University of Miskolc

FACULTY OF MECHANICAL ENGINEERING AND

INFORMATICS

Adaptive Fuzzy Logic Models for Efficient Cloud Service Management and SLA Optimization

PhD DISSERTATION

AUTHOR:

Ihab Razzaq Sekhi
MSc in Information Science

József Hatvany Doctoral School of

Information Science, Engineering and Technology

HEAD OF DOCTORAL SCHOOL

Prof. Dr. László Kovács

ACADEMIC SUPERVISORS

Prof. Dr. Szilvester Kovacs
Dr. Károly Nehéz

Miskolc
2025

Declaration of Authorship

The author hereby declares that this dissertation has not been submitted, either in the same or in a different form, to this or to any other university for obtaining a PhD degree. The author confirms that the submitted work is his own and the appropriate credit has been given where reference has been addressed to the work of others.

Author's declaration

I, the undersigned, Ihab Razzaq Sekhi, affirm that I have independently prepared this doctoral dissertation utilizing solely the sources provided. All content borrowed from external sources, whether quoted verbatim or paraphrased, has been clearly cited with appropriate references.

March 22, 2025.

Ihab Razzaq Sekhi

Acknowledgments

First and foremost, I express my sincere gratitude to Allah, the Almighty, for granting me countless blessings, wisdom, and inspiration, enabling me to complete this dissertation.

I also extend my heartfelt thanks to the University of Miskolc, Faculty of Mechanical Engineering and Informatics, for providing me with the opportunity to pursue a PhD in information technology.

While this dissertation represents the culmination of my efforts, its success is also the result of invaluable guidance and encouragement from many individuals. It would not have been possible without the unwavering support of those around me and the dedication and hard work invested over the past four years.

I am particularly grateful to my supervisors, Professor Szilveszter Kovacs and Dr. Károly Nehéz, for their continued support, direction, and encouragement throughout this journey. This dissertation would not have come to fruition without their expert guidance. I would also like to thank everyone in the Computer Science Department for their contributions.

In addition, I am thankful to my department colleagues for their assistance and support, especially in organizing events related to my doctoral studies.

Finally, I would like to express my profound gratitude to my family, particularly my parents and siblings, whose unwavering support and encouragement have been instrumental in helping me achieve this milestone.

Ihab Razzaq Sekhi

Table of Contents

List of Abbreviations

XaaS	Everything as a Service
SLA	Service Level Agreement
QoS	Quality of Service
RTT	Round-Trip Time
VM	Virtual Machine
RLBGD	Rank-based Load Balancing in Geo-Distributed
FIS	Fuzzy Inference System
CSP	Cloud Service Provider
IVCBS	Intelligent Validation Cloud Broker System
EC2	Elastic Compute Cloud
MEC	Multi-Access Edge Computing
AWS	Amazon Web Services
GCP	Google Cloud Platform
AI	Artificial Intelligence
SaaS	Software as a Service
PaaS	Platform as a Service
IaaS	Infrastructure as a Service
NIST	National Institute of Standards and Technology
API	Application programming interface
VPN	Virtual Private Network
CRM	Customer Relationship Management
ERP	Enterprise Resource Planning
IAM	Identity and Access Management
WAN	Wide area network
PDA	Personal digital assistant
CAF	Cloud Adoption Framework
DCN	Data center network
SLO	Service Level Objectives
EBS	Elastic Block Store
RDS	Relational Database Service
VPC	Virtual Private Cloud
ACL	Access Control Lists
AKS	Azure Kubernetes Service
ML	Machine Learning
BGP	Border Gateway Protocol
QoE	Quality of Experience
MTTF	Mean-Time-To-Failure
MTTR	Mean-Time-To-Recovery
BW	Bandwidth
BTC	bulk transfer capacity
UDP	User Datagram Protocol
TCP	Transmission Control Protocol
ms	milliseconds
ISP	Internet service provider
MF	Membership Function
COG	Center of Gravity

CSU	Cloud service users
PM	Physical machines
PRSF	Performance and Resource-Aware Virtual Machine Selection using Fuzzy
COTD	Cost Optimization based on Task Deadline
ESCE	Equally Spread Current Execution
LB	load balancing
RR	Round Robin algorithm
SBP	Service Broker Policy

List of Figures

List of Tables

Chapter 1 General Introduction

Cloud computing is a transformative technology that provides seamless access to a wide range of computing resources—including applications, servers, storage, and networks—without requiring an upfront investment. This technology supports substantial scalability, allowing users to pay only for the resources they utilize, which makes it highly adaptable to diverse needs. Cloud services, collectively known as "XaaS", facilitate data-driven decision-making, significantly enhancing productivity and customer service. Cloud computing effectively bridges the gap between client expectations and service delivery by offering internet-based services that improve collaboration, ease of access, and security [1]. (SLAs) are fundamental in defining the relationship between service providers and users by establishing the terms of service and quality expectations. SLAs also hold vendors accountable for non-compliance. As cloud computing adoption continues to grow, the importance of SLAs has increased, demanding robust guarantees for availability, uptime, and downtime. Effective SLAs go beyond mere contractual obligations; they are crucial for fostering trust between providers and clients, essential for sustainable success. Consequently, research has focused on developing SLA methodologies that enhance Quality of Service (QoS) and build customer trust, recognizing their significance in managing complex business relationships and shaping modern business practices [2][3]. Evaluating performance in cloud environments is complex due to the components involved, ranging from concrete elements like communication links to abstract ones like packets and protocols. Researchers and engineers must design a comprehensive performance evaluation plan to obtain meaningful results and answer critical questions. Such a plan should clearly define the objectives for assessing the system's performance and identify specific metrics to measure, such as RTT and response time, to provide actionable insights [4]. SLA-oriented resource allocation in cloud computing involves several key components: brokers, SLA resource allocators, (VMs), and PM. Users interact with cloud management systems through brokers, enabling dynamic resource allocation and concurrently operating multiple applications on a single machine. Data centers, composed of numerous servers and networks that function as transmission media for resources, form the backbone of cloud infrastructure. Despite these advanced capabilities, resource availability and privacy remain persistent concerns. Effective LB is crucial for enhancing service quality and optimizing resource utilization. Service brokers select the most appropriate geo-distributed data centers based on transmission delay, network delay, processing time, workload, and cost. The Datacenters (RLBGD) method employs a weighted combination of these criteria for optimization, ensuring efficient cloud resource management [5][6]. Fuzzy logic is a mathematical framework that handles uncertainty and imprecision by enabling approximate reasoning rather than fixed binary logic. Unlike traditional binary systems, where variables are strictly defined as true or false, fuzzy logic allows variables to have truth values between 0 and 1. This approach is beneficial for modeling complex systems where binary logic falls short. Based on fuzzy set theory, fuzzy computing simulates the human brain's nonlinear and imprecise information processing capabilities. It is widely applied in fields like (FIS), often in combination with other AI methods. This approach enables more precise and scientific consumer preference designs by reducing ambiguity through the fuzzy comprehensive evaluation method [7][8]. This thesis introduces several innovative approaches using fuzzy logic-based systems and algorithms to enhance SLA management, VM allocation, and decision-making in cloud computing environments. The study first presents the estimating Cloud Computing RTT Using Fuzzy Logic for Inter-Region Distances, a novel approach for estimating RTT in Amazon cloud environments. This method uses fuzzy logic to account for inter-region distances, providing a nuanced understanding of network latency by categorizing proximity and time and employing both ping tests and mathematical methods for accurate RTT calculation. Additionally, the thesis explores Selecting the SLA Guarantee by Evaluating the QoS Availability, which develops an intelligent SLA guarantee model using fuzzy theory. This model calculates SLA values for CSPs by evaluating specific computing and networking parameters and transforming data to manage ambiguity. The proposed fuzzy logic system classifies SLAs into 9 levels (ranging from 90% to 99.999%) based on QoS availability metrics, including computing (uptime and downtime) and networking (BW, jitter, RTT, and packet loss). The primary objectives are to develop a versatile SLA model that diverges from typical CSP offerings and improve SLA categorization's precision, tailored to user-specific requirements. The work Enhancing Decision-Making in Uncertain Domains through Optimized Fuzzy Logic Systems proposes optimizing fuzzy logic systems by reducing fuzzy rules and improving decision-making accuracy. The study introduces flexible mathematical modeling to minimize time and cost while enhancing precision in fuzzy decision-making processes for classification and scheduling. A comparative analysis shows the advantage of this approach over traditional methods by employing three distinct membership functions (Triangular, Trapezoidal, and Gaussian), enhancing flexibility and accuracy in determining overlapping membership degrees. Another essential contribution is the Efficient Broker-Driven Request Packet Size approach, which introduces a broker-driven model using fuzzy logic for dynamic VM allocation based on request packet size. This method optimizes resource usage, reduces latency, and improves system performance. Compared to traditional techniques, simulations using data from Google Cloud Platform's Europe West3 region demonstrated significant improvements in response time, data center processing, request serving time, and data transfer costs. Furthermore, the thesis presents the (IVCBS), which leverages an algorithm for dynamic VM allocation and intelligent SLA selection. The proposed algorithm utilizes a mathematical model that replicates the behavior of trapezoidal membership functions to compute continuous membership degrees (ranging from 0 to 1) for various parameters, such as VM attributes and user request sizes. These continuous values represent the degree to which inputs belong to linguistic categories (e.g., Poor, Fair, Excellent). In a subsequent decision stage, these membership degrees are transformed into binary scores (1 or 0) using predefined thresholds to streamline real-time resource allocation processes. Thus, while final allocation decisions are based on binary validation, the underlying fuzzy classification operates with continuous values. Tested across 31 AWS data centers worldwide with 11 EC2 types, IVCBS optimizes response time, improves processing efficiency, reduces VM and transfer costs, and enhances power efficiency while maintaining high QoS in cloud environments. Various tools and environments, including CloudAnalyst [9] and MATLAB, were utilized to conduct these studies. Lastly, the study proposes the Reliable and Cost-Effective Fuzzy-based Cloud Broker technique, which assists users in selecting suitable cloud service instances by evaluating user needs and service characteristics. This technique analyzes various scenarios, including static and mobile users, to assess the impact of user mobility on service quality and optimize cloud service management. The work emphasizes the necessity of cloud brokerage services as intermediaries, balancing user needs with service provider interests. The Edge CloudSim simulator [10] implemented the proposed cloud broker on the (MEC) paradigm. This choice was made because services running on the virtualized edge are more sensitive to delay, and the broker's selection of the appropriate service instance significantly impacts such settings. In this scenario, different data centers belonging to AWS, Google Cloud (GC), and Azure Cloud Services (AZURE) were placed in different regions.

1.1 Problem statement

Cloud computing, a cornerstone of modern IT, offers scalable, flexible, and on-demand access to computing resources through various service models governed by (SLAs), formal contracts between a (CSP) and a customer that define the specific level of service the provider guarantees to deliver. However, challenges such as compliance mechanisms by (CSPs), provider lock-in, and the proliferation of CSPs create complexity for users. Inconsistencies in promised Quality of Service (QoS) levels also complicate the decision-making process, leading to inefficiencies and suboptimal outcomes. As cloud data centers scale, energy consumption becomes a critical concern, making energy efficiency a vital aspect of cloud service management. Balancing energy consumption with QoS metrics is crucial for delivering sustainable and efficient cloud services that meet diverse user requirements [11][12]. By addressing these challenges, we can pave the way for more efficient and reliable cloud services, a key goal of this research. This will enhance the user experience and the overall performance of cloud computing. The complexity of cloud computing is amplified by factors such as the physical distance between data centers, which significantly impacts performance and RTT for data transmission. As IT services increasingly migrate to cloud infrastructures, monitoring network performance becomes essential for ensuring optimal service delivery. However, (CSPs) typically provide only qualitative information on network performance, resulting in uncertainties and suboptimal deployment decisions. To address these challenges, it is crucial to focus on cloud-to-user latency and the network paths connecting data centers to globally distributed users. Furthermore, managing distributed transactions in cloud environments involves balancing reliability and consistency, particularly in the face of hardware failures, network outages, and varying latencies. Analyzing these factors can lead to more informed strategies for cloud service deployment and optimization [13][14]. Given the current state of cloud service management, there is an urgent need for more intelligent and adaptive strategies. These strategies should focus on managing Service SLA selection and resource allocation in cloud environments. Their goal should be to optimize response times, reduce latency, and ensure service reliability. A compelling resource management strategy can enable cloud providers to lower energy consumption and minimize SLA violations within data centers, thus enhancing overall service efficiency and sustainability. Moreover, such a strategy can incorporate predictive models that anticipate future resource demands, prevent resource shortages, and dynamically scale resources in response to changing workloads, ensuring optimal performance and resource utilization [15][16]. Traditional approaches to managing cloud service environments often rely on extensive rule-based systems that are computationally intensive and lack the flexibility needed to adapt to these environments' diverse and dynamic nature [17].

Challenges such as data migration, resource allocation, and competition among providers can significantly limit the capabilities of cloud computing environments. Similarly, in (AI), decision-making in uncertain and ambiguous real-world scenarios presents substantial complexities. Fuzzy logic systems have proven valuable tools in these contexts, offering a means to approximate optimal decisions by effectively handling uncertainty and vagueness [18]. While fuzzy logic is a valuable method for modelling computer knowledge, traditional approaches have their limitations. These approaches rely extensively on significant rule sets to determine the degree of membership for elements within a fuzzy set. This reliance results in considerable computational overhead and limits the scalability of such systems, posing challenges to their efficient implementation in complex environments [19]. Efficient allocation of (VMs) is essential for optimizing resource utilization in cloud environments. However, traditional VM allocation methods often face challenges in managing dynamic workloads, leading to suboptimal performance and increased operational costs. Resource management, particularly with a focus on CPU resource utilization, is a complex task that requires advanced strategies to enhance efficiency and reduce overall costs [20]. As cloud computing environments expand in scale and complexity, there is an increasing need for adaptive and efficient resource allocation strategies capable of dynamically responding to varying demand patterns in real time. Such a strategy must optimize resource utilization while maintaining low latency and fast execution times for real-time applications and interactive services (AI) is increasingly being leveraged to automatically manage and optimize cloud resources, addressing challenges such as real-time performance requirements and energy efficiency concerns. The effectiveness of these methods can be further enhanced by incorporating advanced AI models and developing innovative solutions to address emerging challenges in distributed and heterogeneous cloud environments [16]. To address the intertwined challenges of optimizing cloud service delivery, there is a pressing need for innovative cloud brokerage systems that utilize advanced techniques such as fuzzy logic and intelligent algorithms. These systems can act as intermediaries between users and (CSPs), enabling more accurate and efficient service selection by accounting for user requirements and different CSPs' diverse characteristics. Additionally, to tackle environmental and operational concerns, future generations of cloud computing must focus on becoming more energy-efficient and sustainable while maintaining the delivery of high-quality services. This is a crucial direction for the future of cloud computing [21]. In conclusion, cloud computing services' rapid growth and complexity necessitate developing reliable, adaptive, and cost-effective cloud brokerage solutions. These systems improve decision-making accuracy, optimize SLA selection, and manage workload distribution, preventing data center overload and minimizing costs [22].

1.2 The objectives of the thesis

I. Estimating RTT in cloud computing environments using fuzzy logic to account for inter-region distances, providing a nuanced understanding of network latency by categorizing proximity and time, and employing two techniques a ping test and a mathematical approach—for accurate RTT calculation.

II. To develop an intelligent fuzzy theory-based SLA guarantee model that calculates the SLA guarantee value for each CSP by considering specific computing and networking parameters, using fuzzy logic to handle and transform data to address ambiguity in results.

III. This research aims to push the boundaries of cloud computing by improving the precision and accuracy of fuzzy decision-making processes and non-probabilistic models. I propose an innovative approach to flexible mathematical modeling that minimizes time and cost while eliminating the need for extensive fuzzy rules. This approach promises to revolutionize the efficiency of cloud computing environments.

IV. To develop the (IVCBS) using a fuzzy logic-based algorithm aligned with the trapezoidal membership function to optimize (VM) allocation dynamically, enhance response times, improve data center processing efficiency, reduce VM and data transfer costs, and achieve power efficiency, thereby addressing scalability and performance challenges while maintaining high Quality of Service (QoS) in cloud computing environments.

V. Our research is dedicated to developing a broker-driven approach using a fuzzy logic system for the dynamic optimization of (VM) allocation in cloud computing environments. Based on request packet size, this approach promises to optimize resource usage, reduce latency, enhance overall system performance, and improve response times, data center processing times, request serving times, and data transfer costs. This approach will significantly contribute to the efficient management of cloud resources.

VI. To develop a fuzzy logic-based cloud brokerage technique to assist users in selecting the most suitable cloud service instances by evaluating factors like user needs and service characteristics. The study aims to enhance decision-making processes for cloud service selection by analyzing multiple scenarios, including static and mobile users, to assess the impact of user mobility on service quality and explore the effects of implementing a brokerage service that supports service migration, optimizing cloud service management in dynamic environments.

1.3 Dissertation Structure and Organization

The remaining structure of the dissertation is organized as follows:

- **Chapter 2**: Provides an in-depth understanding of cloud service models (IaaS, PaaS, SaaS) and deployment models, discussing their importance for informed decision-making regarding customization, control, and scalability. It also introduces the NIST Cloud Computing Reference Architecture and essential characteristics of cloud computing systems.
- **Chapter 3**: Explores the driving factors behind cloud adoption, emphasizing strategic, operational, and financial aspects. It discusses (CAFs), core business benefits (agility, adaptability, security), and financial advantages (cost savings, economies of scale). Focuses on best practices for successful cloud adoption, including governance, migration, and security. It highlights the benefits of cloud platforms, such as agility, business continuity, and economic advantages, alongside the importance of security.

- **Chapter 4**: Discusses the estimation of RTT in cloud computing environments using fuzzy logic, focusing on challenges like geographical distance, network congestion, and routing policies, with a case study on AWS demonstrating improved RTT estimation.
- **Chapter 5**: Introduces a fuzzy logic-based SLA classification model, categorizing SLAs into 9 levels based on key QoS metrics such as uptime, BW, jitter, and RTT, offering a flexible, transparent, and user-friendly method for improved SLA selection.
- **Chapter 6**: Examines the optimization of fuzzy logic systems for decision-making in uncertain environments, presenting a mathematical model using various membership functions to categorize input data, with comparisons to traditional FIS demonstrating improved performance.
- **Chapter 7**: Discusses the Intelligent SLA Selection through the Validation Cloud Broker System (IVCBS), focusing on improving cloud computing efficiency through optimization algorithms and simulations that show IVCBS outperforms traditional methods in response time, processing, and cost reduction.
- **Chapter 8**: Explores a broker-driven approach to (VM) allocation, using fuzzy logic to dynamically adjust resource distribution based on request packet sizes. The study demonstrates improved performance and cost efficiency through Cloud Analyst simulations.
- **Chapter 9**: Presents the design of a fuzzy logic-based cloud broker system that balances CSP and customer interests by ranking service instances and users. It optimizes service quality and cost through service migration and mobility considerations, with simulations showing superior stability, service delay, and cost-effectiveness compared to other methods.
- **Chapter 10:** presents a comprehensive conclusion of all contributions, outlining three key theses under the section "New Scientific Results," which constitute the primary objectives of this dissertation.

Chapter 2 Cloud Computing

Provides an overview of cloud computing service models, deployment types, and key characteristics. It explains the differences between cloud serivec models, helping users choose the right model for their specific needs. The chapter introduces the NIST Cloud Computing Reference Architecture and examines public, private, community, and hybrid deployments, discussing trade-offs in control, security, cost, and scalability. It also highlights the main benefits of cloud computing, including on-demand self-service, broad network access, and resource pooling.

2.1 Cloud Computing Service Models and Offerings

Choosing the right service model is a critical factor for the successful delivery of cloud-based solutions. To make an informed choice, it is essential to understand each service model and the division of responsibilities between the CSP and the cloud service consumer [23]. Cloud service models include (SaaS), (PaaS), and (IaaS). SaaS operates on top of PaaS, which, in turn, runs on IaaS. In recent years, the number of SaaS offerings has grown significantly, making it challenging for consumers to select the best service among those with similar functionalities [24]. Each cloud service model provides different levels of customization and ownership, depending on the user's needs—ranging from raw computing power to fully developed software solutions. The separation of responsibilities and customization options between the models varies, offering flexibility to users based on their requirements . Appendix 1 (Figure 1) provides an overview of the NIST Cloud Computing Reference Architecture, which identifies the key actors, their activities, and functions in cloud computing. This high-level diagram is designed to help users understand the requirements, uses, characteristics, and standards of cloud computing [25][26]. Three cloud service models offer abstraction levels to simplify system building and deployment [25].

2.2.1 Infrastructure as a Service (IaaS)

IaaS provides virtualized computing resources over the internet, letting users manage servers, storage, and networking without handling physical hardware. It enables rapid provisioning through APIs or web consoles, offering flexibility and cost efficiency.

Key Offerings:

- Compute: VMs, containers, bare metal.
- Storage: Block, object, file.
- Networking: Virtual networks, load balancers, VPNs.

Benefits:

- Full infrastructure control.
- Scalable resources.
- Pay-as-you-go pricing.

Examples: AWS EC2, GCP, Azure VMs.

2.1.2 Platform as a Service (PaaS)

PaaS provides a cloud-based platform for developers to build, run, and manage applications without handling underlying infrastructure. According to NIST, PaaS enables users to deploy applications created with supported languages and tools while the provider manages the underlying networks, servers, and storage. PaaS allows developers to focus on coding and app management, thereby simplifying the development process and accelerating deployment.

PaaS Offerings:

- Development frameworks (e.g., Java, Python, Node.js).
- Application hosting.
- Database services (e.g., MySQL, NoSQL).
- Middleware for messaging and integration.

Benefits:

- Simplifies development.
- Integrated tools streamline workflows.
- Faster time-to-market.

Examples: Google App Engine, Heroku, Azure App Service.

2.1.3 Software as a Service (SaaS)

SaaS delivers ready-to-use software over the internet, accessible via web browsers. Users handle only app settings, while providers manage infrastructure and updates.

Offerings:

- Business apps (e.g., CRM, collaboration).
- Industry-specific tools.
- Data analytics.

Benefits:

- No infrastructure management.
- Automatic updates.
- Subscription pricing.

Examples include Dropbox, Slack, Zoom, and Google Workspace.

2.2 Cloud Deployment Models

Cloud deployment models define how clouds are built, owned, and used, impacting security, cost, and accessibility. NIST identifies four types: public, private, community, and hybrid clouds. Each model varies in infrastructure location, control, and suitability for different organizational needs, offering unique benefits and costs [27].

2.2.1 public cloud

A **public cloud** is a model of cloud computing where services, such as storage, computing power, and applications, are provided by third-party vendors over the Internet. These services are shared among multiple customers, but each user's data and applications remain isolated and secure within their environment.

2.2.1.1 Technical Architecture

- **Shared Resources:** Virtualized infrastructure allows multiple tenants to share and manage resources via web browsers.
- **Elasticity:** Resources scale instantly to handle fluctuating demand for computing, storage, and BW.
- **Network Accessibility:** Infrastructure is accessible online through secure connections, such as VPNs, reducing the need for on-premises management.
- **API Access:** RESTful APIs enable programmatic control, service integration, and support assistive technologies.

- **Self-Service:** Users can provision and manage resources independently via web portals, supporting unlimited scalability.

2.2.1.2 Operational Considerations

- **Cost:** Pay-per-use pricing helps organizations optimize costs by paying only for the resources they use.
- **Security:** Users must secure data and apps with encryption, IAM, and compliance practices supported by audits and monitoring.
- **Performance:** Distributed data centers reduce latency and boost performance by placing services closer to users.

2.2.2 Private Cloud

A private cloud is dedicated to a single organization, offering greater control, security, and customization compared to public clouds, where resources are shared.

2.2.2.1 Technical Architecture

- **Single-Tenant Environment:** Dedicated to one organization, ensuring control, security, and customizable resources.
- **Customization:** Allows for tailored server, software, and security configurations to meet specialized needs.
- **Infrastructure:** Can be on-premises or off-premises, using platforms like VMware, OpenStack, or Hyper-V for resource management.
- **Automation:** Utilizes tools such as Kubernetes and OpenShift to automate provisioning, scaling, and operations.

2.2.2.2 Technical Operational Considerations

- **Control:** Offers complete flexibility and control over security, performance, and infrastructure, tailored to business needs.
- **Security:** Ensures strong protection in a dedicated environment through robust protocols, firewalls, and encryption.
- **Compliance:** Meets regulatory and industry standards, which are crucial for sectors such as finance, healthcare, and government.
- **Cost:** Higher upfront costs require careful management to achieve long-term cost efficiency.

2.2.3 Hybrid Cloud

A hybrid cloud combines public and private clouds, enabling sensitive data to remain private while utilizing public clouds for less critical workloads. The environments remain separate but integrated, offering flexibility and tailored solutions through the addition of complexity.

2.2.3.1 Technical Architecture

- **Integration:** Connects public, private, and on-premises systems using orchestration tools, APIs, and middleware for smooth data and workflow management.

