in *Proc. 39th Int. Conf. Parallel Process. Workshops*, Sep. 2010, pp. 275–279.
[4] P. Bharti, R. Ranjan, and B. Prasad, "Broker-based optimization of SLA negotiations in cloud computing," *Multiagent Grid Syst.*, vol. 17, no. 2, pp. 179–195, Aug. 2021.
[5] R. Nagarajan, P. Vinothiyalakshmi, and R. Thirunavukarasu, "An intelligent cloud broker with service ranking algorithm for validation and verification of cloud services in multi-cloud environment," Tech. Rep., 2023.
[6] R. Dilli, A. Argou, M. Pilla, A. M. Pernas, R. Reiser, and A. Yamin, "Fuzzy logic and MCDA in IoT resources classification," in *Proc. 33rd Annu. ACM Symp. Appl. Comput.*, Apr. 2018, pp. 761–766.
[7] A. Ghasemi, A. Toroghi Haghighat, and A. Keshavarzi, "Enhanced multi-objective virtual machine replacement in cloud data centers: Combinations of fuzzy logic with reinforcement learning and biogeography-based optimization algorithms," *Cluster Comput.*, vol. 26, no. 6, pp. 3855–3868, Dec. 2023.
[8] V. Mongia and A. Sharma, "Performance and resource-aware virtual machine selection using fuzzy in cloud environment," in *Progress in Advanced Computing and Intelligent Engineering*. Singapore: Springer, 2020, pp. 413–426.
[9] H. Singh, S. Tyagi, and P. Kumar, "Comparative analysis of various simulation tools used in a cloud environment for task-resource mapping," in *Proc. Int. Conf. Paradigms Comput., Commun. Data Sci. (PCCDS)*. Singapore: Springer, 2021, pp. 419–430.
[10] S. Mohanty, S. Patra, S. Sarkar, P. Dube, and P. K. Pattnaik, "Load balancing in cloud environment to minimize average response time," in *Proc. Int. Conf. Emerg. Syst. Intell. Comput. (ESIC)*, Feb. 2024, pp. 187–192.
[11] R. Garg, R. K. Sharma, D. Dalip, T. Singh, A. Malik, and S. Kumpsuprom, "Optimization of cloud services performance using static and dynamic load balancing algorithms," Tech. Rep., 2023.
[12] W. Zhao, Y. Peng, F. Xie, and Z. Dai, "Modeling and simulation of cloud computing: A review," in *Proc. IEEE Asia–Pacific Cloud Comput. Congr. (APCloudCC)*, Nov. 2012, pp. 20–24.
[13] F. Qazi, D. Kwak, F. G. Khan, F. Ali, and S. U. Khan, "Service level agreement in cloud computing: Taxonomy, prospects, and challenges," *Internet Things*, vol. 25, Apr. 2024, Art. no. 101126.
[14] S. S. Chauhan, E. S. Pilli, R. C. Joshi, G. Singh, and M. C. Govil, "Brokering in interconnected cloud computing environments: A survey," *J. Parallel Distrib. Comput.*, vol. 133, pp. 193–209, Nov. 2019.
[15] S. G. Ahmad, T. Iqbal, E. U. Munir, and N. Ramzan, "Cost optimization in cloud environment based on task deadline," *J. Cloud Comput.*, vol. 12, no. 1, p. 9, Jan. 2023.
[16] J. Yao, M. Yang, T. Deng, and H. Guan, "The cloud service broker in multicloud demand response," *IEEE Cloud Comput.*, vol. 5, no. 6, pp. 80–91, Nov. 2018.
[17] B. Cinar, "The role of cloud service brokers: Enhancing security and compliance in multi-cloud environments," *J. Eng. Res. Rep.*, vol. 25, no. 10, pp. 1–11, Oct. 2023.
[18] D. Petcu, "Portability and interoperability between clouds: Challenges and case study," in *Proc. Eur. Conf. Service-Based Internet*, Poznan, Poland. Berlin, Germany: Springer, Oct. 2011, pp. 62–74.
[19] Z. Chafai, H. Nacer, O. Lekadir, N. Gharbi, and L. Ouchaou, "A performance evaluation model for users' satisfaction in federated clouds," *Cluster Computing*, vol. 27, pp. 4983–5004, Jan. 2024.
[20] R. N. Calheiros, A. N. Toosi, C. Vecchiola, and R. Buyya, "A coordinator for scaling elastic applications across multiple clouds," *Future Gener. Comput. Syst.*, vol. 28, no. 8, pp. 1350–1362, Oct. 2012.
[21] S. Al-E'mari, Y. Sanjalawe, A. Al-Daraiseh, M. B. Taha, and M. Aladaileh, "Cloud datacenter selection using service broker policies: A survey," *Comput. Model. Eng. Sci.*, vol. 139, no. 1, pp. 1–41, 2024.
[22] A. I. El Karadawy, A. A. Mawgoud, and H. M. Rady, "An empirical analysis on load balancing and service broker techniques using cloud analyst simulator," in *Proc. Int. Conf. Innov. Trends Commun. Comput. Eng. (ITCE)*, Feb. 2020, pp. 27–32.
[23] S. N. M. Achhra, R. Shah, A. Tamrakar, P. K. Joshi, and P. S. Raksha, "Analysis OF service broker and load balancing in cloud computing," *Int. J. Current Eng. Sci. Res. (IJCESR)*, vol. 2, no. 4, pp. 92–98, 2015.