- **Workload Distribution:** Enables the flexible allocation of workloads across environments for enhanced efficiency, optimized resource utilization, and improved business continuity.
- **Cloud Bursting:** Shifts excess demand from private to public clouds to scale resources during peak usage.
- **Network Management:** Ensures secure, high-performance connectivity between integrated systems, addressing security and compliance needs.

2.2.3.2 Operational Considerations

- **Flexibility:** Enables strategic workload placement across on-premises and cloud environments, providing security and scalability tailored to business needs.
- **Interoperability:** Requires seamless data compatibility and tools such as Kubernetes or Azure Arc for managing multiple environments.
- **Data Security:** Combines strong on-premises and cloud security with encryption, governance, and access controls to protect data.

2.2.4 Community Cloud

A community cloud is shared by multiple organizations with common goals, offering more privacy than public clouds. It functions like a private cloud, serving a group and allowing shared resources and responsibilities while maintaining security and providing tailored solutions to meet the group's needs.

2.2.4.1 Technical Architecture

- **Shared Infrastructure:** Multiple organizations share resources, lowering costs and meeting specific industry privacy and compliance needs.
- **Collaboration:** Enables joint projects and resource sharing while maintaining security and regulatory standards with flexible hosting options.
- **Customization:** Allows tailored performance, security, and compliance to fit unique business or regulatory requirements.

2.2.4.2 Operational Considerations

- **Cost:** Shared infrastructure reduces costs compared to private clouds, optimizing resource utilization and expenditure.
- **Governance:** Requires joint governance frameworks to manage data privacy, security, and policy compliance.
- **Security:** Requires robust security measures and coordinated policies to mitigate risks such as misconfigurations or unauthorized access.
- **Compliance:** Ensures regulatory compliance across all members through monitoring and unified data protection practices.

2.3 Characteristics of Cloud Computing

Cloud computing systems possess several key characteristics that make them highly promising for future IT applications and services. The (NIST) has identified five essential characteristics of cloud computing systems [28], as illustrated in Appendix 1 (Figure 2). These characteristics are outlined and described below [29]:

- **On-Demand Self-Service:** Users can automatically provision computing resources as needed, eliminating the need for human intervention from the provider.
- **Broad Network Access:** Cloud services are accessible from diverse devices over the network, including laptops, smartphones, and tablets.
- **Resource Pooling:** Providers share resources among multiple consumers in a multi-tenant model, dynamically allocating resources with location independence.
- **Rapid Elasticity:** Resources scale up or down quickly to meet changing demands, giving users flexibility and cost efficiency.
- **Measured Service:** Usage is monitored and metered, enabling transparency, cost control, and optimized resource allocation for both providers and consumers.

Chapter 3 Adoption and Implementation of Cloud Platforms

Chapter 3 explores key reasons for adopting cloud platforms, including high availability, data durability, virtualization, hardware and network architecture, and SLA management. It emphasizes benefits such as agility, redundancy, and cost reduction and discusses leading providers, including AWS, Google Cloud, and Azure. The chapter also examines the economics of cloud adoption, as well as the roles of virtualization and networking in achieving scalable and cost-effective operations.

3.1 Key Drivers for Cloud Platform Adoption

Organizations increasingly recognize the need for a strategic cloud adoption plan to effectively leverage the advantages of a cloud data platform. Major CSPs offer comprehensive frameworks to help businesses translate their strategic goals into actionable steps, ensuring a structured approach to cloud adoption. Many (CAFs) provide a range of tools and resources, including plan generators, trackers, templates, checklists, and readiness assessments. These tools cover critical areas such as environment preparation, governance, migration, innovation, management, organization, and security of the cloud platform, ensuring organizations follow best practices throughout the adoption process. While the benefits of the cloud over on-premise data centers are substantial, much of the focus has traditionally been on potential economic gains. However, it is important to note that migrating to a public cloud provider does not always guarantee cost savings. In fact, cost savings should not be the primary factor driving cloud adoption. Instead, organizations should prioritize the cloud's ability to enable or enhance their business objectives [30][31][32][33][34][35][27].

3.1.1 Enhancing Business Agility

Business agility refers to an organization's ability to adapt to changes and capitalize on new opportunities quickly. Cloud platforms support this agility by enabling rapid deployment and scalability, unlike traditional IT setups that take weeks to configure. Public cloud services enable the launch of global infrastructure in minutes, fostering innovation and responsiveness.

3.1.2 Business Adaptability

Cloud adoption boosts adaptability by providing flexible, scalable resources and high performance. It enables businesses to adjust capacity, explore new strategies, and leverage services such as AI and analytics to respond quickly to market shifts and customer demands.

3.1.3 Ensuring Business Continuity

Business continuity in the cloud ensures operations persist during disruptions through proactive planning, disaster recovery, and resilient cloud services. When adopting public cloud infrastructure, prioritizing business continuity safeguards critical functions against external events.

3.1.3.1 Cloud Redundancy and Disaster Recovery

Cloud redundancy duplicates resources and data to keep services running during failures. Traffic shifts to backup systems if the primary fails. Public cloud providers offer:

- **Local Redundancy:** Replicates resources within a single data center to protect against local issues.
- **Geographical Redundancy:** Spreads data across distant data centers for protection against regional outages, often at no extra cost.

Redundancy is vital for disaster recovery, preventing data loss and downtime, and ensuring business continuity even during crises. Businesses should integrate redundancy into cloud strategies and regularly review their continuity plans.

3.1.3.2 High Availability in Cloud Adoption

Redundancy in public clouds ensures reliable access to services by duplicating systems and data across environments. Most providers offer 99.99% uptime (≈52 minutes of downtime annually). However, industries that require near-continuous operations, such as healthcare, often require 99.999% uptime, limiting downtime to under 5 minutes per year.

3.1.3.3 Data Durability and Integrity

Data durability ensures that information stays intact and uncorrupted over time. Public cloud providers achieve this through extensive data replication across regions—for example, duplicating data six times across three locations. Many guarantees 99.99999999% durability ("eleven nines"), meaning data loss is extremely rare. High durability protects user experiences and business operations, maintaining trust, reputation, and revenue.

3.2 Security Considerations in Cloud Adoption

Cloud security faces challenges from hidden software and hardware vulnerabilities. Public clouds employ a shared responsibility model, dividing security tasks between the provider and the customer. Providers offer strong security tools, dedicated teams, and multi-level encryption to protect customer data and resources.

3.3 Economic Implications of Cloud Computing

Moving IT to the public cloud can cut costs by over 50%, shifting spending from upfront hardware purchases to flexible, pay-as-you-go models. Cloud providers leverage economies of scale and global reach to lower costs and improve efficiency, helping businesses avoid overprovisioning and reduce operational expenses.

3.4 Virtualization in Cloud Infrastructure

Virtualization is key to modern cloud operations, shifting tasks from hardware to software. It enables multiple (VMs) to run on a single physical server, as shown in Appendix 2 (Figure 1). While traditional servers host few applications, virtualization allows one server to support dozens or hundreds of VMs, reducing costs and hardware needs. Virtualization also encompasses areas such as web applications and databases. For example, data virtualization tools like Denodo enable users to access data from multiple sources as a single virtual database, thereby simplifying data management and improving efficiency. Without virtualization, the cloud's scalability and cost-effectiveness would not be possible.

3.4.1 Fundamentals of Hardware Virtualization

Before exploring how virtualization is implemented, it is essential to understand the fundamental components of a hardware server, Appendix 2 (Figure 2). Similar to workstations or laptops, a hardware server consists of key elements such as central processing units (CPUs), an operating system (OS), memory, and storage. These components provide the necessary infrastructure on which applications can be installed to deliver services to users.

3.4.2 Hypervisor Technologies in Cloud Environments

Hypervisors enable server virtualization by creating and managing (VMs) and allocating hardware resources, such as CPU and memory, to each VM. This ensures independent operation for each VM, optimizing physical server use and reducing costs.

3.4.2.1 Type 1 Hypervisors

Type 1 hypervisors run directly on hardware without a host OS, earning the name "bare-metal" hypervisors (see Appendix 2, Figure 3). They enable servers to host multiple, each running a different operating system, making them ideal for large data centers due to their efficiency and scalability. Examples include Microsoft Hyper-V, VMware ESXi, and Linux KVM.

3.4.2.2 Type 2 Hypervisors

Type 2 hypervisors run on top of a host operating system, such as Windows or Linux (see Appendix 2, Figure 4). They create (VMs) with separate guest operating systems, which can differ from the host—for example, running Linux on a Windows machine. However, dependence on the host OS can add costs, cause performance delays, and require more maintenance, making them less suitable for large enterprises. They are ideal for personal or small-scale use. Examples include Oracle VirtualBox and Microsoft Virtual PC.

3.5 Virtual Machines and Cloud Workloads

VM is software that simulates a physical computer, running its operating system (OS) independently of the host machine. This enables multiple virtual environments on a single physical server.

- **Host Machine:** Physical hardware and main OS.
- **Guest Machine:** VM with a separate guest OS.

Types of VMs:

- **System VMs (Full Virtualization):** Replace real machines, allowing multiple VMs to coexist on one server via a hypervisor that isolates and manages them. Modern hypervisors utilize virtualization-specific hardware to achieve improved performance.
- **Process VMs:** Run specific programs in a platform-independent environment, each acting as a self-contained computer with dedicated OS and resources.

VMs maximize hardware efficiency by creating isolated environments for separate applications and workloads.

3.6 Network Architecture in Cloud Computing

This approach focuses on the data centre network and data centre interconnect network, which are crucial areas in cloud computing. The interconnect network connects multiple data centers in private, public, or hybrid cloud environments, while the public Internet connects end users to public cloud provider data centers [36][37][38][39][40].

3.6.1 Data Center Networks

DCN connects all physical and virtual resources in a cloud data center, enabling efficient communication and high performance. As shown in Appendix 2 (Figure 5), DCNs often use a hierarchical architecture with three layers:

- **Access Layer:** Connects servers via end-of-row (EoR), top-of-rack (ToR), or integrated switches.
- **Aggregation Layer:** Consolidates access switches, supporting multi-tier applications and external connectivity.
- **Core Layer:** Provides high-speed Layer-3 switching for routing traffic between the data center and external networks.

In geographically distributed data centers, Layer-3 peering routing is critical for fast recovery from failures and preventing network loops. New optical technologies enhance throughput by adjusting network topologies, but they also introduce complexity and management overhead. A robust DCN is crucial for delivering scalable and reliable cloud services.

3.6.2 Data Center Interconnect Network

Data Center Interconnect Networks (DCIN) connect multiple data centers to deliver seamless cloud services. While traditional VPNs provide secure connections, they lack the flexibility required for modern needs, such as dynamic server migration and application mobility. DCINs use Layer 2 extensions to support disaster avoidance, high availability, and workload balancing, enabling cloud elasticity. Ongoing research aims to improve performance, load balancing, and security in these networks.

3.7 Cloud Service Providers and Vendor Ecosystem

Cloud vendors sell cloud-related products like software, hardware, and services, offering SaaS, PaaS, and IaaS solutions. Examples include Amazon, Microsoft, Google, IBM, and Oracle. Cloud providers deliver services (mainly IaaS and PaaS) over the Internet, managing physical infrastructure and offering on-demand resources. Major providers include AWS, Azure, and GCP.

When choosing a provider or vendor, organizations should consider:

- **Budget:** Assess financial feasibility.
- **Security:** Check security and compliance features.
- **Scalability:** Ensure the solution can grow with business needs.
- **Services and Tools:** Evaluate required tools and platforms.

It is also important to recognize that the best provider for one organization may not be the best for another, as different companies have varying needs and priorities. To learn more about specific providers, organizations can explore documentation, whitepapers, and case studies provided by vendors. Additionally, many cloud providers offer free trials, webinars, and certification programs to help users make informed decisions [31][42][43].

3.7.1 Service-Level Agreement (SLA) Management in Cloud Computing

In cloud computing, (SLAs) define expectations for performance, availability, and security and protect customers by providing compensation in the event of service failures. They establish a legal and formal framework between providers and consumers, ensuring mutual understanding and reliable service delivery. SLAs can be specified using the Web Service-Level Agreement (WSLA) language, originally for web services but applicable to hosting. WSLA includes parameters, metrics, measurement directives, objectives, and penalties [44][45][46].

Key SLA characteristics:

- **Attainability:** Service levels must be realistically achievable.
- **Meaningfulness:** All terms must be relevant.
- **Measurability:** Service levels should be objectively measurable.
- **Controllability:** Providers must be able to control factors that impact the SLA.
- **Understandability:** Both parties must clearly understand SLA terms.
- **Affordability:** Agreements should be cost-effective.
- **Mutual Acceptability:** SLAs should result from negotiation between both parties.

There are two types of SLAs from the perspective of application hosting. These are described in detail here.

3.7.1.1 Infrastructure SLA

An Infrastructure SLA holds the provider accountable for the availability of core infrastructure like servers, power, and network connectivity. Meanwhile, enterprises manage their applications on dedicated, isolated servers, ensuring privacy and security. Examples of service-level guarantees appear in Appendix 2 (Table 1).

3.7.1.2 Application SLA

An Application SLA in a co-location model enables providers to allocate server resources based on application needs dynamically. Providers ensure customers' (SLOs) are met, including specific performance metrics. An example is shown in Appendix 2 (Table 2).

3.8 Amazon Web Services (AWS)

Launched in 2006, AWS is one of the leading cloud platforms, offering a wide range of services, including computing, storage, networking, databases,ML, analytics, IoT, and enterprise solutions [47]. AWS helps organizations across industries enhance scalability, efficiency, and innovation, maintaining a strong presence in the cloud market since its inception [48].

3.8.1 Core Services of AWS

3.8.1.1 Compute Services (Amazon EC2)

Amazon EC2 enables users to rent virtual servers, known as instances, to run applications. EC2 offers flexible configurations, allowing users to customize the amount of computing power, memory, and storage based on their specific workload needs. With just a credit card, individuals or businesses can access a virtually limitless pool of computing resources, renting

VMs for an affordable hourly rate, making cloud computing accessible to a wide range of users [49].

3.8.1.2 Storage Solutions (Amazon S3 & EBS)

Amazon S3 offers scalable, internet-based storage for large data volumes and diverse use cases. Amazon EBS provides persistent, high-performance block storage for EC2 instances with low latency. Both solutions efficiently support large-scale storage needs [50].

3.8.1.3 Database Services

AWS offers diverse managed database services [51]:

- **Amazon RDS:** Fully managed relational databases like MySQL, PostgreSQL, Oracle, and SQL Server, automating backups, scaling, and patching.
- **Amazon Aurora:** High-performance, managed relational database compatible with MySQL and PostgreSQL.
- **Amazon DynamoDB:** Fully managed NoSQL database for scalable, low-latency applications.
- **Amazon Redshift:** Data warehousing service optimized for big data analytics and complex queries.

These services offer flexible and scalable solutions for relational, NoSQL, and data warehousing needs.

3.8.1.4 Networking Services (Amazon VPC)

Amazon (VPC) allows users to create secure, isolated cloud environments within AWS, connecting cloud resources and on-premises systems [52]. Key components include:

- **Network ACLs:** Stateless controls managing inbound and outbound traffic at the subnet level.
- **Gateways:** Connect VPCs to external networks.
- **Route Tables:** Define rules for directing network traffic.
- **VPC Peering:** Enables private communication between separate VPCs.

Together, these features ensure robust networking and security for AWS resources.

3.8.1.5 Security and Compliance

AWS emphasizes security through services like IAM for access management, KMS for encryption, Secrets Manager for protecting sensitive data, and Shield for DDoS defense [53]. Operating under a shared responsibility model, AWS secures the infrastructure while customers protect their data and access. These tools help organizations reduce risk and stay compliant with industry regulations.

3.8.2 AWS Pricing Models

AWS offers flexible pricing, including pay-as-you-go, so businesses only pay for what they use [54]. Key options include:

- **On-Demand Instances:** Pay-as-you-go pricing for resources like EC2, ideal for short-term or unpredictable workloads.
- **Spot Instances:** Up to 90% off on-demand prices for flexible, fault-tolerant workloads like big data or HPC.

- **Savings Plans:** Discounts are available for committing to consistent usage over 1 or 3 years, covering services such as EC2, Fargate, and Lambda.
- **Geographic Selection:** Deploy resources closer to users to reduce latency, comply with data laws, and optimize costs. AWS regions vary in pricing, and tools like the Simple Monthly Calculator help estimate costs.
- **Third-Party Pricing:** Ensure third-party service costs align with cost optimization goals and scale based on actual outcomes rather than total spending.

3.8.3 AWS Global Infrastructure and Availability

AWS operates globally through regions—geographically separate locations containing multiple isolated Availability Zones (AZs) for resilience and fault tolerance [55]. Each AZ has independent power, networking, and facilities, improving reliability. Deploying resources across regions reduces latency, enhances security, and supports compliance and disaster recovery, ensuring high availability for applications.

3.9 Google Cloud Platform (GCP)

Launched with Google App Engine in 2008, (GCP) has expanded into a robust suite of services, including Cloud Storage (2010), Compute Engine (2013), Cloud SQL (2014), and Kubernetes Engine (2015) [56][57][58][59]. Drawing on Google's experience with services like Search and Gmail, GCP provides scalable, reliable solutions for a diverse range of industries. Notable services, including BigQuery, Bigtable, Pub/Sub, and Dataflow, support advanced analytics and innovation. GCP empowers businesses and developers with tools for data management, AI, and scalable infrastructure.

3.9.1 Comprehensive Cloud Services Portfolio

- **Compute Engine:** IaaS service for creating customizable VMs scalable for web apps.
- **Storage:** Bigtable handles extensive, high-throughput data; Cloud Storage offers secure, scalable object storage tailored to meet cost and performance needs.
- **Data Analytics:** BigQuery is a serverless data warehouse for fast analysis of massive datasets.
- **AI & ML:** Tools like AI and ML Engine help build, train, and deploy models for diverse business challenges.
- **IoT & Networking:** GCP supports scalable IoT deployments and offers robust networking services like Dedicated Interconnect, Partner Interconnect, and Cloud VPN.
- **Serverless Computing:** Services like Cloud Functions and App Engine allow app deployment without managing infrastructure, boosting speed and scalability.

3.9.2 Performance and Scalability

- **Global Network:** GCP uses a worldwide network of data centers to ensure low latency, high availability, cost efficiency, and sustainable operations.
- **Auto-Scaling:** GCP's Kubernetes Engine (GKE) auto-scales resources to match workload demands, optimizing performance and handling traffic spikes effectively.

3.9.3 Industry Adoption and Use Cases

- Media & Entertainment: GCP offers a scalable infrastructure for content delivery, reducing costs and enhancing audience engagement.
- Healthcare & Life Sciences: GCP supports genomics research, secure data storage, and advanced analytics, driving innovation in biotech and healthcare.
- E-commerce & Retail: GCP enables digital transformation, improves analytics, and enhances customer experiences in retail operations.

3.9.4 Compute Engine Resources: Regions and Zones

Google Cloud Compute Engine distributes resources globally across regions and zones [60]. A region is a geographic area comprising at least three zones, each of which is an independent data center. Zonal resources, such as VMs or disks, benefit from fault isolation within zones. Deploying across regions ensures higher resilience. Regions connect zones via high-speed, low-latency networks for fast communication. Resources are classified as global, regional, or zonal, with regional resources shared across zones within the same region. Placement policies help optimize VM proximity, reducing latency and improving reliability.

3.9.5 GCP Pricing Models

Google Cloud offers flexible pricing options [61]:

- Pay-as-you-go: On-demand pricing with no upfront costs, ideal for variable usage but more expensive per hour.
- Long-term Reservations (Committed Use): Discounts of up to 70% are available for committing to one- or three-year usage, making them suitable for consistent workloads.
- Free Tier: Provides limited, ongoing free resources and $300 in credits for new users to explore services.

When selecting a model, organizations should consider their budget, usage patterns, and factors such as computing, storage, and network costs.

3.10 Microsoft Azure: Enterprise Cloud Solutions

Launched in 2008, Microsoft Azure is a rapidly growing cloud platform offering services across AI, analytics, computing, IoT, security, storage, and more [62]. Its strong integration with Microsoft products and flexible services makes it ideal for organizations of all sizes. Azure serves 95% of Fortune 500 companies, offering customizable solutions easily integrated with external systems.

3.10.1 Compute Services in Azure

- **Azure Virtual Machines (VMs):** Provide scalable, on-demand computing for diverse workloads, supporting multiple OS types across 60+ regions with a 99.99% SLA and robust security features [63].

- **Azure App Service:** A PaaS solution for building and scaling web apps and APIs in various languages, offering seamless DevOps integration, high availability, and strong security compliance [64].
- **(AKS):** Fully managed Kubernetes service for deploying and scaling containerized apps, reducing operational overhead and supporting fast, secure app delivery. Variants like K3s and K0s simplify Kubernetes for resource-limited environments [65].

3.10.2 Azure Storage Solutions

- Azure Blob Storage: Stores large unstructured data like text, images, and backups with scalable capacity, multiple cost tiers, and security features like encryption and RBAC. Supports block and page blobs [66].
- HPC Storage: Azure offers HPC-optimized VMs (H- and N-series) and storage solutions, including Blob Storage, Azure Files, and Disk Storage, for high-performance workloads, providing 99.999% availability and robust security [67].
- Azure Files: Delivers serverless, scalable file shares via SMB and NFS, supports multiple OS environments, and integrates with various protocols like SOAP, REST, and XML [68].

3.10.3 Networking in Azure

- Azure Virtual Network (VNet): Creates private, secure networks for deploying and managing VMs and services in Azure [69].
- Azure Virtual WAN: Centralizes networking, security, and routing, connecting branches, data centers, and Azure regions efficiently [69].
- Azure VPN Gateway: Provides secure, encrypted site-to-site communication between Azure networks and on-premises environments [69].

3.10.4 Azure AI and Machine Learning

- Azure ML: Cloud service for advanced analytics and AI, offering secure, scalable solutions across industries [70].
- Azure Cognitive Services: Suite of AI tools for NLP, speech, and vision, integrating with IoT for insights in sectors like retail and healthcare [71].
- Azure Bot Services: Platform for building and deploying conversational AI bots, with easy integration into messaging platforms and cognitive services [72].

3.10.5 Security and Identity Management in Azure

- Azure Active Directory: Cloud-based identity service offering authentication, Single Sign-On, user management, and security protocols like SAML and OAuth. Includes features like MFA and self-service password reset, with free basic functionality [73].
- Azure Security Center: Provides threat protection, security recommendations, continuous monitoring, and compliance management for Azure and hybrid environments. SSO enhances security but requires careful monitoring for potential threats [73].

3.10.6 Azure Global Geographies and Data Center Locations

Azure geographies are designed to meet data residency and compliance requirements, ensuring that critical data remains close to users [74]. Each geography contains one or more regions with fault-tolerant, high-capacity networks. Many regions include availability zones—separate data centers with independent power, cooling, and networking—connected by high-speed networks with latency latency under 2 ms. Zones are spaced to minimize shared risks from outages or weather events, thereby maintaining high availability and ensuring data synchronization. Data center locations are chosen through rigorous risk assessments to ensure resilience and reliability.

3.10.7 Azure pricing models

Azure uses a pay-as-you-go model, charging only for resources consumed, though this is pricier than reserved options [75]. New customers receive 12 months of complimentary popular services, 55 ongoing complimentary services, and a $200 credit for the first 30 days. After 12 months, standard rates apply, with some services remaining free of charge. Cost-saving options include Reserved Instances (up to 72% savings) and Spot VMs (up to 90% off), which utilize unused capacity. Pricing may change, so users should refer to Azure's official pricing page for the most up-to-date details.

Chapter 4 Triangular Membership Function-Based Estimation of Round-Trip Time (RTT) for Optimal SLA Evaluation

This chapter addresses the estimation and optimization of RTT in cloud computing environments, with a specific focus on the impact of geographical distances and network conditions. This chapter introduces a novel approach that integrates multiple triangular membership functions for both input and output variables within a fuzzy logic framework to enhance the accuracy of RTT estimation, addressing the limitations of traditional methods, particularly in time-sensitive cloud applications. The proposed fuzzy logic-based model incorporates key factors influencing RTT, including network congestion, which is evaluated in terms of time (ms) and routing policies and analyzed based on distance (kilometers) and geographic distances. By integrating these parameters, the model provides a more refined and adaptable RTT prediction than conventional estimation techniques, ensuring greater precision in cloud performance assessments. Furthermore, The chapter highlights the benefits of fuzzy logic-based RTT estimation in evaluating and selecting the optimal network performance. By explicitly modeling the impacts of geographical distance and network congestion, the proposed approach enables customers to make informed decisions about service quality, taking into account their proximity to data centers and their awareness of current network conditions. This approach ensures greater accuracy in meeting Quality of Service (QoS) requirements and maintaining compliance with (SLAs). A comparative analysis of RTT values across 28 AWS regions is presented, demonstrating that the fuzzy logic-based system consistently yields more precise and lower RTT estimates than traditional measurement methodologies available through Websites standard online tools. These findings highlight the effectiveness of fuzzy logic in estimating latency and improving SLA evaluation.

4.1 Introduction to Round-Trip Time (RTT) in Cloud Computing

Traditional cloud computing is primarily used for storing, analyzing, and processing large volumes of data. However, it struggles to handle high latency issues in time-critical applications, such as computer gaming, e-healthcare, telemedicine, and robot-assisted surgery. Network latency, which causes delays in data transmission, is a critical factor for real-time applications. Traditional cloud computing methods are often insufficient to meet the stringent Quality of Service (QoS) requirements for devices operating in these environments. Challenges in calculating and expectation the RTT further complicate efforts to minimize latency when transmitting time-sensitive data in real-time [76]. RTT is a crucial determinant of latency in cloud services. Efficient management of RTT can significantly enhance QoS by ensuring faster data exchange and reducing response times. This optimization is essential for applications dependent on real-time interactions, where latency can drastically affect user experience and satisfaction. Ensuring low RTT is also essential for maintaining SLA compliance [77]. Scientists are evaluating cloud infrastructure for next-generation applications by analyzing the impact of geographical distance on latency. Private network backbones and direct peering agreements have been shown to significantly improve latency in cloud environments, reducing the delays experienced by users across different regions [78]. One study assessed the performance of the Tahoe Least-Authority File System (Tahoe-LAFS) by comparing its write operations on community network clouds and the Azure commercial cloud platform. The results revealed that read operations outperform write operations on Azure due to the platform’s network homogeneity, highlighting the performance differences between community and commercial clouds [79]. In the pursuit of optimizing resource management and reducing communication costs, two approaches—queue-based dynamic resource allocation and spatial resource partitioning—were evaluated for their impact on latency, throughput, fairness, and latency fairness. The findings show that queue-based dynamic technology outperforms spatial partitioning in terms of latency reduction and overall performance [80]. DCN are also evolving, with line rates increasing to 200Gbps to support NVMe and distributed (ML) applications. However, this advancement leaves room for imperfect control decisions. To address this, the Bolt system was developed, founded on three core ideas: (i) Sub-RTT Control (SRC), which reacts to congestion faster than traditional RTT control loop delays; (ii) Proactive Ramp-Up (PRU), which anticipates future flow completions to quickly utilize released BW; and (iii) Supply Matching (SM), which explicitly matches BW demand with supply to maximize utilization. Bolt has been shown to reduce latency and improve flow completion times while maintaining near line-rate utilization, even at 400Gbps [81]. Cloud applications often operate exclusively on the servers provided by CSPs, accessible through a simple web browser or similar client interface. For example, AWS offers widely used business applications that are hosted on its servers and accessed online. AWS has demonstrated this by providing scalable infrastructure to accommodate various enterprise needs, further illustrating the potential impact of cloud computing [82]. Similar to how most people today opt to rent homes rather than build them, the future of computing may see organizations favoring scalable and reliable cloud providers instead of constructing their own IT infrastructures. This shift would significantly reduce the risks and costs associated with launching new applications and services, as cloud providers offer ready-made platforms for deployment [83]. The widespread enthusiasm for cloud computing has led to a surge of discussions surrounding network availability, reliability, and latency within cloud environments. Despite these discussions, there is a noticeable lack of empirical measurement studies that validate these claims. Specifically, there is a gap in research comparing networking performance metrics, such as RTT, with the actual RTT experienced by web hosting services across different geographical regions. This gap highlights the need for more comprehensive studies to better understand and address the challenges related to RTT and latency in cloud computing [76]. As a result, our research endeavors to assess the performance of networking services under varying load conditions to determine the validity of the hype generated around cloud computing. We approach the assessment of network availability from two broad perspectives: firstly, by computing network based RTT through ping tests to evaluate connectivity, and secondly, by adopting a mathematical respective with RTT approach to verify the scalability and performance claims made by CSPs [82]. To gain a deeper understanding of these aspects, we employ a fuzzy logic system that utilizes two input variables, each defined by three distinct triangular membership functions. The first input, Geographical Distance, is categorized linguistically into three categories: Small, Medium, and Long. The second input, Network Congestion, is represented by linguistic categories Light, Average, and Peak. The output of this fuzzy logic system is the Expected RTT, defined through nine distinct triangular membership functions labeled RTT1, RTT2, RTT3, RTT4, RTT5, RTT6, RTT7, RTT8, and RTT9. This system enables the measurement of service performance concerning the expected optimal RTT. The study is conducted within the AWS platform, where performance is evaluated based on the interaction between the sender and receiver when retrieving cloud services. RTT values are categorized into three distinct classes: small RTT

(RTT < 100 ms), medium RTT (100 ms < RTT < 200 ms), and large RTT (RTT > 250 ms). Following this classification, a comparative analysis is performed between the expected RTT values obtained using the triangular membership function in the fuzzy logic system and the actual RTT values provided by AWS. The findings indicate that the fuzzy logic-based approach for RTT estimation yields more accurate and predictable results than those promoted by AWS. For further investigation, ping tests were employed to analyze variations while accounting for inter-region distances and network latency. This method provides a practical solution to the first challenge identified in this study: enhancing cloud service management and selection. By integrating fuzzy logic-based SLA optimization, users can make informed decisions regarding cloud service selection based on their geographic proximity to AWS regions, ultimately improving service performance and efficiency. This contribution facilitates the analysis and evaluation of additional Quality of Service (QoS) criteria in both computing and networking, which will be examined in detail in the subsequent chapter. Furthermore, the fundamental principles underlying the fuzzy logic technology employed in this study will be systematically presented and discussed throughout this dissertation in a structured and sequential manner.

4.2 Challenges in Estimating RTT in Cloud Environments

Accurately estimating RTT in cloud environments presents a range of challenges due to the complex, dynamic nature of modern cloud architectures.

4.2.1 Geographical Distance

Cloud data centers are distributed globally, and the physical distance between nodes, such as between locations i and j, can introduce significant delays in data transmission. For example, transcontinental communications between data centers in Europe and Asia often experience higher RTT due to the long distances involved. The geographical separation between the sender and receiver plays a crucial role in network performance, particularly in terms of latency. As the distance increases, data transmission delays grow, which can have a substantial impact on time-sensitive applications that require real-time data exchange. This underscores the importance of optimizing routing and data transmission strategies to minimize the negative effects of geographical distance on network performance [84].

4.2.2 Network Congestion

As cloud networks continue to expand, network congestion becomes a growing concern, leading to variable delays in data transmission. In multi-tenant environments, where multiple clients share network resources, this competition can result in unpredictable fluctuations in RTT. A key issue often cited is the effect of out-of-order packet arrivals on the performance of TCP (Transmission Control Protocol). These out-of-order arrivals are typically interpreted as a sign of network congestion, causing the receiver to generate duplicate acknowledgements. This, in turn, prompts the sender to react as if packets were lost, triggering spurious retransmissions and unnecessary reductions in the sending rate. When it comes to flow control, the combination of traffic from multiple servers can exceed the capacity available at the destination server, further intensifying network congestion. This congestion can also spill over, affecting traffic to neighboring servers and exacerbating overall network performance issues.

Therefore, the management of congestion and the optimization of traffic flow are crucial to ensuring stable and efficient cloud network operations [85][86].

4.3 Transmission Performance Evaluation in Cloud Computing

The Internet serves as a foundational component of computational technologies, facilitating extensive data generation that is stored on servers or within cloud infrastructures. The processes of data migration and transfer are integral to maintaining system integrity, ensuring consistency, and implementing essential security and load-balancing mechanisms. Among the key metrics for assessing transmission performance in network communications is RTT, which quantifies the duration required for a signal to travel from the source to the destination and return. RTT is widely utilized to evaluate the efficiency and Quality of Service (QoS) across diverse network environments, including cellular networks, Internet of Things (IoT) systems, and traditional Internet-based frameworks [87]. RTT analysis is particularly significant in network optimization, as it aids in diagnosing transmission delays and enhancing end-to-end communication performance. Moreover, RTT plays a pivotal role in congestion control protocols, such as TCP BBRv3, which is designed to optimize BW utilization and ensure fairness in networks exhibiting variable RTT values. Within IoT environments, RTT is assessed alongside other key performance indicators, including power consumption, to enhance data transmission reliability. The integration of RTT-based optimizations enables CSPs to maintain high levels of performance and reliability while simultaneously reducing their environmental impact [88]. Cloud computing systems are subject to performance evaluations, generally categorized into resource assessments and network infrastructure assessments. Resource assessments focus on analyzing the computational performance of cloud applications, particularly concerning the hardware and virtualized environments that support these applications. Each CSP employs distinct criteria for measuring CPU utilization. For instance, Google App Engine assesses resource consumption based on "Megacycles used," whereas Amazon EC2 evaluates performance in terms of deployment duration and instance utilization. Conducting such assessments typically requires root-level access permissions, limiting them to cloud providers or certified third-party evaluators [89].

4.4 Intelligent Systems and Network Service Prediction

Intelligent systems encompass a diverse range of computational techniques derived from (AI) research, including fuzzy logic, neural networks, and genetic algorithms [90]. Among these approaches, fuzzy logic provides a powerful framework for managing uncertainty and imprecision, making it particularly effective for solving complex problems where traditional binary logic falls short. By incorporating partial truth values, fuzzy logic facilitates human-like decision-making in ambiguous situations, which is essential for applications such as control systems, decision-making processes, and pattern recognition. Fuzzy logic plays a crucial role in intelligent systems due to its capability to process uncertain, imprecise, and vague data. Unlike conventional logic systems that rely on absolute true or false values, fuzzy logic allows for degrees of truth, mimicking human reasoning and improving adaptability in dynamic environments. A fundamental aspect of fuzzy logic is the fuzzy linguistic approach, which utilizes linguistic variables to represent qualitative system attributes. This methodology is particularly beneficial for ill-defined or highly complex scenarios, enhancing flexibility and adaptability in intelligent problem-solving [91]. Additionally, fuzzy reasoning aids in system behavior analysis, allowing for interpolation between input and output conditions, simplifying complexity management, and supporting induction-based learning—a critical feature for addressing intricate computational challenges. Ensuring balanced uncertainty is essential for optimizing model performance in such systems, particularly in server management and task distribution, which are fundamental to the efficient operation of service-based infrastructures. In cloud computing and networking, fuzzy logic plays a key role in addressing complex challenges such as network delay estimation, which is critical for accurately predicting task completion times and optimizing cloud resource allocation [90][92][93]. Empirical studies and simulations have demonstrated that fuzzy logic-based decision-making models operate effectively in uncertain environments, offering high precision in estimating network delays within cloud-based infrastructures. Given the complexity and dynamic nature of cloud infrastructures, adopting flexible and adaptive methodologies is essential for effective management. By providing a structured decision-making framework, fuzzy logic enables systems to efficiently handle uncertainty, ultimately enhancing efficiency, reliability, and resilience in cloud-based operations [94][95].

4.5 Experimental Methodology for RTT Measurement and Analysis Using Fuzzy Logic

4.5.1 Experimental Testing Model Determination

Several techniques are utilized to calculate RTT in network environments, each offering varying levels of accuracy and application. One widely used method is the Ping Test, which serves as a rapid and reliable tool for assessing network performance and connection quality. This technique measures the latency in ms between a user's device and a specified remote server. The RTT value is significantly influenced by the geographical distance to the server, with greater distances typically resulting in higher RTT values. A stable network connection is indicated by a consistently straight horizontal line on a ping test chart, whereas fluctuations in RTT may signal network instability or congestion [96]. Another method for calculating RTT involves mathematical modeling techniques implemented within network infrastructures. In this context, network performance metrics are derived by measuring transactions, defined as client requests followed by server replies, including TCP and UDP flows. Each read and write transaction between client and server is timed, providing essential data for RTT calculation. Typically, network appliances, such as Exinda device (1), are strategically placed between the client and server to facilitate precise measurement. These devices timestamp each intercepted packet with high-resolution nanosecond accuracy. Since the initial packet transmission from the client is unknown, RTT is calculated by summing the server-side RTT (from appliance to server and back) and the client-side RTT (from appliance to client and back). With increasing packets traversing the Exinda appliance, RTT estimations become more accurate by continuously averaging newly captured data. Consequently, RTT provides a reliable measure of the time required for a minimal packet to travel through the network and receive acknowledgment, improving progressively with ongoing data accumulation. [97][98]. The methodology for calculating RTT, along with its visual representation and governing

(1) (https://docs.exinda.com/).

equations, is depicted in Appendix 3 (Figure 1), which provides a diagrammatic illustration of the RTT computation process. In this study, the ping technique was employed to assess the connectivity between the sender and receiver, enhancing the accuracy of the analysis and enabling precise tracking of the connection process between network nodes within the AWS computing environment. Appendix 3: 0.2 Figure 2. Ping testing process. A sample of the results obtained from the ping testing process was presented to verify the integrity of the connection and establish a reliable link between the user and the endpoint. This verification was performed across all selected servers in this study to ensure network stability and performance.

4.5.2 Data Extraction and Geospatial Analysis for Communication Testing in AWS Regions

In this study, data was systematically extracted to include the names of 28 AWS regions where data centers are located, along with relevant details necessary for conducting a comprehensive communication and connection assessment. These regions were considered as Amazon's endpoints or receivers, facilitating the evaluation of network performance across different geographical locations. To conduct this analysis, the AWS latency testing platform(2) was utilized to measure network latency between the sender and AWS endpoints. Additionally, the Haversine formula was applied to determine the latitude and longitude of each endpoint. The Haversine formula, commonly used in navigation and geospatial analysis, calculates the great-circle distance between two points on a sphere based on their geographic coordinates. This approach enabled precise estimation of the physical distance between the sender and AWS data centers. The sender's location was identified as Kut, Muhafazat Wasit, Iraq (IQ), with an IP address of 37.236.213.12 and geographical coordinates of latitude 32.6024 and longitude 45.7521, The primary objective was to analyze and extract the precise distance between the sender and all AWS regions across multiple continents, Appendix 3 (Figure 3, Table 1).This geospatial analysis facilitated a better understanding of network performance, enabling a more accurate evaluation of latency and connectivity between CSU and data centers worldwide.

4.5.3 Fuzzy Logic Framework

4.5.3.1 Design System

The proposed model employs several triangular membership functions. [99], formulated in Equation (4.1), to convert crisp values into fuzzy sets. The MATLAB Fuzzy Logic Designer tool was utilized to develop the model, as depicted in Figure 4.1, the model integrates two input parameters, as detailed in Appendix 3 (Figures 4 and 5). The model utilizes three triangular membership functions for each input parameter.

$$Triangular\ membership\ function\ (d: l, m, n) \quad = \quad \begin{cases} 0, & d < l \\ \frac{d-l}{m-l}, & l \le d \le m \\ \frac{n-d}{n-m}, & m < d \le n \\ 0, & d > n \end{cases} \tag{4.1}$$

where:

(2)(https://aws-latency-test.com/).

- l is lower bound of the triangle (left endpoint).
- m is peak or center of the triangle (point of maximum membership value, equal to 1).
- n is upper bound of the triangle (right endpoint).
- d is input value being evaluated.

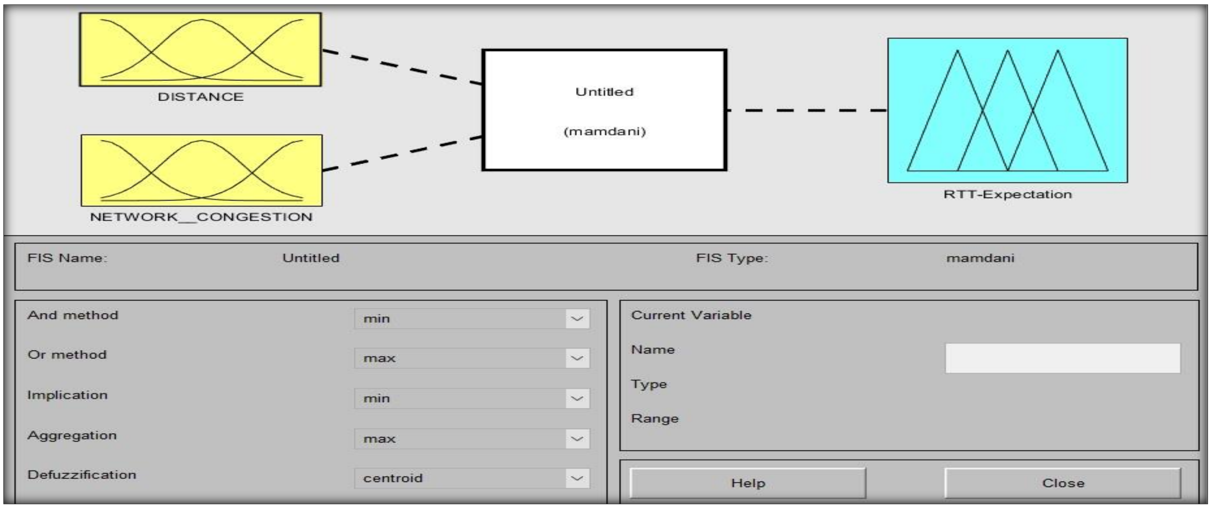


Figure 4.1. Proposed model design.

The fuzzy logic system developed in this study is grounded in a systematic design process for defining both fuzzy sets and inference rules. The membership functions for the two input variables. Geographical Distance and Network Congestion were established based on extensive analysis of RTT measurement data collected across 28 AWS regions. Geographical Distance was divided into three fuzzy sets. Small, Medium, and Long. Using triangular membership functions. The breakpoints were determined from empirical distance ranges corresponding to expected variations in RTT values.

1) Input Variables Definition
   - Distance:

     Small: [0, 862.94, 4516]; Medium: [2689, 8170, 11824]; Long: [9997, 15478, 15478.65]

Network Congestion was similarly divided into three fuzzy sets. Light, Average, and Peak reflecting latency characteristics were measured at different times of the day and under varying network load conditions.

   - Network Congestion:

     Light: [0, 3, 6]; Average: [3, 6, 8]; Peak: [7, 14, 23.59].

2) Output Variables Definition

   The expected (RTT-Expectation) output is defined in Appendix (Figure 7) as follows: RTT1: [0, 0, 25]; RTT2: [10, 50, 75]; RTT3: [50, 100, 125]; RTT4: [100, 150, 175]; RTT5: [150, 175, 200]; RTT6: [175, 200, 250]; RTT7: [200, 250, 325]; RTT8: [250, 325, 350]; RTT9: [325, 430, 500].

The output variable, Expected RTT, Table 4.1, was defined using nine triangular membership functions labeled RTT1 through RTT9, each corresponding to specific ranges of RTT delays identified in our measurements, Appendix 3 (Figure 6)., in accordance with fuzzy logic system standards (3 × 3) rules, as depicted in Appendix 3 (Figure 7).

Table 4.1 Expected RTT.

Distance / Network congestion	Small	Medium	Long
	RTT Expectation		
Light	RTT1	RTT4	RTT7
Average	RTT2	RTT5	RTT8
Peak	RTT3	RTT6	RTT9

4.5.3.2 Description of the Proposed Model

The fuzzy logic system designed for estimating RTT comprises four integral components: fuzzification, inference engine, knowledge base, and defuzzification. The fuzzification process transforms precise numerical inputs into fuzzy sets using linguistic variables, effectively managing uncertainty and variability inherent in network conditions. The inference engine utilizes a defined set of fuzzy rules to process these input fuzzy sets, generating output fuzzy sets that determine RTT estimations. The knowledge base includes a rule base of conditional (if-then) rules and a database of membership functions specifying fuzzy sets for various network parameters. Finally, defuzzification converts fuzzy output values back into precise numerical values, yielding practical RTT estimates suitable for network performance decisions [100]. By leveraging these components, fuzzy logic offers an adaptive and intelligent approach to RTT estimation, superior to traditional deterministic methods, especially in handling unpredictable network fluctuations. The structured methodology ensures accurate transformation of raw data into meaningful RTT predictions, enhancing evaluation precision and network adaptability. In the fuzzification stage, crisp numerical inputs such as Distance (measured in kilometers, indicating geographical separation between sender and receiver) and Network Congestion (measured in ms, representing network traffic intensity and its impact on latency) are translated into linguistic terms mapped onto fuzzy sets using triangular membership functions. Following fuzzification, the system applies nine comprehensive if-then fuzzy rules, enabling dynamic adaptation to varying network conditions. The fuzzy outputs derived from the inference process are subsequently converted into precise numerical values through defuzzification using the centroid defuzzification method, also known as the (COG) method. This technique ensures realistic and weighted RTT estimates that accurately reflect real-world network conditions, significantly enhancing reliability, precision, and interpretability, thereby optimizing Quality of Service (QoS) and ensuring compliance with (SLAs) in cloud computing and network management contexts. By explicitly modeling the impacts of geographical distance and network congestion, our fuzzy logic approach provides precise and realistic RTT predictions that reflect actual network conditions. This enables users and network operators to make informed decisions regarding the selection of cloud services, choosing data center regions closer to their location to minimize RTT and ensure stable performance. Users can also anticipate periods of higher network congestion, helping them avoid scheduling heavy transmissions during peak times. Moreover, RTT prediction serves as an essential optimization tool by acting as an early warning system for potential congestion. When the predicted RTT increases unexpectedly, indicating growing network congestion, dynamic protocols such as TCP can adjust transmission rates proactively, reducing data flow to prevent congestion collapse. Conversely, when predicted RTTs are low, the system allows higher throughput, maximizing available BW and improving resource utilization. RTT predictions also inform adaptive routing protocols, enabling the network to select paths with lower expected delays and to avoid congested links. This contributes to efficient BW allocation and load balancing, ensuring optimal network performance. Real-time RTT feedback supports traffic shaping policies, allowing prioritization of latency-sensitive applications like VoIP and video streaming, thus maintaining consistent QoS even under varying traffic conditions. Additionally, sustained increases in RTT act as early indicators of deeper network issues, allowing administrators to implement preemptive measures, such as upgrading links or adding capacity, to prevent severe congestion. This proactive management ensures network stability and enhances reliability. Figure 4.2 presents a surface viewer of the proposed fuzzy logic system, illustrating the relationship between distance, network congestion, and the expected RTT. The X-axis represents the geographical distance (in kilometers) between the service consumer and the cloud data center, ranging from 0 km to approximately 15,478 km, thereby covering local, regional, and global communication scenarios. The Y-axis corresponds to the network congestion level, mapped linguistically as *Light*, *Average*, and *Peak*, and modeled over a 24-hour time scale to reflect hourly fluctuations in network load. The Z-axis indicates the expected RTT, measured in ms, and the estimated delay for a data packet to travel from the user to the cloud and back. RTT values range from 0 ms to 500 ms, where higher values signify network performance degradation. The surface behavior shows that the RTT remains minimal at short distances and under light congestion conditions (e.g., RTT1: 25 ms). As the distance increases or the network congestion becomes more intense, the RTT values rise accordingly, aligning with intermediate fuzzy rule outputs such as RTT2 through RTT8. Under long-distance communication and peak congestion scenarios, the model estimates the highest RTT values (e.g., RTT9: 500 ms), which may indicate potential service delays or connection timeouts. The system employs triangular membership functions for all inputs and outputs and is governed by nine fuzzy rules defining how input combinations translate into RTT classifications. For instance, a rule such as *"If Distance is Long and Congestion is Peak, then RTT is Very High (RTT9)"* exemplifies the model's logic structure. The inference engine processes these rules to produce fuzzy output sets, which are then translated into precise RTT estimates through defuzzification using the Centroid (Center of Gravity) method, resulting in realistic and actionable RTT values that enhance network performance assessment and SLA compliance.

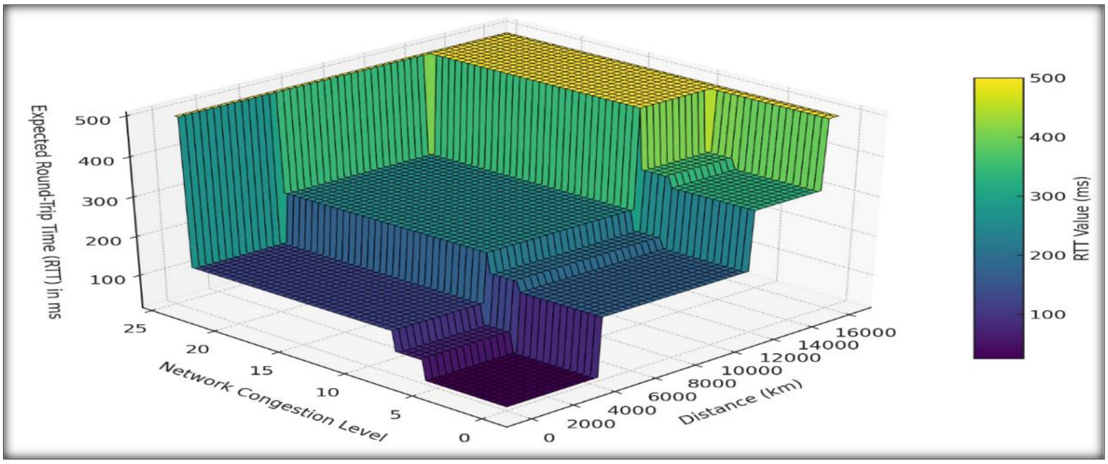


Figure 4.2 Fuzzy Logic-Based RTT Estimation by Distance and Network Congestion.