[24] A. Wittig and M. Wittig, *Amazon Web Services in Action: An in-Depth Guide to AWS*. New York, NY, USA: Simon and Schuster, 2023.
[25] S. Manvi and G. Shyam, "Cloud computing: Concepts and technologies. Boca Raton, FL, USA: CRC Press, 2021.
[26] L. A. Zadeh, G. J. Klir, and B. Yuan, *Fuzzy Sets, Fuzzy Logic, and Fuzzy Systems: Selected Papers*, vol. 6. Singapore: World Scientific, 1996.
[27] K. O. Oluborode, "Adaptive neuro-fuzzy controller for double lane traffic intersections," Doctoral dissertation, Federal Univ. Technol. Akure, Nigeria, 2021.
[28] A. Ahmed and A. S. Sabyasachi, "Cloud computing simulators: A detailed survey and future direction," in *Proc. IEEE Int. Advance Comput. Conf. (IACC)*, Feb. 2014, pp. 866–872.
[29] R. Srujana, Y. M. Roopa, and M. D. S. K. Mohan, "Sorted round Robin algorithm," in *Proc. 3rd Int. Conf. Trends Electron. Informat. (ICOEI)*, Apr. 2019, pp. 968–971.
[30] D. H. Youm, "Load balancing strategy using round Robin algorithm," *Asia–Pacific J. Convergent Res. Interchange*, vol. 2, no. 3, pp. 1–10, Sep. 2016.
[31] H. V. Patel and R. Patel, "Cloud analyst: An insight of service broker policy," *Int. J. Adv. Res. Comput. Commun. Eng.*, vol. 4, pp. 122–127, Jan. 2015.
[32] A. Gaur and K. Garg, "Survey paper on cloud computing with load balancing policy," *Int. J. Eng. Res.*, vol. 2, no. 7, 2015.
[33] D. G. Arseniev, L. Overmeyer, H. Kälviäinen, and B. Katalinić, Eds., *Cyber-Physical Systems and Control*, vol. 95. Cham, Switzerland: Springer, 2019.
[34] T. Puri, R. K. Challa, and N. K. Sehgal, "Energy-efficient delay-aware preemptive variable-length time slot allocation scheme for WBASN (edpvt)," in *Proc. 2nd Int. Conf. Commun., Comput. Netw. (ICCCN)*, NITTTR Chandigarh, India, Singapore: Springer, 2019, pp. 183–194.
[35] Z. Benlalia, A. Beni-Hssane, K. Abouelmehdi, and A. Ezati, "A new service broker algorithm optimizing the cost and response time for cloud computing," *Proc. Comput. Sci.*, vol. 151, pp. 992–997, Jan. 2019.
[36] M. Radi, "Efficient service broker policy for large-scale cloud environments," 2015, *arXiv:1503.03460*.
[37] M. R. Mesbahi, M. Hashemi, and A. M. Rahmani, "Performance evaluation and analysis of load balancing algorithms in cloud computing environments," in *Proc. 2nd Int. Conf. Web Res. (ICWR)*, Apr. 2016, pp. 145–151.
[38] K. M. Khalil, M. Abdel-Aziz, T. T. Nazmy, and A. B. M. Salem, "Cloud simulators—An evaluation study," *Int. J. Inf. Models Analyses*, vol. 6, no. 1, 2017.
[39] S. Nayak and P. Patel, "Analytical study for throttled and proposed throttled algorithm of load balancing in cloud computing using cloud analyst," *Int. J. Sci. Technol. Eng.*, vol. 1, no. 12, pp. 90–100, 2015.
[40] K. Bahwaireth, L. Tawalbeh, E. Benkhelifa, Y. Jararweh, and M. A. Tawalbeh, "Experimental comparison of simulation tools for efficient cloud and mobile cloud computing applications," *EURASIP J. Inf. Secur.*, vol. 2016, no. 1, pp. 1–14, Dec. 2016.
[41] S. Mondal, F. B. Faruk, D. Rajbongshi, M. M. K. Efaz, and M. M. Islam, "GEECO: Green data centers for energy optimization and carbon footprint reduction," *Sustainability*, vol. 15, no. 21, p. 15249, Oct. 2023.
[42] J. Liu, L. Yan, C. Yan, Y. Qiu, C. Jiang, Y. Li, Y. Li, and C. Cérin, "Escope: An energy efficiency simulator for internet data centers," *Energies*, vol. 16, no. 7, p. 3187, Mar. 2023.

IHAB SEKH was born in Iraq, in 1982. He received the M.Sc. degree in computer science from Osmania University, Hyderabad, India, in 2015. He is currently pursuing the Ph.D. degree with the Institute of Information Technology, University of Miskolc, Hungary. Since 2006, he has been a Teacher. He was an Assistant Teacher with the Department of Computer Science, Southern Technical University, Iraq.

KÁROLY NEHÉZ was born in Nyiregyhaza, in 1974. He received the M.Sc. and Ph.D. degrees in mechanical engineering from the University of Miskolc, in 1997 and 2003, respectively. His Ph.D. thesis was "Computer Simulation and Optimization Issues of Milling." Since 2000, he has been a Teaching Assistant with the Department of Applied Informatics, where he has been an Assistant Professor, since 2007, and an University Associate Professor, since 2010. Since 2017, he has been the Head of the Department of Applied Informatics. He teaches IT systems design. His main research interests include software development, IT systems, and data mining.