4.6 Evaluation and Analysis of the Proposed Model for RTT Estimation: Results and Discussion

The proposed model was rigorously tested to ensure its accuracy and adherence to established standards. The primary objective of this evaluation was to validate the model's reliability in estimating RTT by simulating real-world conditions. One of the critical aspects of this assessment involved verifying communication between two points on a network, specifically between a sender located in Kut, Iraq, and recipients across all AWS geographical regions. This verification, conducted using the ping tool, ensured the integrity and responsiveness of the network connection. Additionally, since RTT is influenced by factors such as geographical distance, network congestion, and peak cloud service usage, the distance between the sender and receiver was precisely calculated to account for its impact on RTT fluctuations. Following the implementation of the proposed system, the model successfully extracted and estimated RTT over 24 hours, capturing its variations across different congestion levels. The results demonstrated that during low congestion periods—typically corresponding to off-peak hours when cloud service and network traffic are minimal—the estimated RTT remained significantly low. Conversely, during moderate congestion periods, which generally coincide with regular business hours in companies and organizations, RTT values exhibited a gradual increase. The model also effectively estimated RTT under peak congestion conditions, representing the highest levels of cloud service utilization. Unlike conventional CSPs, such as AWS, which often display a single, static RTT value, the proposed model offers a dynamic and comprehensive RTT estimation. This approach enhances user confidence by providing a more detailed representation of RTT fluctuations, allowing users to make more informed decisions regarding their network performance. Table 4.2 presents details of the calculated distances between the sender and each recipient region, the RTT values reported by AWS, and the detailed RTT estimates generated by the proposed model. Furthermore, the results indicate that RTT1 to RTT3 correspond to optimal network performance, characterized by minimal latency and efficient data transmission. Conversely, RTT9 signifies severe network degradation, which may result in connection termination due to excessive delays. Intermediate RTT values, ranging from RTT4 to RTT8, reflect progressive performance deterioration, where users experience increased latency, extended page load times, and diminished service quality. Each estimated RTT result in the proposed system is labeled accordingly, allowing users to identify the most suitable geographic region based on their network requirements.

4.7 Summary of an Innovative Fuzzy Logic-Based Model for RTT Assessment in AWS Cloud Services and SLA Optimization

This research This research introduces a novel fuzzy logic-based model designed to accurately estimate RTT in AWS cloud environments. The primary objective is to improve the precision of RTT predictions by integrating multiple network parameters, particularly geographical distance, and network congestion, within a rule-based fuzzy inference framework. Compared to traditional RTT calculation methods, the proposed model offers a more detailed, dynamic, and adaptable assessment, thereby enhancing user decision-making when selecting (SLAs) from cloud providers. Traditionally, AWS supports RTT measurements using diagnostic tools such as ping and traceroute, which transmit Internet Control Message Protocol (ICMP) echo requests to specified destinations and measure the elapsed. AWS documentation describes this process as involving the execution of the ‘ping’ command, followed by the target IP address or hostname, which results in the collection of individual RTT measurements. However, this traditional mechanism has inherent limitations. RTT is highly variable, fluctuating due to factors like network congestion, routing changes, server load, and geographical distance. As a result, reporting a single RTT value for a region provides only a static snapshot rather than a comprehensive view of latency dynamics. AWS likely simplifies RTT reporting to avoid overwhelming users with excessive technical detail, abstracting the measurement methods—whether ICMP ping, TCP handshakes, or application-layer metrics—to ensure consistency across services. For the sake of user experience, AWS typically rounds or averages RTT values to present information in a more accessible format, recognizing that many users are not network engineers. While this approach is user-friendly, it can obscure important nuances, particularly for users requiring granular diagnostic insights. Moreover, revealing detailed RTT paths or node-level latency information could expose sensitive aspects of AWS's internal infrastructure or routing strategies, which are kept confidential for security and competitive reasons. In contrast, the proposed approach developed in this study offers a dynamic and adaptive assessment of RTT for each AWS data center region. It considers both the geographical distance between users and data center locations, as well as variations in network conditions, such as peak traffic periods and congestion levels. This method enables context-aware predictions rather than static, single-value estimates, allowing users to anticipate service performance and potential latency issues better. It is essential to recognize that RTT measurements can vary significantly due to fluctuating network conditions and the inherent limitations of diagnostic tools, which pose challenges for accurate RTT estimation. This study highlights the crucial importance of accurate RTT prediction in ensuring optimal Quality of Service (QoS) in cloud computing, particularly for latency-sensitive applications. The proposed model categorizes RTT into various performance levels using triangular membership functions, allowing detailed analysis of network efficiency. Furthermore, it accounts for RTT variability across different congestion scenarios, distinguishing between optimal conditions, moderate degradation, and severe latency problems that could lead to service disruptions. A significant contribution of this research lies in the comparative evaluation between the proposed fuzzy logic model and RTT values reported by AWS. While AWS typically provides static RTT measurements, the proposed system dynamically estimates RTT variations throughout different times of the day, delivering more realistic and context-sensitive insights into network performance. This dynamic capability empowers users to make informed choices when selecting cloud regions that align with their specific networking and computational requirements. Additionally, this work addresses challenges related to the availability and reliability of critical network metrics, such as RTT, which are essential for assessing the performance of cloud-based services. Future sections of this thesis will explore additional network performance indicators, including vCPU, RAM, Storage, downtime, jitter, packet loss, and BW utilization, to achieve a 99.999% reliability target. The developed fuzzy logic based RTT estimation model represents a robust, scalable, and intelligent tool for cloud service selection, significantly enhancing network performance monitoring and resource allocation. By leveraging fuzzy inference techniques, the model enables more accurate, adaptive, and real-time RTT predictions, thereby improving reliability, operational efficiency, and SLA compliance in contemporary cloud computing infrastructures.

Table 4.2 Comparison of the Proposed Model Results with AWS Round-Trip Time (RTT) Measurements.

NO	Computed Distance Between the Sender and Receiver(km)	Amazon (RTT) (ms) During Daytime	Estimated Latency Values in the Proposed RTT Classifications During Daytime Hours(ms)		
			Light congestion	Average congestion	Peak congestion
1	862.94	62	9	45	92

2	1234.23	50	9	45	92
3	3089.72	361	30	65	110
4	3428.79	88	50	86	128
5	3525.01	100	57	92	134
6	3601.23	102	62	97	138
7	3607.54	113	62	97	139
8	4009.87	115	93	127	166
9	4202.65	112	110	144	181
10	4238.49	127	113	147	184
11	4682.33	138	142	175	208
12	5981.25	388	142	175	208
13	6012.87	212	142	175	208
14	6789.34	347	142	175	208
15	7056.22	339	142	175	208
16	7289.64	369	142	175	208
17	7435.78	414	142	175	208
18	7832.90	426	142	142	208
19	8053.21	374	142	142	208
20	8923.45	181	142	142	208
21	10023.67	172	143	143	210
22	10289.47	198	155	155	232
23	12345.89	279	258	258	418
24	12678.56	242	258	258	418
25	13756.90	390	258	258	418
26	14321.76	427	258	258	418
27	14989.34	266	258	258	418
28	15478.65	300	258	258	418

Chapter 5 Quality of Service (QoS) Availability Assessment for Optimal SLA Selection

This chapter presents a significant advancement in cloud computing service selection by introducing a fuzzy logic-based classification model for evaluating Quality of Service (QoS) levels. The proposed method enhances user decision-making by enabling the confident selection of the most appropriate SLA, thereby improving the accuracy and reliability of cloud service utilization. Building upon the RTT estimation framework discussed in the previous chapter, this model expands the analysis to encompass a comprehensive set of quality-of-service parameters. It systematically evaluates computing and networking metrics, including virtual CPU (vCPU), RAM, storage, BW, delay, jitter, and packet loss. The model categorizes SLAs into nine distinct service availability levels, ranging from 90% to 99.999%. It organizes them into structured tiers, beginning with entry-level agreements such as Normal SLA and Bronze SLA, culminating in the highest reliability classification under the Gold SLA. This granular classification framework empowers users to align SLA selection with their specific performance and reliability requirements. By leveraging fuzzy logic principles, the model supports a more adaptive SLA selection process, dynamically aligning service guarantees with real-world user demands and fluctuating network conditions. This approach enhances quality of service by increasing the precision and reliability of SLA classification, particularly benefiting users with high availability and performance needs. It also facilitates intelligent cloud service provisioning by enabling responsive adjustments to variations in service quality. Overall, the proposed model establishes a robust foundation for SLA optimization, contributing to improved network efficiency, more effective resource management, and greater reliability across modern cloud computing environments.

5.1 Evaluating QoS metrics for determining SLA

Cloud computing represents a transformative paradigm in networking, enabling seamless, real-time access to a range of computing resources, including applications, servers, storage, services, and networks, without the need for upfront infrastructure investment. This model provides users significant scalability and flexibility, allowing them to pay only for the resources they consume. As a result, cloud computing facilitates the convergence of global data and service accessibility from any location at any time. Cloud infrastructure typically offers three primary service models: (SaaS), (PaaS), and (IaaS). Service providers deliver these models reliably and cost-effectively, earning user trust [101]. As cloud computing becomes increasingly ubiquitous across desktop and mobile platforms, new challenges have emerged for providers and users. The growing user base and rising storage demands have intensified concerns surrounding data privacy and system security [102]. Although cloud providers offer a broad array of services, a significant issue remains the lack of transparent guarantees regarding availability, uptime, and downtime as specified in (SLAs) [103]. In addition, network performance indicators—such as throughput RTT, jitter, and packet loss—are also critical to overall service availability [104]. These technical parameters are essential for meeting user expectations but are often presented in complex or unclear ways. Therefore, understanding the SLA decision framework is essential for ensuring timely and cost-effective service delivery. Users must ensure that cloud providers offer comprehensive guarantees regarding networking QoS metrics (e.g., BW, RTT, jitter, and packet loss) and computing QoS metrics (e.g., uptime and downtime). Before adopting cloud services, customers must conduct detailed assessments and maintain clear communication with providers to establish reliable SLA terms. A trustworthy relationship between provider and customer hinges on this clarity. Moreover, defining guarantees in a cloud environment entail identifying key performance indicators such as task execution speed and responsiveness. Cloud providers must demonstrate transparency in their service offerings through detailed documentation, SLA disclosures, and performance metrics. Significantly, validation of SLA commitments operates within the shared responsibility model, wherein accountability is distributed between the cloud provider and the customer [105]. The Shared Responsibility Model is a foundational framework for cloud security and compliance. It delineates responsibilities for various components of the cloud environment, including hardware, infrastructure, endpoints, data, configurations, operating systems, network controls, and access management. This model clearly establishes the boundary between cloud providers' obligations and those of the customers. Irrespective of the chosen service model—be it IaaS, PaaS, or SaaS—the shared responsibility framework applies universally [106]. However, the increasing complexity and variability of component-level services present additional challenges in SLA selection. Existing selection methods are generally limited to formal service attributes and fail to accommodate unquantifiable user preferences or subjective opinions. Many web interfaces only allow customers to select pre-configured service packages without explicitly articulating the guarantees these packages offer. The key challenge lies in capturing and expressing consumer preferences, which often involve abstract and non-measurable factors, and incorporating them into the decision-making process for optimal service selection [107]. To address these limitations, this research proposes a service selection mechanism that integrates users' subjective judgments into SLA decision-making. By allowing users to express qualitative preferences—referred to as "human opinions"—for each service requirement, the model ensures alignment between selected services and individual user expectations. In SLA selection, a comprehensive understanding of Quality of Service (QoS) is vital, as QoS parameters are closely linked to user needs and application demands [108]. Accordingly, this study introduces a fuzzy logic-based QoS classification model designed to support efficient and practical SLA selection. The model systematically categorizes SLAs into nine distinct availability levels, ranging from 90% to 99%, reflecting the diverse needs of cloud users. This classification incorporates both computing QoS metrics—such as vCPU, RAM, and storage—and networking QoS metrics, including BW, jitter, RTT, and packet loss. By integrating these parameters, the model facilitates a comprehensive evaluation of service quality, thereby enabling informed SLA selection. The proposed model enhances user empowerment by enabling informed decisions based on specific application requirements, budget constraints, and desired QoS guarantees. For instance, users with minimal computing demands, such as those using basic office applications, may select entry-level service tiers. Conversely, users engaged in activities like virtual conferencing may require enhanced service levels, while high-performance users, such as gamers or professionals working in video editing or scientific computation, may necessitate premium gold-tier services. The motivation for this research arises from the observed lack of clarity and interpretability in SLA representations provided by major cloud platforms. Leading providers such as AWS and GCP present SLA terms that are often difficult for users to interpret. For example, AWS specifies uptime guarantees ranging from 99.0% to 99.95%, while GCP offers guarantees for single-instance services at or above 99.95% uptime. Given the range of computing and networking services offered at varying price points, a transparent classification model is needed to assist users in navigating service availability levels. The fuzzy logic-based model presented in this study addresses this need by providing a systematic classification of SLA options. By organizing SLAs into structured tiers—ranging from Normal and Bronze to premium gold levels—the model improves clarity, enabling users to make strategic choices that optimize cost-efficiency, performance, and reliability. Additionally, it incorporates user-defined qualitative factors, making the SLA selection process more adaptive and personalized. Ultimately, this model supports better resource allocation, enhances service performance, and boosts confidence in decision-making within modern cloud computing environments.

5.2 Existing SLA Selection Methods and Service Availability Comparative Analysis

Patel et al. [109] propose an architecture for managing cloud (SLAs) using the Web Service Level Agreement (WSLA) specification, distinguishing their approach by presenting three core WSLA services that facilitate cloud SLA automation. Their method also incorporates trusted third parties to enhance security within the SLA process. Similarly, Alhamad et al. [110] outline essential criteria for formulating SLAs across service models, including (IaaS), (PaaS), and (SaaS). They emphasize specific factors for IaaS, such as boot time, scale-up/downtime, and response time, as critical components of effective SLA design. Building on the work of Alhamad and Baset, Qiu et al. [111] analyze 29 SLAs from various public cloud services, including 17 IaaS SLAs, identifying commonly mentioned attributes and significant gaps that impact the relationship between cloud providers and consumers. They note that many SLAs lack specific provisions concerning customer data, including security, privacy, protection, and backup policies, even as availability is consistently guaranteed. However, Qiu et al. also highlight a lack of detailed commitments on availability and penalties, suggesting a need for greater clarity and accountability in SLA agreements. As the demands of network applications evolve, the focus has shifted to include factors such as media quality, interactivity, and responsiveness, leading to a broader definition of (QoE). In telecommunications networks, QoE considers user satisfaction, expectations, and enjoyment [112]. In a related study, Baset [113] examines SLAs across five IaaS and PaaS providers, focusing on compute and storage services. Baset's framework dissects SLAs into various components, facilitating comparisons between providers and aiding them in defining clear, comprehensive SLAs. In line with Baset's approach, this study focuses on availability and provides a detailed classification of provider commitments to service availability. Expanding on SLA methodology, Godhrawala and Sridaran [114] propose a service-oriented architecture (SOA) that leverages a ML-based Apriori algorithm to connect quality of service (QoS) metrics, enhancing SLA strength and simplifying resource management. This approach improves SLA definitions, facilitates QoS management, reduces costs, and optimizes revenue. Akbari-Moghanjoughi et al. [115] underscore the importance of SLAs in managing service demands within ICT networks. Their survey reviews the current state of SLA establishment, deployment, and management, covering core concepts, methodologies, and challenges. The study also emphasizes the need to go beyond traditional networking by linking each Service Level Objective (SLO) to relevant service domains, with the ultimate goal of developing a comprehensive methodology for effective SLA definition, establishment, and deployment. Finally, Saqib et al. [116] address the limitations of conventional traffic classification, advocating for adaptive solutions in response to evolving traffic patterns. They introduce a framework to quantify SLA violations and an economic model to assess profitability impacts. Their study suggests adaptive ML techniques to sustain classification accuracy over time. It concludes that an adaptive traffic classifier can mitigate penalties, optimize resources, and uphold SLA integrity, offering network operators a robust approach to managing traffic dynamics.

5.3 Understanding Availability

When a failure lasts more than a few seconds, it can disrupt not only individual user requests but also subsequent retries. If repeated attempts fail, the issue is considered a service outage, impacting availability metrics. Prolonged disruptions may eventually lead users to abandon access attempts, marking the service as unavailable. In complex systems, outages are classified as either service impact outages or network element impact outages. Service impact outages directly affect end-user access and are visibly disruptive. In contrast, network element impact outages involve failures within a network component that could impact service depending on redundancy and recovery time. High-availability systems must distinguish between these types to effectively monitor downtime and ensure backup resources are in place. Suppose a second failure occurs before resolving a network element outage. In that case, a prolonged service impact outage may result, emphasizing the need for robust redundancy and quick recovery to maintain consistent service availability [117][118]. The following criteria are commonly used to classify and rank availability [117]. In practical scenarios, cloud availability calculation necessitates consideration of additional elements, such as:

$$Availability = \frac{ServiceTime - DownTime}{ServiceTime} \tag{5.1}$$

Availability is a critical metric in cloud computing, quantified as a percentage representing the ratio of system uptime to total operational time. Uptime denotes the total duration for which a system or service is expected to remain operational, whereas Downtime refers to periods of inoperability. By incorporating these variables into the standard availability formula, availability can be expressed either as a ratio or as a percentage, providing a standardized measure of service reliability. CSPs prioritize high availability to ensure continuous access to applications and data, thereby minimizing service disruptions. (SLAs) define and guarantee a specific percentage of uptime, reflecting the provider's commitment to service reliability. Service outage, commonly referred to as downtime, is determined by subtracting the uptime percentage from 100%, thereby quantifying the proportion of time during which the service remains unavailable. The availability commitment represents the extent to which cloud providers assure service availability, often serving as a key differentiator in cloud service offerings. It is important to note that reliability is either conceptually like or a broader construct encompassing service availability [119]. Among surveyed SLAs, providers generally express their commitment in terms of availability rate [120]. Highly available systems, particularly those used in telecommunications and critical cloud services, are expected to meet a minimum of 99.999% availability, commonly referred to as the "five-nines" (5–9s) reliability standard. Appendix 4 (Table 1) illustrates the maximum allowable downtime for various levels of availability commitment across different operating intervals. For example, a system adhering to the 5–9s standard permits only 5 minutes and 15 seconds of downtime over a full year of continuous operation [121]. Such stringent availability requirements are fundamental in ensuring uninterrupted service delivery, particularly in mission-critical cloud-based infrastructures.

5.3.1. Measurement Period

The Measurement Period refers to the timeframe in which cloud providers calculate their services' availability. There are two common forms: the billing month and the calendar month. The commitment level of cloud providers can vary depending on the length of the measurement period. Suppose the measurement period is set to one year. In that case, cloud providers can perform inconsistently for a few months while maintaining stability for the rest, still fulfilling the overall availability requirement. On the other hand, a measurement period of one month necessitates that providers consistently maintain stable and available services every month [122].

5.3.2 Accuracy in Service Provision

Accuracy in service provision is the extent to which cloud providers classify failed services as unavailable, varying by component, such as VMs, hosts, or entire Availability Zones. Amazon EC2, for example, considers an outage only if multiple Availability Zones lose connectivity, while Aliyun Cloud treats any instance downtime as unavailable. To improve cloud system dimensioning, analytical and simulation models at the IaaS level are employed. These models account for the heterogeneous nature of cloud systems and physical server limitations. By using analytical tools, they approximate real traffic and calculate request loss probability, offering a reliable means to evaluate service availability and optimize resource allocation [120][123].

5.3.3 Time-Based Accuracy in Availability

The accuracy in Time provision, refers to the unit of downtime used in the measurement period. Currently, three types of unit downtime are prevalent: 1 minute, 5 minutes, and half an hour. The way downtime is handled varies among cloud providers. Sometimes, if the downtime does not align perfectly with the time granularity, certain clouds may exclude those periods from the total service downtime calculation. On the other hand, other providers would include such periods in the calculation. For example, consider a cloud service experiencing a downtime of 7 minutes with a time granularity of 5 minutes. In this scenario, the eventual downtime is either 5 minutes or 10 minutes, depending on the specific policies adopted by the cloud provider. This difference in handling time granularity becomes more pronounced when using more extended periods, such as half an hour, and can significantly impact the availability calculation [120]. define availability as:

$$Availability = \frac{MTTF}{MTTF+MTTR} \tag{5.2}$$

MTTF represents the mean-time-to-failure, and MTTR denotes the mean-time-to-recovery. This measure is based on the duration when the system is either up or down, which holds significance for users. Consequently, it is unsurprising that several cloud providers, such as Microsoft's Office 365, employ this measure. Uptime corresponds to the time between failures, while downtime refers to the time taken to recover from a failure [121].

5.3.4 Exclusions in Availability Calculations

Exclusions refer to scenarios not considered when determining whether cloud services are available. Several events are not taken into account when calculating availability. In most cases, occurrences of natural disasters, regularly scheduled maintenance, network outages that occur beyond the demarcation point of the cloud provider, and internet attacks are excluded from coverage under this policy. Because these occurrences are deemed extraordinary and transient, they are not taken into account in the calculation of cloud service availability, as they may not accurately reflect the typical service performance of the provider [124].

5.4 Availability in Computing and Networking Environment

In cloud computing, ensuring the availability of critical resources such as virtual CPUs (vCPUs), RAM, and storage is essential for maintaining a reliable and efficient computing environment. The availability of these resources is governed by multiple factors, including performance, scalability, fault tolerance, SLA guarantees, elasticity, monitoring, and security. Performance optimization is a crucial aspect of cloud computing, requiring resource availability to be adaptable to workload fluctuations. Efficient allocation of vCPUs is necessary to meet processing power demands, while RAM provisioning must be adequate to support memory-intensive applications and large-scale datasets. Similarly, storage infrastructure, particularly high-performance options such as solid-state drives (SSDs), must be capable of seamlessly accommodating growing data volumes. These performance criteria directly impact the expected availability of vCPU, RAM, and storage, establishing clear reliability benchmarks for CSU. To enhance service resilience, cloud providers must implement availability strategies that encompass network monitoring, fault tolerance, and proactive system management. Network monitoring has evolved from basic connectivity checks to sophisticated analytical techniques leveraging big data, ML, and (AI). These advanced approaches enable the optimization of network traffic flow, improved efficiency, and enhanced security by predicting and mitigating potential disruptions. (SLAs) serve as contractual frameworks that define performance metrics and ensure compliance with predefined quality standards. Key SLA parameters, including delay, jitter, packet loss, and BW, play a critical role in maintaining optimal network performance. These metrics facilitate the identification of network inefficiencies, enabling CSPs to address issues that may impact overall system productivity and user experience. The assessment of core performance metrics provides valuable insights into network efficiency and availability, allowing for continuous improvement and the prevention of service degradation. By incorporating these availability and performance criteria, cloud providers can offer resilient, high-performance services that meet user expectations for reliability, scalability, and security in modern cloud computing infrastructures [120][123][125].

5.4.1 Bandwidth Considerations

The (BW) of a channel refers to the amount of data that can be transmitted per unit time, typically measured in bits per second. However, its interpretation varies depending on the context and underlying parameters [126]. One common definition equates BW with a path's capacity. For an end-to-end path composed of *n* sequential links indexed by *i = 1,.., n*, the *path capacity* C* is determined by the link with the smallest transmission capacity:

$$C^* = \min_{i=1,..,n} C_i \tag{5.3}$$

Here, C_i is the capacity of link *i*. The links where this minimum is attained—i.e., those satisfying $C_i = C^*$ are referred to as the *narrow links* or *bottlenecks* of the path. There may be multiple such links. Let i_K denote the K-th index such that $C_{ik}=C^*$. In this context, k indexes the set of links that constitute the bottlenecks. Alternatively, BW may refer to available BW, which is the unused portion of the link's capacity at a given time t. It complements the utilized BW, expressed by the utilization factor: $u_i^t \in [0,1]$ for each link. The instantaneous available BW of the path is defined as:

$$A_t^* = \min_{i=1,..,n} [C_i \,.\, (1 - u_i^t)] \tag{5.4}$$

In this formulation, the link i_K such that $A_{iK} = A_t^*$ is referred to as the *tight link*, representing the current performance bottleneck under existing traffic conditions. To account for temporal variation, the available BW is often averaged over a time interval [t, t + τ], yielding:

$$\overline{A^*}(t, t+\tau) = \min_{i=1,..,n} [C_i \,.\left(1 - \overline{ui}(t, t+\tau)\right)] \tag{5.5}$$

Where $\overline{ui}(t, t+\tau)$ is the average utilization of link i over the interval. This averaged metric offers a more stable and meaningful reflection of path availability, particularly in dynamic or congested network environments. The (BTC) refers to the upper limit of data transmission per unit of time achievable by a congestion management method, such as TCP, when implemented within a protocol. The statistic in question is influenced by various elements [127], including the quantity of concurrent TCP sessions and conflicting traffic from the (UDP), among other variables. In order to conduct measurements of body weight (BW), two approaches can be employed: an active method or a passive approach. The efficacy of active techniques is influenced by the choice of transport protocol, resulting in potential variations in the reported parameters of measurements. For instance, the utilization of the packet train technique [127], which employs UDP, enables precise determination of the path's capacity C*. Conversely, estimations of the BTC can be obtained by measurements conducted with TCP traffic. Passive techniques are dependent on the monitoring of BW utilization by applications or hosts, thereby accounting for the number of transmitted bytes within a specific time frame. Absolute thresholds are not that helpful, but when the client detects BW is low (< 100 Kbps) audio quality can easily be impacted by other applications or network congestion.

5.4.2 Network Latency and Delay

Network delay, also known as latency, is a key metric for assessing network performance. It measures the time required for a data packet to travel from its source to its destination and back, a duration referred to as RTT and typically measured in (ms). High latency can cause significant communication delays, impacting the performance of applications that rely on real-time interaction, such as video conferencing and online gaming. Factors affecting network delay include the distance between endpoints, network congestion, and the quality of network equipment [128]. The delay can be calculated using the following equation:

$$Delay\ average = \frac{Total\ delay}{Total\ packet\ received} \tag{5.6}$$

5.4.3 Network jitter

Network jitter, defined as the variation in time delay between data packets as they traverse a network, often leads to irregular arrival times that can cause lag, buffering, and reduced quality in real-time applications such as video conferencing, online gaming, and calls. High jitter is typically caused by varying traffic loads and frequent packet collisions (network congestion), which can lower Quality of Service (QoS) levels. Contributing factors include network congestion, where heavy traffic delays packets as they compete for BW; poor hardware performance from outdated or malfunctioning equipment; and insufficient packet prioritization, where important packets are not given precedence [129]. The Network jitter can be calculated using the following equation:

$$jitter = \frac{Total\ delay\ variation}{Total\ packet\ received} \tag{5.7}$$

5.4.4 Packet Loss

Network packet loss, occurring when data packets fail to reach their destination, can lead to slow internet speeds, buffering, and lag in applications like streaming, gaming, and video calls. Causes include network congestion, hardware issues (faulty routers or cables), Wi-Fi interference, software bugs, ISP issues, and bit errors due to hardware malfunctions or random noise in wireless communications. Packet loss measurement for UDP traffic often uses protocols like Q4S or IPPM, which track sequence numbers to gauge reliability. Solutions include restarting routers and devices, checking connections, switching to wired setups, reducing network load, updating firmware and drivers, minimizing router interference, adjusting Quality of Service (QoS) settings, and contacting the ISP for unresolved issues [129]. The Network packet loss can be calculated using the following equation:

$$Packet\ loss = \frac{data\ packets\ are\ sent - data\ packet\ received}{data\ packet\ are\ sent} * 100 \tag{5.8}$$

5.5 QoS Availability

5.5.1 Calculation of QoS Computing Availability Metrics

QoS computing availability is computed by aggregating the individual availability percentages for vCPU, RAM, and Storage using a weighted average, as follows:

$$A_{Computing}=(W_{vCPU}\times A_{vCPU})+(W_{RAM}\times A_{RAM})+(W_{Storage}\times A_{Storage}) \tag{5.9}$$

- A_{vCPU}, A_{RAM}, $A_{Storage}$ is represent the individual availability percentages.
- W_{vCPU},W_{RAM}, $W_{Storage}$ is represent the relative weights assigned to these metrics.

If explicit weights are not provided, equal weighting (1/3 each) is assumed, thus simplifying the formula to:

$$A_{Computing}=A_{vCPU}+A_{RAM}+A_{Storage}/3 \tag{5.10}$$

5.5.2 Calculation of QoS Networking Availability Metrics

Similarly, QoS networking availability aggregates four network metrics: BW, RTT, Jitter, and Packet Loss. The weighted average aggregation formula is:

$$A_{Networking}=(W_{BW}\times A_{BW})+(W_{RTT}\times A_{RTT})+(W_{Jitter}\times A_{Jitter})+(W_{PacketLoss}\times A_{PacketLoss}) \tag{5.11}$$

- A_{BW}, A_{RTT},A_{Jitter}, $A_{PacketLoss}$ is represent individual network metric availabilities.
- W_{BW}, W_{RTT}, W_{Jitter}, $W_{PacketLoss}$ is represent metric weights.

If explicit weights are not provided, equal weighting (1/4 each) simplifies this equation to:

$$A_{Networking}=A_{BW}+A_{RTT}+A_{Jitter}+A_{PacketLoss}/4 \quad (5.12)$$

These equations provide a structured, transparent, and reproducible approach to calculating the fuzzy inputs clearly from the individual QoS metrics.

5.6 Methodology for SLA Assessment and Optimization

5.6.1 Proposed Framework for SLA Selection

A fuzzy logic-based service guarantee model is proposed to enhance the assurance of (SLAs) within cloud computing environments (see Figure 5.1). The model employs Quality of Service (QoS) availability metrics as input variables to the fuzzy logic system, effectively capturing customer preferences, service requirements, and performance expectations. By systematically classifying QoS availability, the model facilitates a precise and context-aware evaluation of service reliability. The classification framework defines distinct SLA tiers based on availability levels: Normal SLA (90%–92%), Bronze SLA (93%–95%), Silver SLA (96%–97%), and Gold SLA (98%–99.999%). This categorization provides a clear and structured mechanism for SLA differentiation. The model ensures input consistency by validating that both QoS-computing and QoS-networking parameters are evaluated over the same domain, defined within the universe of discourse spanning from 90% to 100%. Appendix 4 (Table 2) presents the detailed definition of this domain, which serves as a reference for both input categories. The proposed model integrates two sets of input variables into the fuzzy logic system: QoS-computing parameters—including virtual CPU (vCPU), memory (RAM), and storage capacity—and QoS-networking parameters, such as BW, delay, jitter, and packet loss. These inputs collectively enable a comprehensive classification of cloud services. The methodology for estimating QoS availability and its incorporation into the fuzzy inference process is further detailed in Table 5.2. To establish a granular and structured representation of QoS availability levels, a systematic approach is adopted to define the progression of values within the universe of discourse. This sequence begins with an initial increment of approximately 0.09999, with each subsequent increment decreasing by 0.00001. The result is a smoothly increasing, non-linear sequence that converges toward a high-precision endpoint at 99.999%. The mathematical formulation governing this progression is defined in Equation (5.9):

$$A_n = 90 + (n-1) \times (0.09999 - (n-1) \times 0.00001) \quad (5.13)$$

- A_n is the nth availability level in the sequence.
- n is the index of the term ranging from 1 to 101 (for n=1, the first term A_1 is 90).

The equation initiates the sequence with a maximum increment of 0.09999, which then decreases linearly by 0.00001 per term. This formulation generates a precisely calibrated, non-uniform stepwise scale, making it particularly suitable for applications such as service level classification, where fine-grained availability tiers are necessary.

- Strengths of the Equation: When n=1:
  $A_1 = 90 + 0.(0.09999 - 0.\ 0.00001)$ $=90$, which correctly sets the starting point.
- Controlled Increment: The term:

$$(0.09999 - (n-1).\ 0.00001)$$

When n=101:

$$A_{101} = 90 + 100.(0.09999 - 100\ .\ 0.00001)$$

$$=90 + 100.(0.09999 - 0.001)$$
$$=90 + 100 \,.\, 0.09899 = 99.999$$

Furthermore, to express the output fuzzy logic-based SLA availability, the model considers uptime and corresponding downtime for a given period (e.g., daily, weekly, monthly, or yearly), based on the input QoS availability to the fuzzy logic system. The general equations for calculating uptime and downtime are formulated as follows:

$$Uptime = Total\,Time\ per\ period \times\ Uptime\ percentage \quad (5.14)$$

$$Downtime = Total\,Time\ per\ period \times (1 - Uptime\ percentage) \quad (5.15)$$

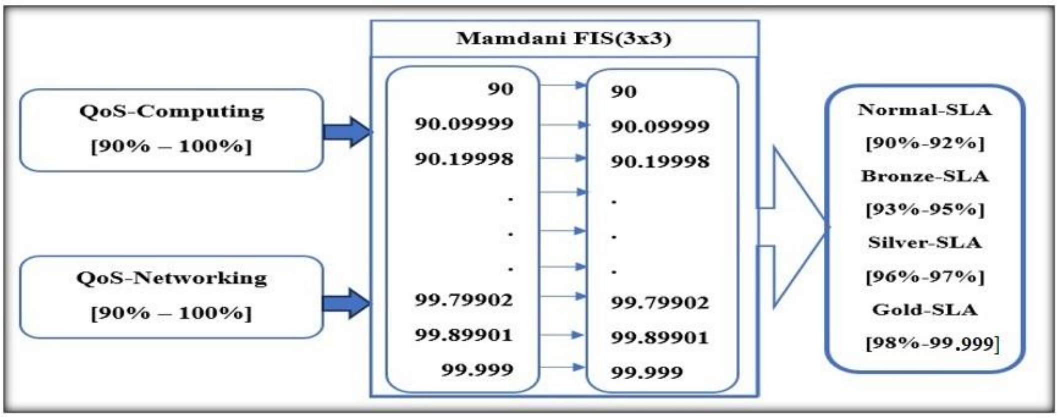


Figure 5.1 Proposed SLA guarantee model.

The detailed results of these calculations are presented in Appendix 4 (Table 3), offering a comprehensive analysis of service availability and performance assurance in cloud computing environments. By integrating fuzzy logic principles, this model provides a structured, scalable, and intelligent framework for SLA classification, ensuring an optimized and adaptive cloud service selection process.

Table 5.2 QoS Network and Computing Metrics Availability.

QoS Network Metrics Availability		
Band width	BW <500 Mbps	[90% - 92%]
	500 Mbps <= BW <1Gbps	[93% - 95%]
	1Gbps <= BW =<2.5Gbps	[96% - 97%]
	BW >2.5Gbps	[98% - 99.999]
Round Trip Time	RTT > 500 ms	[90% - 92%]
	250< RTT<=500 ms	[93% - 95%]
	100 < RTT<=250 ms	[96% - 97%]
	1<RTT<=100 ms	[98% - 99.999]
jitter	35<= Jitter <=45 ms	[90% - 92%]
	25< Jitter <=35 ms	[93% - 95%]
	15< Jitter <=25 ms	[96% - 97%]
	1< Jitter <=15 ms	[98% - 99.999]
Packe t loss	10 < Packet loss <=25 ms	[90% - 92%]
	5 < Packet loss <=10 ms	[93% - 95%]
	1 < Packet loss <=5 ms	[96% - 97%]

	0< Packet loss <=1 ms	[98% - 99.999]
QoS Computing Metrics Availability		
vCPU	1< VCPU <=2	[90% - 92%]
	2< VCPU <=16	[93% - 95%]
	16< VCPU <=64	[96% - 97%]
	64< VCPU <=192	[98% - 99.999]
RAM	4< RAM <=8 GB	[90% - 92%]
	8< RAM <=64 GB	[93% - 95%]
	64< RAM <=256 GB	[96% - 97%]
	256< VCPU <=768 GB	[98% - 99.999]
STORAAGE	1< Storage <=2 GB	[90% - 92%]
	2< Storage <=12 GB	[93% - 95%]
	12< Storage <=32 GB	[96% - 97%]
	32< Storage <=88 GB	[98% - 99.999]

5.6.2 Fuzzy Logic-Based Methodology for QoS Evaluation

5.6.2.1 Key Input Parameters

Fuzzification is a foundational process in fuzzy logic systems through which crisp numerical inputs are converted into fuzzy sets characterized by linguistic variables, terms, and corresponding membership functions [98]. This transformation enables the system to represent imprecise or uncertain information, supporting more flexible, adaptive, and human-like reasoning in decision-making contexts. The input parameters for the model were designed using the Fuzzy Logic Designer, following the same methodological framework introduced in Chapter 4. However, the division of the universe of discourse in this chapter has been modified to suit the specific primitives and structural requirements of the model developed herein. Through this approach, the model systematically converts crisp QoS input values into fuzzy sets, allowing for the nuanced evaluation of computing and networking resource availability. These fuzzy sets serve as the basis for inferring the final SLA classification, thus supporting the accurate and optimized categorization of service levels. The first input to the fuzzy logic system corresponds **to** QoS-computing availability**.** This input is defined over a universe of discourse ranging from 90% to 100% and is represented using three triangular membership functions, structured as follows:

- Light Availability: [90, 90, 95]
- Middle Availability: [90, 95, 100]
- High Availability: [95, 99.999, 100]

The second input to the fuzzy logic system is QoS-networking availability, which reflects the availability of networking resources. Like the QoS-computing input, this parameter is defined over a universe of discourse spanning from 90% to 100% and is represented using three triangular membership functions, structured as follows:

- Low Availability: [90, 90, 95]
- Average Availability: [90, 95, 100]

- Top Availability: [95, 99.999, 100]

By integrating these membership functions, the fuzzy logic system systematically evaluates availability conditions for both computing and networking resources. This structured approach enhances the model's ability to classify SLAs, ensuring that cloud service consumers receive accurate, reliable, and context-aware service guarantees tailored to their specific needs.

5.6.2.2 Implementation of FIS and Defuzzification for SLA Analysis

To achieve an accurate and adaptive SLA classification, the proposed model implements a Mamdani FIS, utilizing three membership functions for the first input (QoS-computing) and three membership functions for the second input (QoS-network). Given this structure, the model requires 3 × 3 inference rules, ensuring a comprehensive decision-making process by considering all possible input-output relationships.

i. Fuzzy Inference Rules
Fuzzy inference rules play a critical role in fuzzy logic systems, using IF...THEN conditions to interpret input values and generate corresponding decisions. These rules effectively handle uncertain or imprecise information, transforming crisp input values into fuzzified outputs, which are then utilized for intelligent decision-making [98]. The model employs the following fuzzy rule base in table 5.3:

Table 5.3 Fuzzy rule base.

QoS- Computing / QoS-Network	Light	Middle	High
	(SLA) Guarantees		
Low	Normal-SLA1	Bronze-SLA1	Silver-SLA1
Average	Normal-SLA2	Bronze-SLA2	Silver-SLA2
Top	Normal-SLA3	Bronze-SLA3	Gold-SLA

This rule base ensures that SLA classification is performed systematically, considering both computing resource availability (vCPU, RAM, and Storage) and networking parameters (BW, delay, jitter, and packet loss).

ii. System Outputs
Once the fuzzification and inference process is completed, the final step involves defuzzification, which converts fuzzy outputs into precise (crisp) values. This transformation is crucial for practical decision-making, as it provides a definitive SLA classification. The proposed model utilizes the centroid method of defuzzification, a widely adopted mathematical technique in fuzzy logic systems [130]. In the proposed model, triangular membership functions are employed during the fuzzification phase to map crisp inputs into fuzzy sets. After the inference process, the fuzzy output is converted into a single crisp value via the centroid defuzzification method. This crisp output, lying within the range of 90 to 100, is then mapped into SLA categories based on defined availability thresholds, supporting precise SLA classification. The SLA classification follows nine membership functions, as described below:

1) Normal-SLA1: [90, 90, 91]

2) Normal-SLA2: [90, 91, 92]
3) Normal-SLA3: [91, 92, 93]
4) Bronze-SLA1: [92, 93, 94]
5) Bronze-SLA2: [93, 94, 95]
6) Bronze-SLA3: [94, 95, 96]
7) Silver-SLA1: [95, 96, 97]
8) Silver-SLA2: [96, 97, 98]
9) Gold-SLA9: [97, 99.999, 100]

The model enables precise classification of QoS availability by implementing a fuzzy logic system, ensuring that cloud service consumers receive context-aware and reliable SLA commitments aligned with their specific computing and networking requirements.

5.6.2.3 Development and Validation of the Fuzzy Rule Base

The development of the fuzzy inference rules in the proposed SLA classification model was conducted through a systematic and rigorous process designed to ensure both technical correctness and practical relevance.

i. Rule Development Process

The fuzzy rule base was established by combining three primary sources of knowledge:

- Review of Existing Literature and Standards

A thorough examination of existing research, industry standards, and publicly available SLA documentation from major CSPs (e.g., AWS, GCP) was performed. This review provided essential insights into typical thresholds and relationships among various QoS metrics and their mapping to different service levels. These references informed the selection of membership function ranges and the structuring of the rule base.

- Domain Expert Knowledge

Input from domain experts specializing in cloud architecture, network engineering, and SLA management was sought to translate practical operational realities into rule definitions. Experts provided guidance on how different combinations of computing and networking availability metrics should correlate with SLA tiers such as Normal, Bronze, Silver, and Gold. For example, configurations combining high networking availability with only medium computing availability were designated to mid-tier SLAs rather than premium tiers, reflecting practical service constraints.

- Logical Consistency and Monotonicity

A key design principle in the rule formulation was maintaining logical consistency and monotonicity. The rules were constructed to ensure that an increase in input availability metrics would not result in a decrease in the classified SLA level. This principle preserves intuitive system behavior and guarantees a smooth and logically coherent transition between service levels as input conditions improve.

ii. Validation of the Fuzzy Rule Base

Validation of the fuzzy rule base was performed through multiple complementary approaches:

- Expert Review

The initial rule set was reviewed by domain experts who evaluated the rules for correctness, completeness, and practical applicability. Feedback from these experts led to refinements that ensured the rules accurately captured realistic service relationships and expectations in cloud environments.

- Simulation-Based Testing

Extensive simulations were conducted in MATLAB to verify the behavior of the FIS across the entire defined universe of discourse (90% to 100%). The simulations involved systematically varying input parameters to confirm:

a) The logical consistency of the output as inputs changed.
b) Correct handling of boundary conditions.
c) Absence of contradictory or ambiguous rule interactions.

- Benchmarking Against Real-World SLA Policies

The output classifications from the fuzzy system were compared with published SLA guarantees from major CSPs. This benchmarking ensured that the model's thresholds and classifications aligned closely with industry practices and expectations, further validating the practical reliability of the developed rules.

The entire process was iterative in nature. Based on simulation outcomes and expert feedback, several rounds of refinement were undertaken to adjust the rules until the desired level of accuracy and robustness was achieved.

iii. Responsibility for Rule Development and Validation

The creation and validation of the fuzzy rule base were collaborative efforts. The rule development and computational testing were conducted by the authors of this thesis. Domain experts provided critical reviews and validation, ensuring that the fuzzy logic model's rules were technically sound and operationally meaningful. This integrated approach to rule development and validation strengthens the credibility and practical applicability of the proposed fuzzy logic-based SLA classification system. It ensures that the model is not only theoretically sound but also well-suited for real-world deployment in cloud computing environments.

5.7 Experimental Evaluation

The proposed model was extensively analyzed within the MATLAB environment to assess its effectiveness in evaluating SLA classifications based on Quality of Service (QoS) parameters for computing and networking resources. The model was designed to process customer preferences by computing the availability ratio of virtualized computing resources—such as vCPU, RAM, and storage—alongside network resources, including BW, delay, jitter, and packet loss. By integrating these metrics into a Fuzzy Logic-based framework, the model systematically classified services into multiple SLA categories to provide a granular and data-driven approach to service selection. The Fuzzy Logic inference system extracted results according to predefined conditions and criteria, which were established during the model design phase. These results were systematically categorized into multiple SLA levels based on their corresponding availability ratios. The classification hierarchy begins with the Normal

SLA tier, which includes Normal-SLA 1, Normal-SLA 2, and Normal-SLA 3; as availability conditions improve based on input classifications and the selected fuzzy inference rules, the model sequentially transitions into the Bronze SLA tier, which consists of, Bronze-SLA 1, Bronze-SLA 2, Bronze-SLA 3, In each classification level, the availability percentage progressively increases according to the pre-established input classification rules, ensuring a systematic and logical increase in service quality. Following this, the model advances to the Silver SLA tier, which further refines the service levels with improved availability metrics, Silver-SLA 1, Silver-SLA 2; at the highest tier, the Gold SLA classification represents the most optimal service category, characterized by the highest levels of availability and reliability, suitable for mission-critical applications requiring minimal downtime. The classification hierarchy, As illustrated in Figure 5.2, the model dynamically adjusts service availability ratios in response to varying QoS computing and networking inputs. This structured classification enables cloud consumers to identify and select the most suitable SLA level based on their specific performance requirements and budgetary constraints. Additionally, Table 5.4 presents a detailed explanation of the fuzzy input-output mappings and their corresponding SLA guarantees, showcasing the effectiveness of the proposed system implementation.

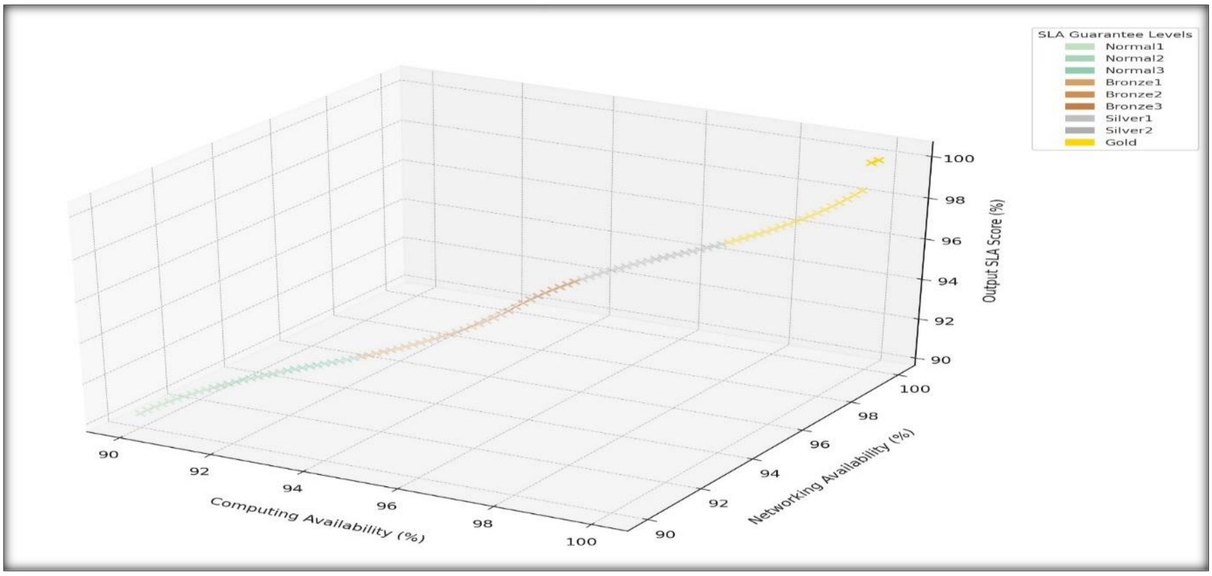


Figure 5.2 Results of the proposed model.

Table 5.4 Fuzzy Input-Output Mapping and Corresponding SLA Guarantees.

No	First input (Computing)	Second input (Networking)	Output	SLA Guarantees
1	90	90	90.333	SLA-Normal1 (90%)
2	90.09999	90.09999	90.467	SLA-Normal1 (90%)
3	90.19998	90.19998	90.592	SLA-Normal1 (90%)
4	90.29997	90.29997	90.708	SLA-Normal1 (90%)
5	90.39996	90.39996	90.816	SLA-Normal1 (90%)
6	90.49995	90.49995	90.916	SLA-Normal1 (90%)
7	90.59994	90.59994	91.010	SLA-Normal1 (90%)
8	90.69993	90.69993	91.098	SLA-Normal1 (90%)
9	90.79992	90.79992	91.181	SLA-Normal1 (90%)

This table extends and complements the information presented in Table 5.4.				
10	90.89991	90.89991	91.259	SLA-Normal1 (90%)
11	90.9999	90.9999	91.333	SLA-Normal2 (91%)
12	91.09989	91.09989	91.402	SLA-Normal2 (91%)
13	91.19988	91.19988	91.468	SLA-Normal2 (91%)
14	91.29987	91.29987	91.530	SLA-Normal2 (91%)
15	91.39986	91.39986	91.589	SLA-Normal2 (91%)
16	91.49985	91.49985	91.645	SLA-Normal2 (91%)
17	91.59984	91.59984	91.699	SLA-Normal2 (91%)
18	91.69983	91.69983	91.749	SLA-Normal2 (91%)
19	91.79982	91.79982	91.798	SLA-Normal2 (91%)
20	91.89981	91.89981	91.844	SLA-Normal2 (91%)
21	91.9998	91.9998	91.888	SLA-Normal3 (92%)
22	92.09979	92.09979	91.931	SLA-Normal3 (92%)
23	92.19978	92.19978	91.971	SLA-Normal3 (92%)
24	92.29977	92.29977	92.010	SLA-Normal3 (92%)
25	92.39976	92.39976	92.047	SLA-Normal3 (92%)
26	92.49975	92.49975	92.083	SLA-Normal3 (92%)
27	92.59974	92.59974	92.122	SLA-Normal3 (92%)
28	92.69973	92.69973	92.163	SLA-Normal3 (92%)
29	92.79972	92.79972	92.205	SLA-Normal3 (92%)
30	92.89971	92.89971	92.249	SLA-Normal3 (92%)
31	92.9997	92.9997	92.296	SLA-Bronze1 (93%)
32	93.09969	93.09969	92.344	SLA-Bronze1 (93%)
33	93.19968	93.19968	92.395	SLA-Bronze1 (93%)
34	93.29967	93.29967	92.448	SLA-Bronze1 (93%)
35	93.39966	93.39966	92.503	SLA-Bronze1 (93%)
36	93.49965	93.49965	92.562	SLA-Bronze1 (93%)
37	93.59964	93.59964	92.623	SLA-Bronze1 (93%)
38	93.69963	93.69963	92.688	SLA-Bronze1 (93%)
39	93.79962	93.79962	92.756	SLA-Bronze1 (93%)
40	93.89961	93.89961	92.828	SLA-Bronze1 (93%)
41	93.9996	93.9996	92.904	SLA-Bronze2 (94%)
42	94.09959	94.09959	92.984	SLA-Bronze2 (94%)
43	94.19958	94.19958	93.070	SLA-Bronze2 (94%)
44	94.29957	94.29957	93.161	SLA-Bronze2 (94%)
45	94.39956	94.39956	93.257	SLA-Bronze2 (94%)
46	94.49955	94.49955	93.360	SLA-Bronze2 (94%)
47	94.59954	94.59954	93.470	SLA-Bronze2 (94%)
48	94.69953	94.69953	93.588	SLA-Bronze2 (94%)
49	94.79952	94.79952	93.715	SLA-Bronze2 (94%)
50	94.89951	94.89951	93.851	SLA-Bronze2 (94%)
51	94.9995	94.9995	93.999	SLA-Bronze3 (95%)
52	95.09949	95.09949	94.172	SLA-Bronze3 (95%)
53	95.19948	95.19948	94.332	SLA-Bronze3 (95%)
54	95.29947	95.29947	94.481	SLA-Bronze3 (95%)
55	95.39946	95.39946	94.620	SLA-Bronze3 (95%)
56	95.49945	95.49945	94.749	SLA-Bronze3 (95%)

This table extends and complements the information presented in Table 5.4.				
57	95.59944	95.59944	94.870	SLA-Bronze3 (95%)
58	95.69943	95.69943	94.983	SLA-Bronze3 (95%)
59	95.79942	95.79942	95.090	SLA-Bronze3 (95%)
60	95.89941	95.89941	95.190	SLA-Bronze3 (95%)
61	95.9994	95.9994	95.285	SLA-Silver1 (96%)
62	96.09939	96.09939	95.374	SLA-Silver1 (96%)
63	96.19938	96.19938	95.459	SLA-Silver1 (96%)
64	96.29937	96.29937	95.539	SLA-Silver1 (96%)
65	96.39936	96.39936	95.615	SLA-Silver1 (96%)
66	96.49935	96.49935	95.687	SLA-Silver1 (96%)
67	96.59934	96.59934	95.755	SLA-Silver1 (96%)
68	96.69933	96.69933	95.821	SLA-Silver1 (96%)
69	96.79932	96.79932	95.883	SLA-Silver1 (96%)
70	96.89931	96.89931	95.942	SLA-Silver1 (96%)
71	96.9993	96.9993	95.999	SLA-Silver2(97%)
72	97.09929	97.09929	96.054	SLA-Silver2(97%)
73	97.19928	97.19928	96.106	SLA-Silver2(97%)
74	97.29927	97.29927	96.155	SLA-Silver2(97%)
75	97.39926	97.39926	96.203	SLA-Silver2(97%)
76	97.49925	97.49925	96.249	SLA-Silver2(97%)
77	97.59924	97.59924	96.305	SLA-Silver2(97%)
78	97.69923	97.69923	96.364	SLA-Silver2(97%)
79	97.79922	97.79922	96.425	SLA-Silver2(97%)
80	97.89921	97.89921	96.488	SLA-Silver2(97%)
81	97.9992	97.9992	96.555	SLA-Gold (98%)
82	98.09919	98.09919	96.624	SLA-Gold (98%)
83	98.19918	98.19918	96.697	SLA-Gold (98%)
84	98.29917	98.29917	96.773	SLA-Gold (98%)
85	98.39916	98.39916	96.853	SLA-Gold (98%)
86	98.49915	98.49915	96.936	SLA-Gold (98%)
87	98.59914	98.59914	97.024	SLA-Gold (98%)
88	98.69913	98.69913	97.117	SLA-Gold (98%)
89	98.79912	98.79912	97.215	SLA-Gold (98%)
90	98.89911	98.89911	97.318	SLA-Gold (98%)
91	98.9991	98.9991	97.427	SLA-Gold (99%)
92	99.09909	99.09909	97.543	SLA-Gold (99%)
93	99.19908	99.19908	97.665	SLA-Gold (99%)
94	99.29907	99.29907	97.795	SLA-Gold (99%)
95	99.39906	99.39906	97.934	SLA-Gold (99%)
96	99.49905	99.49905	98.081	SLA-Gold (99%)
97	99.59904	99.59904	98.239	SLA-Gold (99%)
98	99.69903	99.69903	98.408	SLA-Gold (99%)
99	99.79902	99.79902	98.590	SLA-Gold (99%)
100	99.89901	99.89901	99.899	SLA-Gold (99%)
101	99.999	99.999	99.999	SLA-Gold (99.999)

The inputs for QoS availability—both for computing and networking—are inherently continuous variables. However, Table 5.4 is not intended to serve as a discrete or static "lookup" table. Instead, it presents a sampled output from the continuous fuzzy mapping function that is defined and implemented via our Mamdani-type (FIS). As detailed in Section 5.5.2.1 of the manuscript, both QoS-Computing and QoS-Networking availabilities are fuzzified using triangular membership functions over a continuous universe of discourse ranging from 90% to 100%. These inputs are then processed using a fuzzy rule base (outlined in Section 5.5.2.2) consisting of 9 inference rules. The output SLA classification is derived through fuzzy reasoning and defuzzification (via the centroid method), producing a continuous mapping function from input QoS metrics to a numerical SLA guarantee level. Table 5.4 merely illustrates a dense sampling from this function, incremented using a mathematically defined non-linear progression (as explained in Equation 5.13), for demonstration and analysis purposes. These values are generated from a MATLAB simulation and demonstrate how the fuzzy model transitions through SLA categories (Normal, Bronze, Silver, and Gold) as input availabilities gradually increase. Therefore, while Table 5.4 may appear tabular, it is a result of a continuous fuzzy mapping, not a discrete mapping in the classical sense.

5.8 Summary of the SLA selection Model

One of the central contributions of the proposed model lies in its ability to align user preferences with optimal SLA classifications in real-time dynamically. The system effectively accommodates the inherent uncertainties in computing and networking performance by applying fuzzy logic principles, enabling a more adaptive and responsive approach to SLA selection. This intelligent mechanism surpasses traditional, static SLA models defined solely by service providers, offering enhanced flexibility and personalization. Furthermore, the model introduces a structured method for calculating and classifying availability ratios, equipping (CSPs) with a systematic framework for delivering tiered service offerings tailored to individual user requirements. Unlike conventional frameworks that depend on fixed SLA definitions, the proposed approach enables dynamic SLA mapping, ensuring more responsive and context-aware service delivery. The experimental analysis provides compelling evidence of the model's practical relevance. A comprehensive simulation in MATLAB was conducted using over 100 paired input values representing computing and networking QoS availability. The FIS generated output SLA classifications that followed a consistent, continuous gradient aligning with widely recognized SLA tiers such as SLA-Normal, Bronze, Silver, and Gold, as detailed in Table 5.4. For instance, the model produced granular availability scores, including 90.333%, 91.333%, 92.296%, 95.999%, and 99.999%, each accurately mapped to the corresponding SLA category. These classifications are consistent with publicly published SLA policies by providers such as AWS EC2, which outline guarantees for availability levels such as 99.5% and 99.99%. The output labels assigned by the fuzzy system (e.g., SLA-Bronze3 for the availability of 95.999%) closely mirror the expected service levels defined by industry standards. This correlation affirms the model's classification accuracy and real-world applicability, positioning it as a robust decision-support tool for SLA compliance assessment in cloud environments. Building on these results, our focus shifts to enhancing decision-making accuracy, which is addressed further in this study's next contribution. This next step involves refining fuzzy logic systems through optimization techniques to improve decision-making in complex systems. We aim to develop adaptive fuzzy logic models for efficient cloud service management and SLA optimization, tackling the challenges identified in this thesis.

Chapter 6 Enhanced Decision-Making in Uncertain Domains

Thia Chapter presents an advanced mathematical methodology designed to facilitate decision-making in uncertain environments. The advanced mathematical methodology introduced in this chapter consists of three original algorithms (Sections 6.4.1 to 6.4.3) that mathematically define the computation of membership functions using geometric and probabilistic models. Unlike conventional fuzzy logic approaches dependent on toolboxes and heuristic tuning, our method formulates precise mathematical calculations for determining membership degrees, offering enhanced precision, efficiency, and independence from specialized software tools. This chapter introduces a mathematically streamlined methodology for defining and computing fuzzy membership functions. Rather than relying on heuristic adjustments or external fuzzy toolboxes, we propose optimized analytical algorithms that directly compute membership degrees for triangular, trapezoidal, and Gaussian functions. Here, 'optimization' refers to the efficient, precise calculation process derived from mathematical principles, reducing complexity and dependency on manual tuning rather than numerical optimization in the classical sense. The primary contribution of this chapter is the formulation of an optimized strategy for the selection and implementation of fuzzy membership functions. Notably, the novelty of this approach is explicitly situated in the methodological innovations rather than the mere act of classifying input values. Specifically, the introduced mathematical model incorporates systematic and optimized algorithms for efficiently computing membership degrees. Unlike traditional fuzzy logic approaches that rely heavily on predefined, static membership functions—such as standard triangular, trapezoidal, or Gaussian forms, typically defined manually or through heuristic adjustments—the proposed methodology utilizes structured mathematical optimization techniques. This enables the dynamic and precise classification of crisp input values into corresponding fuzzy sets, thereby significantly enhancing accuracy and computational efficiency. The distinctiveness of this model arises from its structured mathematical optimization approach, systematically refining the process of classifying crisp inputs into fuzzy sets. Doing so achieves greater precision and computational efficiency than conventional methods reliant on heuristics or manual adjustments. This model explicitly incorporates optimization algorithms to streamline and enhance the calculation of membership degrees via three specialized algorithms, each analogous to traditional fuzzy logic membership functions, namely triangular, trapezoidal, and Gaussian. A significant aspect of the proposed approach lies in its independence from conventional fuzzy logic implementations that frequently depend on specialized fuzzy logic software, such as MATLAB's Fuzzy Logic Toolbox or other simulation frameworks. Traditional methods often rely on specific software dependencies, plugins, or graphical tools to define membership functions and inference mechanisms, which limits their adaptability and operational efficiency across various computational contexts. In contrast, the proposed method introduces a simplified, mathematically driven, and tool-independent model that does not necessitate external fuzzy logic software or environment-specific configurations. The advantage of this independence is evident in its broader applicability, simplified integration processes, and reduced computational requirements. Due to its inherent simplicity, computational efficiency, and high adaptability, the proposed method exhibits substantial potential across diverse AI applications, eliminating the necessity for complex adaptive systems or specialized software environments. This simplified mathematical framework ensures faster and more accurate classification of input values, effectively reducing computational overhead and enhancing operational performance in practical AI deployments.

6.1 Overview of Decision-Making Challenges

Fuzzy logic has become a cornerstone of intelligent control systems, seamlessly integrating with advanced methodologies such as neural networks and genetic algorithms. It is widely applied to interpret, analyze, and resolve the inherent ambiguities associated with complex human-centric needs and challenges. Its unique ability to handle imprecise and uncertain data through fuzzy sets and rules positions it as a powerful tool for decision-making in dynamic and intricate systems. The core processes of fuzzy logic—fuzzification, inference (driven by IF-THEN rules and an extensive knowledge base), and defuzzification—facilitate the conversion of vague inputs into precise, actionable outputs, ensuring effective and reliable system performance. This capability supports the suitability of robust control and decision-making across various applications. Integrating fuzzy logic with adaptive systems enhances its flexibility and optimization capabilities, making it indispensable in robotics, industrial automation, and (AI) domains. These fields frequently encounter inaccuracies from sensor data or other unpredictable inputs, whereas fuzzy logic systems demonstrate exceptional efficiency and reliability. The Mamdani fuzzy logic system is widely favored among the many fuzzy logic approaches for its straightforward structure and interpretability. In electric drive systems, fuzzy logic has been employed to develop an adaptive proportional-integral (PI) speed controller for vector control of induction motors (IM) [131]. This controller uses an Adaptive Neuro (ANFIS) to optimize control gains, ensuring resilience against parametric variations. Validation through MATLAB-Simulink simulations demonstrated its robust performance and suitability for enhancing electric drive reliability. In agriculture, fuzzy logic has addressed environmental uncertainty. For instance, a wheeled robot with a microcontroller was developed for autonomous pesticide spraying, achieving high decision-making accuracy in weed identification despite challenging environmental conditions [132]. Hydraulic systems have also benefited from fuzzy logic. Researchers proposed a discrete-time switching controller strategy for pumping stations, integrating fuzzy-PD or fuzzy-PID controllers with PI controllers. A fuzzy supervisor facilitates controller switching, ensuring robustness, stability, and asymptotic error correction [133]. In high-performance electric motor applications, integrating Model Reference Adaptive Systems (MRAS) with fuzzy logic has significantly improved rotor speed and resistance estimation in induction motors. The study "High-Performance Control of IM using MRAS-Fuzzy Logic Observer" highlights this advanced control strategy's effectiveness in high-demand environments [134]. Further advancements include a method for simultaneously estimating rotor resistance and speed using two independent adaptive observers alongside a streamlined algorithm for optimal controller gains [135]. The adaptability of fuzzy logic extends to managing ambiguity and vagueness, which occur when boundaries and alternatives are unclear. By employing fuzzy numbers and membership functions, fuzzy logic offers a structured approach to handling uncertainty, surpassing traditional Boolean logic [136][137]. This flexibility allows fuzzy logic systems to adapt to tasks such as navigation, object handling, and decision-making in uncertain environments, enabling human-like control in (AI) systems [138][139]. Classical information theory reduces uncertainty by increasing information; however, fuzzy logic uses membership functions to quantify degrees of association between inputs and sets within a universe discourse. These functions form the backbone of fuzzy logic systems, linking input values to degrees of membership and enabling approximate reasoning in complex scenarios [140][141][142]. Optimization algorithms enhance fuzzy logic by refining membership functions and improving actuator precision and control, especially in autonomous systems [143]. The development of fuzzy logic systems hinges on constructing fuzzy partitions and defining the shape and number of membership functions (MFs). These MFs are essential as they quantify the degree to which a specific input belongs to a fuzzy set. Expert knowledge is pivotal in this process, guiding the selection and parameterization of appropriate MFs. Optimizing these systems minimizes reliance on subjective trial-and-error approaches, thereby enhancing the accuracy of input/output mappings [144]. Membership functions are fundamental to representing the degree of membership for each variable, serving as critical inputs for the inference rules that drive system functionality [145]. Building upon the findings of our previous contribution, this study seeks to enhance further the accuracy and robustness of the proposed classification approach. This section provides a detailed exposition of the mathematical methodology, which centers on applying three specialized classification algorithms. These algorithms operate analogously to the membership functions used in the Mamdani fuzzy logic system. The core of this approach is a novel mathematical model designed to systematically classify crisp input values into their appropriate fuzzy sets, thereby enhancing the accuracy of membership degree computations. Optimization techniques refine these computations through three distinct algorithms, corresponding to triangular, trapezoidal, and Gaussian membership functions. The model was implemented in MATLAB and evaluated using a dataset of 10000 user task size entries with varying magnitudes. The primary objective was to assess the performance of the proposed algorithms in categorizing task sizes into three predefined classes: Small, Medium, and Big. A comparative analysis with the Mamdani fuzzy logic system demonstrated that the proposed model produces classification results that are either equivalent to or slightly more precise than those generated by Mamdani’s approach, particularly regarding numerical accuracy. These findings validate the proposed method as a viable and competitive alternative to Mamdani’s model for classification tasks. Additionally, the mathematical simplicity and independence of the proposed model from simulation environments or third-party tools, such as dynamic-link libraries (DLLs), software extensions, or external simulation frameworks, make it particularly suitable for broader deployment in AI applications. This is especially advantageous in contexts where tool-dependent environments are unavailable or impractical.

6.2 Advancements and Applications of Fuzzy Logic in Decision-Making

Fuzzy logic systems have become influential in decision-making, particularly in uncertain contexts. They offer flexibility and approximate reasoning; however, the literature points to challenges such as the complexity of fuzzy rule formulations and computational inefficiencies. These challenges underscore the need for further optimization to enhance the applicability and effectiveness of fuzzy logic across various fields. In his seminal work on fuzzy sets, Zadeh defined a fuzzy set as "a class of objects with a continuum of grades of membership," where a membership function assigns each object a grade ranging from zero to one. This work extends traditional notions such as inclusion, union, intersection, and complement to fuzzy sets, establishing various properties within this context. Notably, Zadeh also proved a separation theorem for convex fuzzy sets without requiring the sets to be disjoint [146]. Building on this foundation, researchers expanded fuzzy set theory by exploring its theoretical underpinnings and practical applications in managing uncertainty and imprecision across various domains [147]. However, these approaches often overlook the computational inefficiencies that arise when applying fuzzy logic in real-world decision-making scenarios. Recent advancements have attempted to address these inefficiencies. For instance, researchers have proposed a novel approach to healthcare decision-making that integrates fuzzy logic with ML [148]. This hybrid model aims to improve diagnostic accuracy and resource utilization, particularly when dealing with incomplete and uncertain data, thus addressing traditional inefficiencies. However, it has faced criticism for relying on subjective inputs, which can introduce biases and affect the consistency of outcomes [149]. Moreover, researchers have highlighted limitations in the fuzzy linguistic approach, particularly regarding information loss during fusion processes. They propose a 2-tuple model to enhance precision and extend aggregation operators [150], although its complexity continues to pose challenges for practitioners, making implementation cumbersome [151]. Further research has discussed adaptive fuzzy systems, which show promise but frequently experience stability issues [152], leading to inconsistent decision-making in dynamic environments [153]. The Mamdani fuzzy inference model, while foundational, is often critiqued for its limited robustness under varying conditions [154]. Although recent studies have sought to enhance this model's applicability, challenges persist in managing time-sensitive decisions effectively [155]. Additionally, the researchers provided extensive insights into fuzzy systems but focused primarily on theoretical aspects [156], which hinders practical application and adoption by industry practitioners [157]. Doong et al. explored fuzzy risk assessment in engineering [158], yet their approach does not adequately address the interactions among risk factors, potentially oversimplifying complex decision-making contexts [159]. In the context of business applications, researchers reviewed fuzzy decision-making [160], underscoring the pressing need for improved methodologies to handle severe uncertainties, particularly when data is sparse or incomplete [161]. Lastly, the integration of fuzzy logic with genetic algorithms has been explored [162]. However, this approach often struggles with computational efficiency and convergence issues, complicating its practical use in real-time decision-making scenarios [163]. In summary, the literature underscores significant gaps in the application of fuzzy logic systems within uncertain domains, highlighting the need for optimized methodologies to enhance robustness, efficiency, and applicability in decision-making processes. This study aims to address these critical gaps by focusing on accurately determining the degree of membership of input elements and their association with the most appropriate membership functions. The proposed mathematical model seeks to improve fuzzy logic systems' capacity to handle uncertainty and make accurate decisions by refining the process of selecting the best membership function and aligning it with closely related decisions.

6.3 Background of Fuzzy Logic System

6.3.1 Core Principles of Fuzzy Logic Systems

Fuzzy logic is a form of many-valued logic that deals with approximate rather than fixed and exact reasoning. Unlike traditional binary logic, which operates with true or false values, fuzzy logic allows for a range of values between 0 and 1, which makes it particularly useful for handling the concept of partial truth. This approach is often referred to as "computing with words" because it can model the way humans think and reason with imprecise information [164] [165]. Figure 6.1 depicts the architecture of a fuzzy logic system.

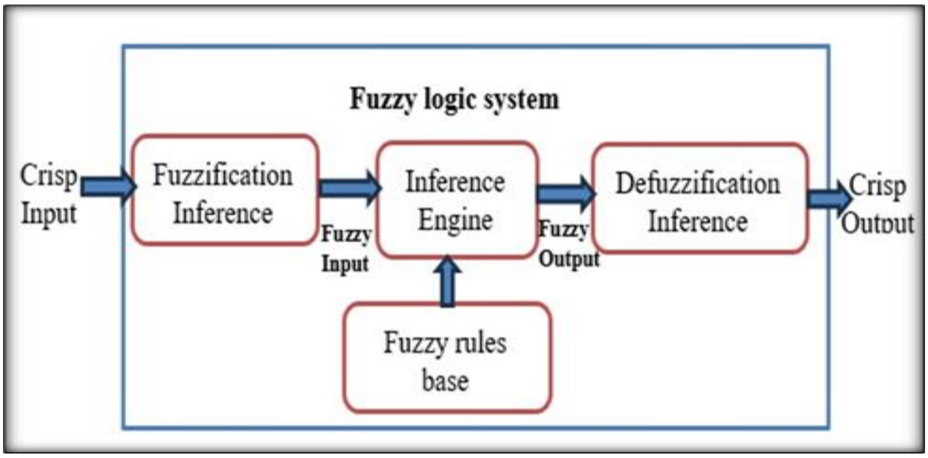


Figure 6.1 Architecture of a fuzzy logic system.

6.3.1.1 Fuzzy System Basics

6.3.1.1.1 Crisp Input Processing

In fuzzy logic, a crisp set refers to a set in which each element has a membership value that is strictly either 0 or 1, signifying complete exclusion or inclusion. This differs from fuzzy sets, where membership values can vary continuously between 0 and 1, enabling partial membership. In a crisp set, individuals are categorized into two distinct groups: members, who belong unequivocally to the set, and non-members, who are definitively excluded. Crisp sets adhere to classical binary logic, emphasizing a clear and absolute boundary for set membership. The indicator function for a crisp set, A, where elements in the set are assigned a value of 1 and those outside the set are assigned a value of 0, can be expressed as:

$$\mu_A(x) = \begin{cases} 1, & x \in A \\ 0, & x \notin A \end{cases} \tag{6.1}$$

6.3.1.1.2 Fuzzification Process

Fuzzification inference is a process that converts input data into fuzzy sets, which are subsequently used to generate outputs based on a predefined set of rules, typically expressed in the "IF…THEN" format. This process plays a vital role in FIS, facilitating the transformation of uncertain or imprecise information into structured, actionable outcomes for decision-making [166].

6.3.1.1.3 Inference Engine

An inference engine is a critical component of an expert system, employing logical rules to derive information or make decisions based on a knowledge base. It maps fuzzified inputs (obtained through the fuzzification process) to the rule base, generating fuzzified outputs for the applicable rules. The fuzzy inference engine follows a structured process comprising several key steps. Initially, it performs rule matching by identifying relevant rules from the knowledge base and comparing the input data to the conditions specified in each rule. Once the relevant rules are identified, the engine evaluates the degree of truth for each rule, determining the extent to which the input satisfies the conditions. Subsequently, it aggregates the conclusions derived from the matched rules by combining their outputs to generate a coherent decision or conclusion. This process is iterative, with the engine continuously applying rules and updating the knowledge base until a solution is achieved or no further rules apply. This systematic approach enables the fuzzy inference engine to handle complex and dynamic scenarios effectively. Inference engines are widely used in AI applications, including diagnostic systems, recommendation systems, and other decision-making tasks [167].

6.3.1.1.4 Fuzzy Rule Base

A fuzzy rule base is a set of fuzzy rules that describe the relationship between input variables and output results in a fuzzy logic system. These rules, often derived from linguistic expressions, characterize the dynamic behaviour of the system. Each rule consists of an antecedent (the "IF" part) and a consequent (the "THEN" part) based on the knowledge and expertise of a domain expert. Fuzzy rules generally follow the format:

$$\boldsymbol{if} \rightarrow \boldsymbol{antecedent}(\boldsymbol{s})\ \ \boldsymbol{then}\ \ \boldsymbol{consequent}(\boldsymbol{s})$$

Enabling the system to infer outputs under various input conditions. These rules are crucial for managing uncertainty and imprecision in control algorithms within systems [168][169].

6. 3.1.1.5 Defuzzification Process

Defuzzification is the final step in a fuzzy system and is responsible for converting the fuzzy output generated by the inference engine into a precise numerical value. This process translates the fuzzy set produced during inference into a specific, actionable numerical value suitable for decision-making or control applications. Standard defuzzification techniques, such as the Centre of Gravity (COG) method illustrated in equation 6.2, derive a crisp result by calculating a representative value from the combined fuzzy sets generated by multiple rules. This step ensures the system's outputs are interpretable and practical for real-world implementation [170].

$$Z = \sum_{i=1}^{n} (\mu_i\ \beta_i) / \sum_{i=1}^{n} \mu_i \tag{6.2}$$

Z: The crisp output (defuzzified value); μ_i *: The membership degree of the fuzzy set for the i-th rule;* β_i *: The representative value (often the centroid) of the output fuzzy set for the i-th rule.; n: The total number of rules in the system.*

6.3.2 Membership Functions and Their Significance

The membership function is a core concept in fuzzy logic. It quantifies the degree of belonging of a given input to a fuzzy set. Mapping inputs to values from 0 to 1 provides a nuanced representation of uncertainty and partial truth, enabling more flexible and accurate modelling than traditional binary logic. The function adheres to specific constraints and has a range of [0, 1]. For every x ∈ X, μ _A(x) must be unique [171]. In this study, have selected three widely used membership functions recognized as essential in fuzzy logic systems: triangular, trapezoidal, and Gaussian.

6.3.2.1 Triangular Membership Function

Triangular membership function can be represented by the parameters {a, b, c}. As referenced in the previous sections.

6.3.2.2 Trapezoidal Membership Function

Fuzzy trapezoidal MF is defined by the parameters {a, b, c, d} as in equation (6.3).

$$\mu_{F(x)} = \begin{cases} 0; \ x \leq a \\ \frac{x-a}{b-a} \ ; a< x <b \\ 1; b \leq x \leq c \\ \frac{d-x}{d-c}; \ c< x <d \\ 0; x \geq d \end{cases} \tag{6.3}$$

6.3.2.3 Gaussian Membership Function

A fuzzy Gaussian membership function uses the Gaussian distribution to measure membership levels within a fuzzy set. It creates bell-shaped curves that manage uncertainty and vagueness. The function provides a continuous range of membership values between 0 and 1. The general formula for a fuzzy Gaussian membership function is:

$$\mu A(x) = e^{-(\frac{x-c}{\sigma})^2} \tag{6.4}$$

6.4 Methodology for Enhanced Decision-Making in Uncertain Domains

The Mamdani fuzzy inference method, also known as the Max-Min method, is a widely used technique for designing control systems based on linguistic rules derived from expert knowledge. This method utilizes fuzzy set theory to establish mappings between input and output variables, making it a powerful tool in various applications. Its implementation typically follows a structured process in which fuzzy rules and membership functions are systematically defined to determine the final output [153]. Within this framework, every value in the universe of discourse is assigned a specific degree of membership across relevant fuzzy sets, regardless of its simultaneous association with other sets. This characteristic enables the proposed method to evaluate the degree to which any given value belongs to all pertinent membership functions in the model. Such an approach facilitates a comprehensive assessment of how each value influences the decision-making process within its operational context. The design of the system is further supported by mathematical principles and equations presented in the following sections.

6.4.1 Mathematical Formulation for Algorithms 1 and 2

The general equation for a straight line is expressed as in equation (6.5).

$$y=mx+c \tag{6.5}$$

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 in equation (6.6).

$$y-y1=m(x-x1) \tag{6.6}$$

Where (x, y) denotes an arbitrary point on the line. The absolute value parent function is represented as:

$$f(x) = |x| \tag{6.7}$$

It is defined as:

$$f(x) = \begin{cases} x, \; if \;\; x > 0 \\ 0, \; if \;\; x = 0 \\ -x, \; if \; x < 0 \end{cases} \tag{6.8}$$

The stretching or compressing of the absolute value function $y = |x|$ is defined by the function $y = \alpha|x|$ where α is a constant. The graph opens if $\alpha > 0$ and opens down when $\alpha < 0$. In a more general context, the equation for an absolute value function takes the form:

$$y = \alpha|x - h| + k \tag{6.9}$$

$$\alpha = \frac{y2 - y1}{x2 - x1} \tag{6.10}$$

Here, h signifies the horizontal translation, and k represents the vertical translation [163].

6.4.2 Mathematical Formulation for Algorithm 3

The Gaussian random variable is the most utilized and highly significant when investigating random variables. A Gaussian random variable is characterized by a probability density function (PDF) that can be expressed in a general form.

$$fX^{(x)} = \frac{1}{\sqrt{2\pi\sigma^2}} exp\,(-\frac{(x-m)^2}{2\sigma^2}) \tag{6.11}$$

$$\sigma = \sqrt{\frac{\sum(x_i - \overline{\overline{x}})^2}{n-1}} \tag{6.12}$$

The PDF of the Gaussian random variable has two parameters, m and σ, which have the interpretation of the mean and standard deviation (σ), respectively. The parameter σ^2 is referred to as the variance [172] [173].

6.4.3 Classifying Variables and Determining Membership Degrees in Uncertain Domains

The proposed methodology introduces a rigorous mathematical framework for categorizing inputs within a defined universe of discourse, facilitating precise and efficient determination of membership function levels. This approach incorporates three distinct algorithms derived from the mathematical formulations central to this study. The first algorithm enhances the construction of precise triangular membership functions, while the second refines the formation of trapezoidal membership functions. Additionally, the third algorithm optimizes the generation of Gaussian membership functions. At its core, this method employs a robust mathematical model that simplifies the computation of membership degrees, resulting in significantly improved processing speed compared to traditional methods such as the Mamdani fuzzy logic system. An inherent strength of this approach lies in its systematic classification of input values based on specific membership functions. By effectively addressing issues of ambiguity and uncertainty, the methodology ensures a more accurate determination of membership degrees, thereby supporting enhanced decision-making outcomes. Appendix 5 provides detailed explanations and illustrative examples validating the effectiveness of these algorithms.

Algorithm 1: Input Partitioning and Membership Classification as similar work as Triangular MF
//Membership degrees are calculated for each input value V_i with respect to membership functions defined over the universe of discourse. *//The parameters PV (parameter values) is defining the shape, boundaries, or centers of the membership functions—not the input data itself.* • *Input: V, a set of crisp input values for which the degree of membership will be calculated.* *//Parameters: Definitions of membership functions (PVs specifying boundaries, centers, or slopes for triangular MFs.* *n: The number of fuzzy partitions (i.e., number of membership functions) into which the universe of discourse is divided.* • *Output:* *A matrix of membership degrees $\mu(v_i)$ for each v_i across all defined membership functions.* *Procedure:* *1. Initialization:* • *Max(V) ← max(Vi) // Calculate the maximum value of sets V in the universe discourse.* *2. Parameter Value Calculation:* • *PV_1 ← (Max(V)/n) // Determine the first parameter value.* • *PV_n ← n × PV_1 // Compute the last parameter value.* *3. Iterate Over Each Input Value V_i in the Set of Parameter Values:* *for each $V_i \in V$:* • *Case 1:if $V_i \geq 0$ and $V_i \leq PV_1$* *$MF_1 \leftarrow (\frac{-V_i}{PV2})+1$; Output ← ($MF_1$, Degree ($V_i$))* *//Compute Membership Function 1.* *Output ← (MF_2, MF_3,…,MF_{m-1}, Degree(V_i)) // Determining the degree of element in the remaining MF domain.* • *Case 2: if $V_i \geq PV_1$ and $V_i \leq PV_2$* *$MF_1 \leftarrow (\frac{-V_i}{PV2})+1$; Output ← ($MF_1$, Degree ($V_i$))* *// Compute the degree of element affiliated with both domains MF_1 and Subsequent it, as MF_2.* *$\alpha \leftarrow (V_i - PV_2)$ // Define the alpha variable.* *$MF_2 \leftarrow (\frac{-1}{PV2-PV1}) \times (\lvert\alpha\rvert + 1)$* *// Compute the degree of element affiliated with both domains MF_2 and previous it, as MF_1.* *Output ← (MF_3, MF_4,…,MF_{m-1}, Degree(V_i))* *//Determining the element's degree of membership across the remaining membership functions.* • *Case 3: if $V_i \geq PVn - 1$ and $V_i \leq PV_n$* *$MF_m \leftarrow ((\frac{1}{PVn-PVn-1}) \times (Vi - Pn - 1)$; Output← ($MF_m$, Degree ($V_i$))* *// Calculate Membership Function m.* *Output← (MF_1,MF_2,…,MF_{m-1}, Degree(V_i))*

//Determining the element's degree of membership across the remaining membership functions.

4.End of Algorithm 1

Algorithm 2: Input Partitioning and Membership Classification as similar work as Trapezoidal MF

//Membership degrees are calculated for each input value V_i with respect to membership functions defined over the universe of discourse.
//The parameters PV (parameter values) is defining the shape, boundaries, or centers of the membership functions—not the input data itself.

- *Input: V, a set of crisp input values for which the degree of membership will be calculated.*
  //Parameters: Definitions of membership functions (PVs specifying boundaries, centers, or slopes for Trapezoidal MFs.
  n: The number of fuzzy partitions (i.e., number of membership functions) into which the universe of discourse is divided.
- *Output:*
  A matrix of membership degrees $\mu(v_i)$ for each v_i across all defined membership functions.

Procedure:

1. *Initialization:*
   - *Max (V) ← max (Vi) // Calculate the maximum value from the sets V.*
2. *Parameter Value Calculation:*
   - *PV_1 ← (Max (V)/n) // Determine the first parameter value.*
   - *PV_n ← $n \times PV_1$*
     // Compute the last parameter value.
3. *Iterate Over Each Input Value V_i in the Set of Parameter Values:*
   for each $V_i \in V$:

- *Case 1: if $Vi \geq 0$ and $V_i \leq PV_1$*
  Degree (Vi) ← 1; Output ← (MF_1, Degree (V_i)) // Compute Membership Function 1.

  Output ← (MF_2, MF_3,…,MF_{m-1}, Degree(V_i))

//Determining the element's degree of membership across the remaining membership functions

- *Case 2: if $V_i \geq PV_1$ and $V_i \leq PV_2$*
  MF_1 ← $(((\frac{-Vi}{PV2}) - PV_1)) + 1$; Output ← ($MF_1$, Degree ($V_i$))
  // Compute the degree of element affiliated with both domains MF_1 and Subsequent it, as MF_2.
  - *α ← ($V_i - PV_2$) // Define the alpha variable; MF_2 ← $(((\frac{-1}{PV2-PV1})) \times (abs(\alpha))) + 1$*
  - *Output ← (MF_2, Degree (V_i))*
  // Compute the degree of element affiliated with both domains MF_2 and previous it, as MF_1.
  - *Output ← (MF_3, MF_4,…,MF_{m-1}, Degree(V_i))*

//Determining the element's degree of membership across the remaining membership functions. • *Case 3: if $V_i \geq PV_{n-1}$ and $V_i \leq PV_n$*   ○ *Degree (V_i) ← 1*   ○ *Output ← (MF_m, Degree (V_i))*   *// Calculate Membership Function m.*   ○ *Output ← ($MF_1, MF_2, \ldots, MF_{m-1}$, Degree($V_i$))*   *//Determining the element's degree of membership across the remaining membership functions.* 4)End of Algorithm 2

Algorithm 3: Input Partitioning and Membership Classification as similar work as Gaussian MF
//Membership degrees are calculated for each input value V_i with respect to membership functions defined over the universe of discourse. *//The parameters PV (parameter values) is defining the shape, boundaries, or centers of the membership functions—not the input data itself.* • *Input: V, a set of crisp input values for which the degree of membership will be calculated.* *//Parameters: Definitions of membership functions (PVs specifying boundaries, centers, or slopes for Gaussian MFs.* *n: The number of fuzzy partitions (i.e., number of membership functions) into which the universe of discourse is divided.* • *Output:*   *A matrix of membership degrees $\mu(v_i)$ for each v_i across all defined membership functions.* *Procedure:*   1. *Initialization:*   • *Max (V) ← max (Vi) // Calculate the maximum value from the sets V.*   • *σ ← 16339 //Define standard deviation of the Gaussian MF.*   2. *Parameter Value Calculation:*   *PV_1 ← 0; PV_2 ← MAX(V)/2; PV_n ← MAX(V); MF_1 center ← PV_1; MF_2 Center ← PV_2; MF_m Center ← PV_n*   3. *Iterate Over Each Input Value V_i in the Set of Parameter Values:*   *for each $V_i \in V$:* • *Case 1: if $V_i \geq 0$ and $V_i \leq PV_n$*   *MF_1 ← EXP (-(($V_i - PV_1$)^2) /(2* σ ^{2})); Output ← (MF_1, Degree (V_i))*   *// Compute Membership Function 1.*   *Output ← ($MF_2, MF_3, \ldots, MF_{m-1}$, Degree($V_i$))*   *//Determining the element's degree of membership across the remaining membership functions.* • *Case 2: MF_2 ← EXP (-(($V_i - PV_2$)^2) /(2* σ ^{2}))*   *Output ← (MF_2, Degree (V_i)) // Compute Membership Function 2.*   *Output ← ($MF_3, MF_4, \ldots, MF_{m-1}$, Degree($V_i$))*   *//Determining the element's degree of membership across the remaining membership functions*

- *Case 3:* $MF_m \leftarrow EXP\,(-((V_i - PV_m)^2)\,/(2.\,\sigma^{2}))$

  Output $\leftarrow$ *(*MF_m*, Degree (*V_i*)) // Compute Membership Function m.*

  Output $\leftarrow$ *(*$MF_1, MF_2, MF_3, \ldots, MF_{m-1}$*, Degree(*$V_i$*))*

  //Determining the element's degree of membership across the remaining membership functions.

4.End of Algorithm 3

6.5 Experimental Results and Analysis

Our proposed method has been applied to a dataset comprising over 10,000 user tasks of varying sizes, which was extracted from the Parallel Workloads Archive. This archive is a comprehensive repository that contains detailed logs of job-level usage data from large-scale parallel supercomputers, clusters, and grids. The logs encompass crucial information about the size of user tasks, which can vary significantly depending on the specific workload and system specifications. Given that each user base requests the cloud environment to perform its tasks, the data size is measured per request. For further specifics regarding user task sizes, you can explore the raw workload logs and models available on the Parallel Workloads Archive website at[(3)]. In our work. These task sizes are generally random and unstructured, encompassing "small," "medium," and big" The recorded data consists of task sizes measured in bytes, ranging from a minimum of 0 to a maximum of 67170 bytes. This wide range reflects the diverse nature of user activities. The data were obtained directly from the database in their original form without preprocessing. Appendix 6 (Figure 1). depicts the database titles selected for the work. The task column data, specifically identified and prepared for analytical purposes, was systematically extracted from the database to serve as the foundation for the subsequent experimentation, Appendix 6 (Figure 2), shows the tasks before classifying. Operations using the MATLAB® (R2018b) software [174]. This program was selected due to its robust computational capabilities, enabling precise mathematical analysis, data manipulation, and visualization. The processing steps included data filtering and targeted analysis to derive meaningful insights and ensure the integrity of the results.

6.5.1 Determine the Degree of Membership as The Triangular Membership Function

In this context, tasks are classified by size using the proposed method, as outlined in Section 4. To demonstrate this, determine the degree of membership through the triangular membership function by applying the first algorithm to values within the universe discourse. The implementation results are systematically illustrated to demonstrate the classification processes based on fuzzy logic principles. Figure 6.2 presents a classified single triangular membership function, showcasing the initial classification structure with a single membership function type for clarity and precision.

(3)https://www.cs.huji.ac.il/labs/parallel/workload/.

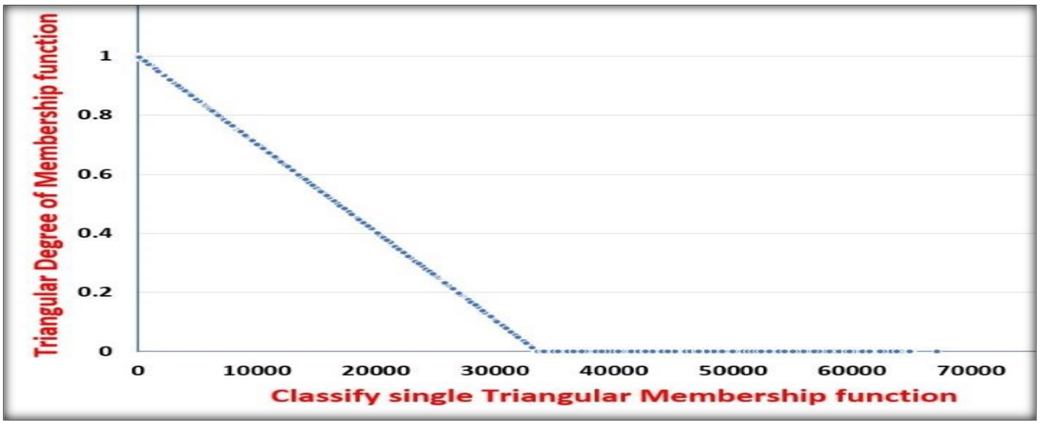


Figure 6.2 Classify single Triangular MF.

Figure 6.3 extends this analysis by depicting the classification of all nested membership functions, emphasizing the hierarchical arrangement and interactions between multiple membership functions within the system. In contrast, Appendix 6 (Figure 2), demonstrates the classification of the membership function achieved through the application of the Mamdani fuzzy logic system, which integrates fuzzy rules and inference mechanisms to produce comprehensive and interpretable classification results. These figures collectively highlight the progressive refinement of membership function classification, illustrating the effectiveness of fuzzy logic systems in managing uncertainty and delivering accurate outcomes.

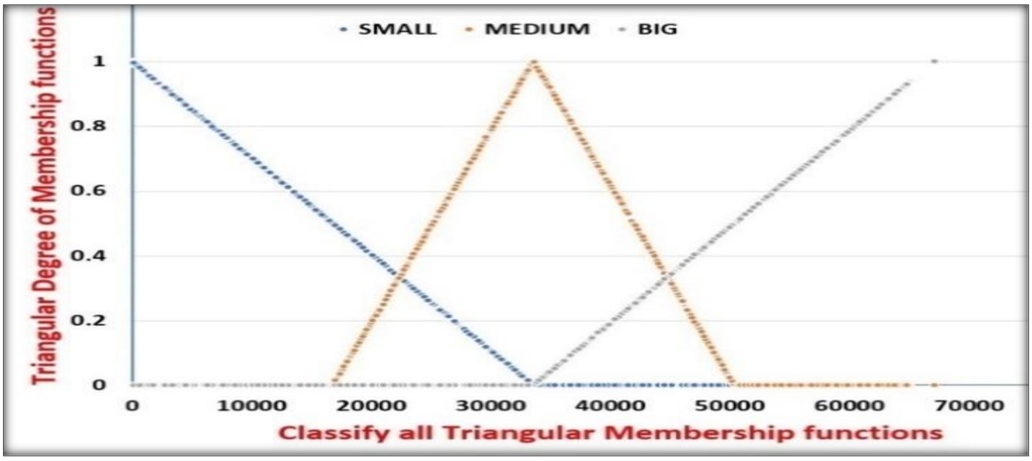


Figure 6.3 Classify all Triangular MF.

6.5.2 Determine the degree of membership as the trapezoidal membership function

In this context, tasks are classified based on their size using the proposed method, as outlined in Section 4. The classification process is achieved by determining the degree of membership through the implementation of a trapezoidal membership function. This function is applied using the second algorithm, which assigns membership values to data points within the defined universe discourse, ensuring a systematic and accurate task classification. The results of this implementation are illustrated in Figures 6.4 and 6.5. Figure 6.4 presents the classification of a single trapezoidal membership function, while Figure 6.5 depicts the classification of all trapezoidal membership functions, demonstrating the effectiveness of the second algorithm in assigning precise membership values. In contrast, Appendix 6 (Figure 3), presents the corresponding Mamdani system membership functions, showcasing the fuzzy inference process and its integration into the classification framework. This detailed analysis highlights the significance of the proposed method and algorithms in accurately determining membership degrees, thereby enabling a precise and meaningful classification of tasks within the system.

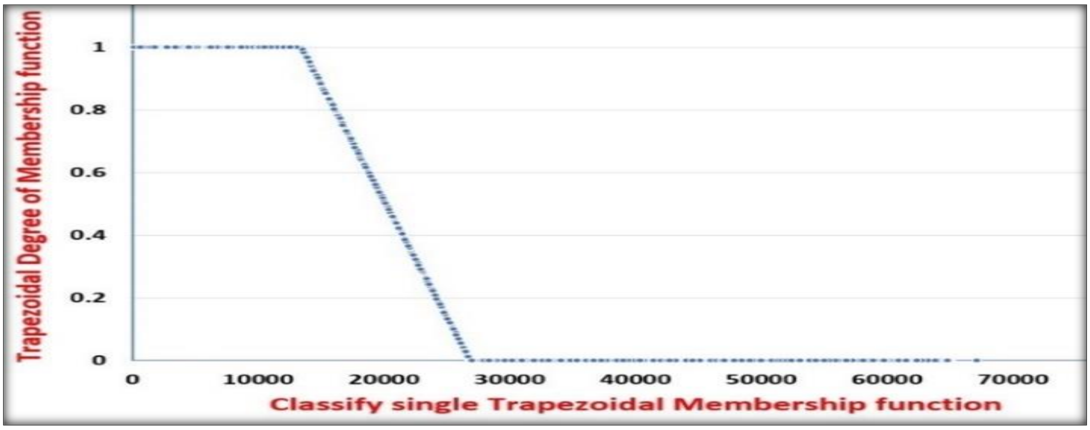


Figure 6.4 Classify single Trapezoidal MF.

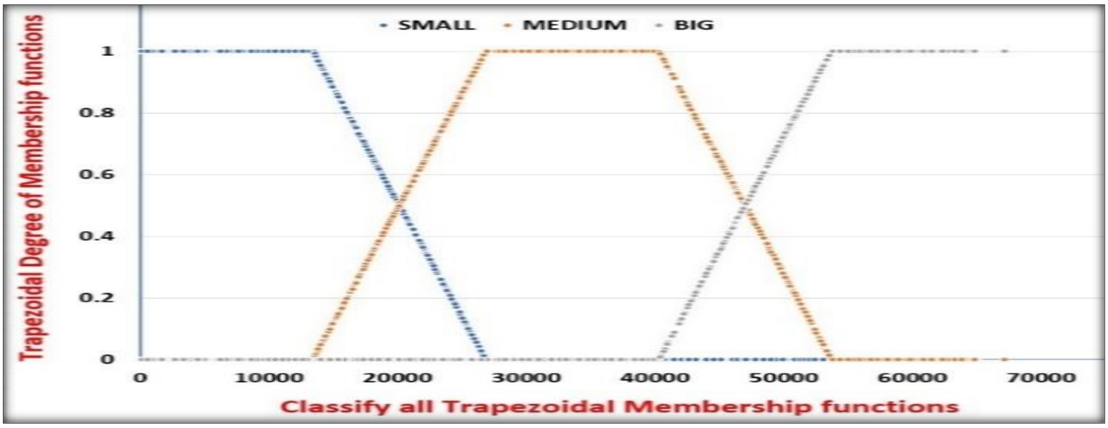


Figure 6.5 Classify all Trapezoidal MF.

6.5.3 Determine the Degree of Membership as The Gaussian Membership Function

In this context, tasks are classified based on their size using the proposed method, as outlined in Section 4. To demonstrate the effectiveness of this approach, the degree of membership is determined using the Gaussian membership function by implementing the third algorithm on values within the defined universe discourse. The Gaussian membership function, chosen for its smooth and continuous nature, ensures precise membership value assignment, facilitating accurate classification of task sizes. The results of this implementation are presented as follows: Figure 6.6 illustrates the classification using a single Gaussian membership function, providing a clear and focused representation of membership values for task sizes. Figure 6.7 expands on this by presenting the classification of all Gaussian membership functions simultaneously, showcasing the system's ability to handle multiple overlapping membership functions effectively. In contrast, Appendix 6 (Figure 4), depicts the classification results using the Mamdani fuzzy system membership functions, highlighting the integration of fuzzy inference rules with membership functions to produce comprehensive, interpretable, and consistent outcomes. These results collectively validate the robustness and flexibility of the proposed method, demonstrating the precision of Gaussian membership functions and the effectiveness in managing uncertainty and enhancing task size classification.

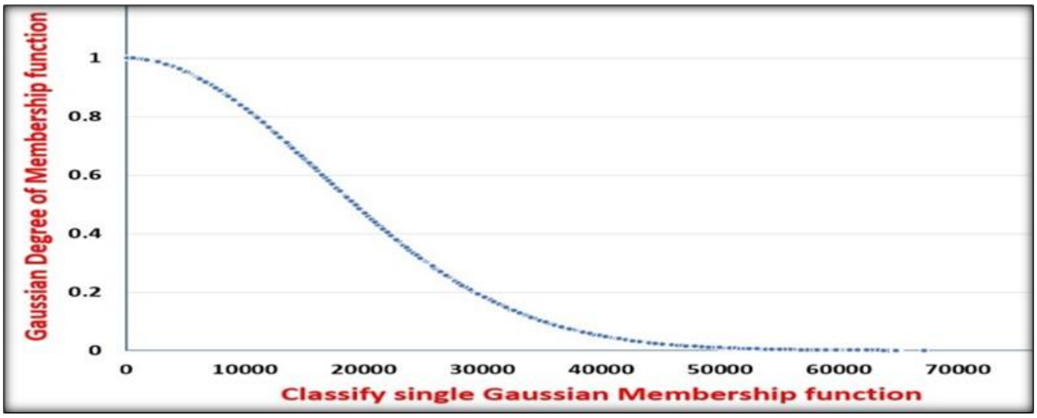


Figure 6.6 Classify single Gaussian MF.

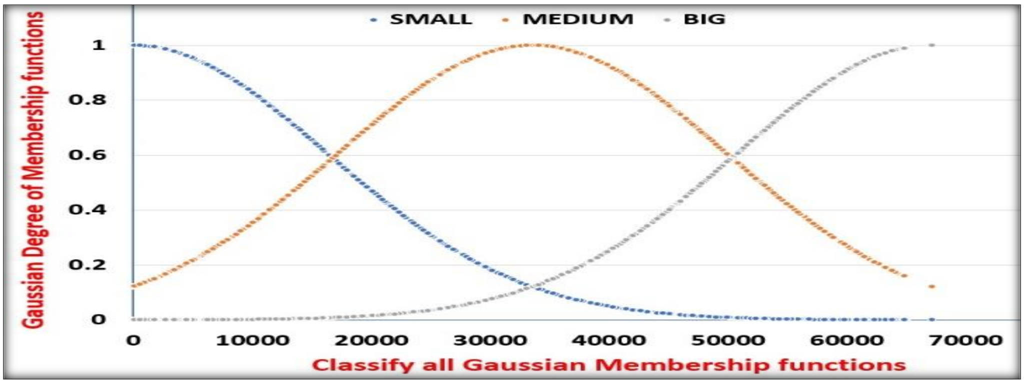


Figure 6.7 Classify all Gaussian MF.

6.5.4 Validation-Based Comparative Analysis of Mamdani FIS and a Proposed Mathematical Model

This study introduces algorithms for systematically classifying input values into fuzzy sets using mathematical methods analogous to standard fuzzy membership functions (triangular, trapezoidal, and Gaussian). These algorithms are integrated within a robust mathematical framework, providing an alternative to the heuristic or manually tuned fuzzy partitions typically employed in Mamdani-based inference systems. The proposed model demonstrates a novel application of standard fuzzy classification algorithms integrated within an optimized mathematical framework, specifically triangular, trapezoidal, and Gaussian membership functions. This innovative integration enhances fuzzy partitions' precision, computational efficiency, and systematic adaptability compared to conventional heuristic-based methods. The algorithms are capable of systematic input classification within the universe of discourse and precise computation of membership degrees. These algorithms are grounded in robust mathematical formulations: Triangular membership functions utilize point-slope line equations, Trapezoidal functions employ linear interpolation techniques, and Gaussian functions are based on probabilistic Gaussian distribution functions. Together, they replicate and enhance the behavior of traditional membership functions while significantly reducing computational overhead. The integration of these analytical methods offers substantial benefits. The proposed algorithms maintain the interpretability of classical fuzzy logic systems while enhancing scalability, computational efficiency, and precision—qualities critical for modern intelligent applications. Moreover, the framework reduces dependency on simulation programs and environments, minimizing the need for extensive storage space, processors, and office software functions. To evaluate the effectiveness of the proposed model, a comparative validation study was conducted using ten representative input samples strategically selected from the universe of discourse. Each input underwent analysis to determine its membership degrees across all relevant functions, with outputs outside the input range assigned zero membership degrees. Results from the proposed mathematical model are detailed in Table 6.1, juxtaposed with outcomes from the classical Mamdani approach in Table 6.2, facilitating direct performance comparison. To further validate the robustness of the proposed method, a comprehensive validation study was conducted using 10,000 input samples representing a wide range of task sizes. The proposed framework exhibits superior adaptability and precision compared to the classical Mamdani system, particularly in managing complex and uncertain inputs. This thorough evaluation reaffirms the method's robustness, computational efficiency, and improved accuracy, thereby significantly contributing to the advancement of intelligent fuzzy classification systems.

Table 6.1 Results of the Proposed Method Applied to Selected Samples.

Samples of Degree of Triangular Membership Function			
value	small	medium	big
0	1	0	0
16823	0.499091856	0.001816287	0
17129	0.489980646	0.020038708	0
17361	0.4830728	0.033854399	0
17579	0.476581807	0.046836385	0
25978	0.226499926	0.547000149	0
26931	0.198124163	0.603751675	0
28842	0.141223761	0.717552479	0
31475	0.062825666	0.874348668	0
33565	0.000595504	0.998808992	0
Samples of Degree of Trapezoidal Membership Function			
value	small	medium	big
20162	0.499181182	0.500818818	0
21582	0.393479232	0.606520768	0
23875	0.222792914	0.777207086	0
25331	0.114411195	0.885588805	0
26846	0.001637636	0.998362364	0
46120	0	0.566919756	0.433080244
45451	0	0.616718773	0.383281227
44329	0	0.700238202	0.299761798
42852	0	0.810183117	0.189816883
40336	0	0.997469108	0.002530892
Samples of Degree of Gaussian Membership Function			
value	small	medium	big
0	1	0.120934543	0.000213895
1	0.999999998	0.120949757	0.000213949
10090	0.826402652	0.355634634	0.002238294
32026	0.146469985	0.995458374	0.098946015
49791	0.009627715	0.611475933	0.567984183

54045	0.004209592	0.456574063	0.724241188
61138	0.000911417	0.241274197	0.934125619
64852	0.000379417	0.160259114	0.989987311
65069	0.000359903	0.156223736	0.991766863
67170	0.000213895	0.120934543	1

Table 6.2 Results of the Traditional Method Applied to Selected Samples.

Samples of Degree of Triangular Membership Function			
value	small	medium	big
0	1	0	0
16823	0.499076941,400667	0.001846117,1986660315	0
17129	0.489965459,74273464	0.020069080,51453073	0
17361	0.483057408,28966176	0.033885183,420676514	0
17579	0.476566222,01048116	0.046867555,979037634	0
25978	0.226476893,75893282	0.547046212,4821344	0
26931	0.198100285,8504	0.603799428,2991901	0
28842	0.141198189,61410197	0.717603620,7717961	0
31475	0.062797760,83849452	0.87440447,83230109	0
33565	0.000565745,5931395903	0.998868508,8137208	0
Samples of Degree of Trapezoidal Membership Function			
value	small	medium	big
20162	0.499181182,07533124	0.500818817,9246688	0
This table extends and complements the information presented in Table 6.2.			
21582	0.393479231,7999107	0.606520768,2000894	0
23875	0.222792913,50305197	0.777207086,496948	0
25331	0.114411195,47417002	0.885588804,52583	0
26846	0.001637635,849337502	0.998362364,1506625	0
46120	0	0.783443757,9096255	0.216556242,0903744 4
45451	0	0.808345120,2263083	0.19165487,97736916 7
44329	0	0.850107943,1251396	0.149892056,8748604 3
42852	0	0.905084493,4117472	0.094915506,5882528
40336	0	0.998734459,9121566	0.001265540,0878433 707
Samples of Degree of Gaussian Membership Function			
value	small	medium	big
0	1	0.122	0.0002
1	1	0.122	0.0002
10090	0.8418	0.7201	0.0053
32026	0.2931	0.996	0.1097
49791	0.0304	0.5364	0.7211
54045	0.0124	0.2917	0.8431
61138	0.0028	0.1097	0.9959
64852	0.0011	0.0566	0.9881
65069	0.0010	0.0532	0.9926
67170	0.0002	0.1218	1

6.6 Summary

This chapter introduced and validated a novel mathematical framework designed to enhance decision-making under uncertainty by providing precise fuzzy classification. The primary contribution lies in systematically classifying input values into predefined fuzzy sets—specifically, triangular, trapezoidal, and Gaussian membership functions—to significantly enhance accuracy and computational efficiency in determining membership degrees. The developed methodology integrates three optimized algorithms mathematically aligned with traditional fuzzy logic membership functions. These algorithms facilitate systematic input partitioning and precise computation of membership degrees, ensuring clear differentiation among distinct membership levels (small, medium, and big). Compared to traditional Mamdani FIS, our approach delivers more accurate, computationally efficient, and robust results while preserving interpretability and simplicity crucial for broad practical adoption. Extensive validation using over 10,000 user-task-size samples confirmed that the proposed algorithms consistently match or surpass the performance of the traditional Mamdani method. Our model efficiently manages distinct and overlapping fuzzy set classifications, underscoring improved flexibility and precision.

The main contributions of this chapter include:

- A novel mathematical model enables precise input classification through triangular, trapezoidal, and Gaussian membership functions.
- Algorithmic innovation through developing three original algorithms leveraging rigorous mathematical formulations to optimize fuzzy classification.
- Enhanced computational efficiency, significantly reducing computational overhead without compromising accuracy or interpretability.
- A robust comparative analysis demonstrates the proposed methodology's superior flexibility and effectiveness against traditional Mamdani-based fuzzy logic systems.

The demonstrated effectiveness of this methodology highlights its potential applicability across diverse AI domains, notably in QoS categorization. Looking ahead, this chapter establishes a foundational model beneficial for future research endeavors, especially in real-time decision-making contexts requiring high precision and scalability, such as healthcare diagnostics, financial forecasting, and cloud computing environments. Future work will expand this methodology's application within the (IVCBS), directly addressing QoS scalability and classification accuracy challenges and further validating the model’s suitability in practical, real-world decision-making scenarios.

Chapter 7 Intelligent Validation Cloud Broker System

Chapter 7 contributes to the (IVCBS), which enhances SLA selection. In this work, the classification algorithm uses mathematical formulations like trapezoidal membership functions to assign fuzzy membership degrees to input values. These formulations mimic the shape and behavior of trapezoidal membership functions by employing linear equations to define ascending, plateau, and descending regions across the universe of discourse. Thus, the classification of VM resources and user request sizes is performed using a method that mathematically resembles trapezoidal membership functions, improves decision-making, reduces data centre processing time, and lowers VM costs. Simulations show that IVCBS, using the "Optimize Response Time" policy, outperforms traditional methods in response time, VM cost, and energy efficiency. This system also reduces data transfer costs and enhances power usage efficiency by improving data center request servicing times, offering a more efficient and cost-effective approach to cloud resource management.

7.1 Overview of SLA Selection and the IVCBS Framework

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 [170]. 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 [171]. 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 AWS, Google App Engine, Microsoft's Azure Services Platform, 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 [172]. The dynamic nature of cloud computing necessitates efficient resource allocation, which can be challenging due to potential resource shortages and conflicting interests between (CSPs) and (CSUs). SLA negotiations can mitigate these issues, and the proposed broker-based mediation framework optimizes these negotiations [173]. 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 [174]. Selecting the most suitable resources to meet diverse user demands is a significant research challenge. 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 pre-classify resources, reducing computational costs [175]. Fuzzy logic-based optimization algorithms present Fuzzy-RLVMrB and Fuzzy-MOVMrB, designed to balance horizontal and vertical loads across (PMs) by managing processor, BW, and memory resources. Simulations demonstrate that these algorithms excel in LB and energy efficiency compared to other methods [176]. Performance and Resource-Aware VM Selection using Fuzzy in Cloud Environment (PRSF) develops a VM 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 [177]. 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 [178]. Simultaneously, Cloud Analyst is a simulation tool extended from CloudSim. LB 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 LB 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 [179]. 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 [180]. Building upon the methodologies discussed in prior studies, which focus on enhancing decision-making accuracy, this research advances solutions to the identified challenges within this thesis. The study introduces the IVCBS aimed at optimizing the allocation of AWS-EC2 resources based on user demands. Key AWS-EC2 specifications, such as VCPUs, RAM, storage, and BW, collectively influence VM costs, power consumption, and processing times, impacting user confidence and decision-making in selecting (SLAs) that align with budgetary and performance needs. The study addresses a scenario involving one million customers entering a cloud environment, each presenting varying demands, utilizing real-world data from diverse datasets, with a particular emphasis on 11 types of AWS-General Purpose EC2 Instances. Employing MATLAB, an algorithm was developed to classify and organize EC2 resources. Furthermore, user demand sizes were categorized using a proposed mathematical model employing five membership functions: Poor, Fair, Good, Very Good, and Excellent, structured like the Trapezoidal Membership Function. The proposed mathematical framework calculates membership degrees for each input value—such as VM attributes (e.g., vCPU count, RAM size, storage capacity, BW) and user request sizes—using equations that emulate trapezoidal membership functions. These membership degrees, ranging between 0 and 1, represent the degree to which each input belongs to fuzzy linguistic categories such as Poor, Fair, Good, Very Good, or Excellent. This process enables precise and consistent classification of both cloud resources and user workloads, supporting effective decision-making in resource allocation. The IVCBS operates in two stages. In the first stage, it calculates continuous membership degrees ranging from 0 to 1 for inputs such as VM attributes and user request sizes. This computation relies on mathematical models that emulate the behavior of trapezoidal membership functions, quantifying the degree to which each input belongs to categories such as Poor, Fair, or Excellent. In the second stage, these continuous values are converted into a binary membership score (either 1 or 0) based on a defined threshold, simplifying real-time decision-making. Only inputs assigned a score of 1 are validated for resource allocation. This method preserves the precision of fuzzy classification while maintaining operational efficiency. This binary criterion ensures simplicity and operational efficiency by eliminating the need to manage intermediate fuzzy values during resource allocation. Specifically, if the computed fuzzy membership value exceeds the predefined threshold, the decision to allocate the resource is validated (membership score = 1); otherwise, the allocation is disregarded (membership score = 0). Thus, although continuous membership values derived from trapezoidal membership functions effectively capture nuanced, fuzzy categorizations of resources and user requests, the IVCBS strategically converts these continuous values into binary membership scores for practical real-time cloud resource allocation. Consequently, the system effectively integrates fuzzy logic principles for initial classification and categorization with crisp decision-making, ensuring efficient, straightforward, and transparent resource validation and allocation. However, the proposed algorithm categorizes AWS EC2 cloud computing resources and user request sizes based on linguistic variables, where a membership score of 1 denotes the highest relevance. This score serves as a validation criterion through broker validation processes. For example, CPU resources falling within specified values (vCPU: 1, 2, 4) are classified as 'Poor' according to the algorithm, driven by their membership score 1, aligning firmly with the 'Poor' membership function. Similarly, user request sizes categorized within ranges (3, 5, 10) MB also receive a membership score of 1, confirming their classification within the 'Poor' category. This systematic approach extends across all data in the 'Poor' membership function domain, maintaining the same principle for the remaining four membership functions, focusing exclusively on values assigned a score of 1. Subsequently, the second algorithm, the matching algorithm, plays a pivotal role in the broker validation process by verifying whether all system metrics attain a membership score of 1. VM-EC2 resources are allocated to execute user requests when this condition is met. Conversely, if the score is 0, the matching process is disregarded. This streamlined methodology ensures efficient allocation of VM-EC2 resources based on validated criteria. The matching process validates all values derived from the algorithm, ensuring that each classification scenario defined by the five membership functions, whether for EC2 criteria or user request sizes, achieves a score of 1. Upon validation, the broker initiates the allocation process, assigning an EC2 VM to execute user requests effectively. Expanding the scope, the study distributes user requests across data centers in six geographic regions (North America (R0), South America (R1), Europe (R2), Asia Pacific (R3), Africa (R4), and Australia (R5)). It compares the performance of the traditional method with the (IVCBS). Using Cloud Analyst tools, two distinct broker policies were evaluated: the Optimize Response Time Policy, directing requests globally, and the Dynamic Reconfigure with Load Service Broker Policy, routing requests within users' regions. Across 11 scenarios involving one million users, simulations across 31 AWS data centers demonstrated the superiority of IVCBS, particularly with the Optimize Response Time policy, over the Dynamic Reconfiguration with Load policy. IVCBS consistently exhibited superior performance metrics, including overall response time, processing efficiency, total VM cost, and Data Center Request Servicing Times, highlighting its efficacy in enhancing cloud computing efficiency across diverse global environments.

7.2 Limitations of Traditional Methods and Advances in Intelligent Decision-Making

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 [181]. Qazi et al. [2] 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. [182] 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 work provides in-depth discussions to enhance the comprehension of cloud brokers in multi-cloud environments. Yao et al. Ahmad et al. [183] introduce the (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. [184] 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 CSPs, cloud service brokers, and customers. Any issues arising within any of these parties will undoubtedly impact the broker's performance. Cinar et al. [185] 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 [186] 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 CSPs. Chafai et al. [187] This work 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. [188] 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 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 work does not cover the decision-making strategies for purchasing and selling services. Al-E'mari et al. [22] 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. Ahmed I. El Karadawy et al. [189] conducted a detailed examination of the cloud analyst simulator, focusing on different (LB) algorithms and service broker policies. They specifically evaluated three unique LB algorithms: (RR), throttled, and (ESCE). Sunny Nandwani et al. [190] examined various service broker policies and (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.

7.3 Proposed System

The proposed study centers on intelligently identifying cloud services through rigorous validation. This process ensures uniform attainment of a value of 1 across all outcomes from the classification algorithm, applicable to resource allocation and user request sizes, as discussed earlier. By maintaining this consistent criterion, the study assures the reliability and accuracy of the classification algorithm's outputs, thereby optimizing resource management and enhancing service efficiency in cloud computing environments. This systematic and uniform validation approach highlights its critical role in achieving precise identification of high-quality cloud services. Figure 7.1 depicts the proposed system.

7.3.1 Extraction information Factors from AWS Cloud Environment

Within the AWS cloud environment, users have access to a variety of service instance types, including General Purpose[(4)], Compute Optimized, Memory-Optimized, Accelerated Computing, and Storage-Optimized, all falling under the broad category of 'XaaS'. 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[(5)], ensuring robust global infrastructure and service availability.

7.3.2 AWS General-Purpose Instance Types

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 [191]. By sharing certain standardized features, these EC2 instances are grouped into 11 categories based on similarities in their specifications. Tables 7.1 and Appendix 7 (Table 1), highlight the adopted AWS-EC2 families' specifications. while Appendix 7 (Table 2), lists the actual on-demand cost of each EC2 device, as indicated on AWS's official pricing page[(6)]. Table 7.2 displays the number of customers entering the cloud for each scenario and the sizes of their requests.

(4) (https://aws.amazon.com/ec2/instance-types/).
(5)(https://aws.amazon.com/about-aws/global-infrastructure/regions_az/).
(6) (https://aws.amazon.com/ec2/pricing/on-demand/).

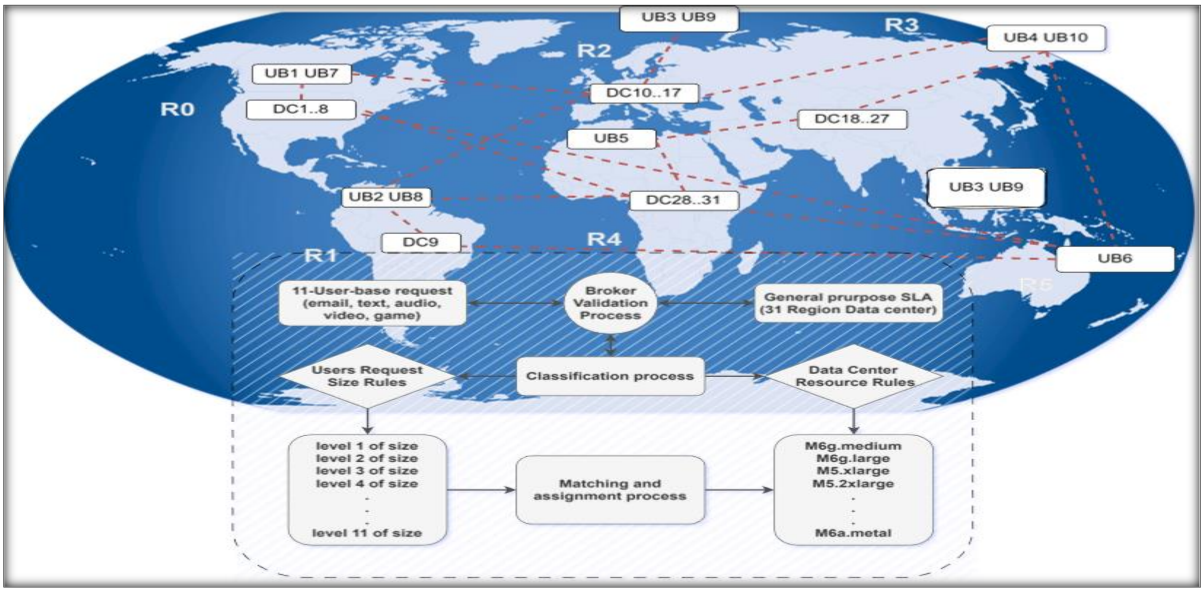


Figure 7.1 Intelligent Validation Cloud Broker System Framework.

Table 7.1 AWS-General purpose instance features.

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

7.3.3 Theoretical Framework and Methodology

7.3.3.1 Mathematical Modeling in the Intelligent Validation Cloud Broker System (IVCBS)

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 [192]. In the (IVCBS), this intelligence is utilized through optimization algorithms rooted in a mathematical model influenced by the trapezoidal membership function. Implementing this model generates membership scores of 1 and 0 for the input values across all proposed membership functions within the system's universe of discourse. This approach significantly improves SLA selection and enhances overall system efficiency.

Table 7.2 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

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 7.2 demonstrates how the mathematical approach closely reflects the characteristics of a trapezoidal membership function, particularly in determining and generating degrees of membership or belonging. The equations and concepts presented in this figure provide the foundation for the outcomes produced by the algorithms detailed in Table 7.3. The behavior of the mathematical model as a membership function, which classifies and assigns membership levels to input values within the proposed system, can be effectively illustrated using equations that relate to point-slope lines and absolute values, as discussed in Chapter Six.

$$y = mx + c \tag{7.1}$$

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:

$$y - y1 = m(x - x1) \tag{7.2}$$

In this equation, (x, y) denotes an arbitrary point on the line [140][164]. The mathematical model employed in the IVCBS is classifies and arranges (VM) resources (e.g., VCPU, RAM, Storage, BW) and user request sizes. This model defines mathematical functions (Poor, Fair, Good, Very Good, and Excellent) similar to the trapezoidal membership function. These functions are used to classify and determine the membership degree for each input value within the discourse universe, evaluating the suitability of EC2 selections that adapt to client SLA criteria. The classification outcomes directly influence the decision-making process for validating the broker mechanism. A result of (1) indicates an effective decision, while (0) suggests exclusion. This section introduces a novel model to explore the intelligent features integrated into the (IVCBS).

It focuses on the intricate management of VCPU resources, using them as a key example. This rigorous method is consistently applied to all VM-EC2 resources and user request sizes, ensuring SLA-level classification uniformity and reliability. The MATLAB script demonstrates how this approach reinforces the consistency of resource allocation within the system. Furthermore, to illustrate the alignment of the mathematical model with the proposed membership functions, this approach has been integrated into the discussion on initializing and visualizing the membership function, as depicted in Appendix 7 (Figures 1 and 2).

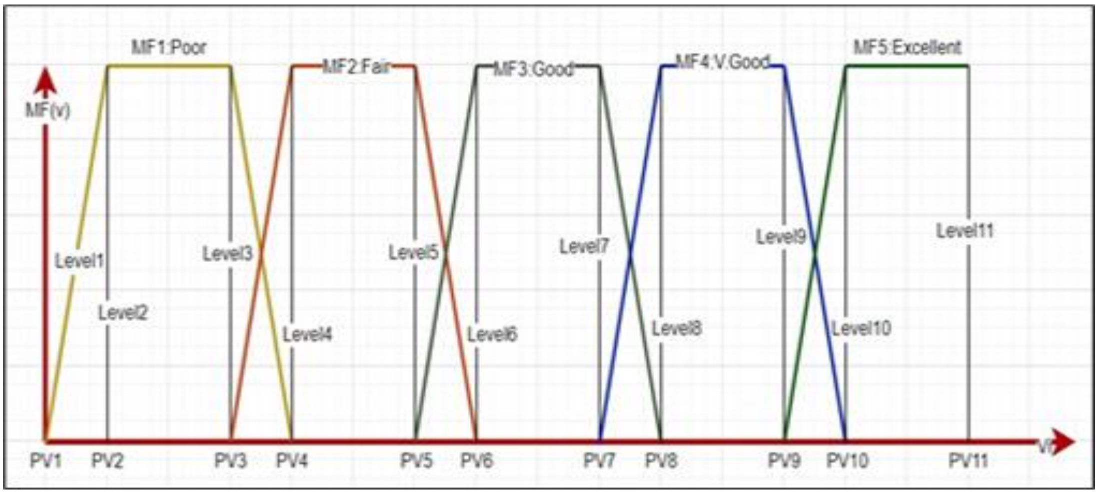


Figure 7.2 Fuzzy Partition Using Mathematical Model.

7.3.3.2 Modeling and Implementing Algorithms in the Intelligent Validation Cloud Broker System (IVCBS)

This section addresses the handling of ten user-base requests, employing the round-robin algorithm to evenly distribute workloads across VM 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, applied the CloudAnalyst tool under a designated SBP in two distinct scenarios (optimizing response time and dynamically reconfiguring based on load).

Eq. (7.3) is given by n as the number of sets for the load (L) or requests that need to be scheduled to servers.

$$L = \{L_1 , L_2 , L_3 , \ldots , L_n \} \tag{7.3}$$

This equation is coherent in indexing because it uses sequential indices $1,2,3,\ldots,n$ to denote each element L_i The indexing starts from 1 and progresses sequentially up to n.

Eq. (7.4) DC represents a set of data centers, with $dc_1, dc_2, dc_3, \ldots, dc_k$ denoting each data center indexed from 1 to k.

$$\mathrm{DC} = \{\mathrm{dc_1}\ , \mathrm{dc_2}\ , \mathrm{dc_3}\ , \ldots . , \mathrm{dc_k}\} \tag{7.4}$$

This equation is coherent as well. It uses indices *1,2,3,…,k* to denote each data center dc_i. Similar to Equation (7.3), the indexing starts from 1 and proceeds sequentially up to *k*, maintaining a consistent and logical index structure.

The following equation (7.5) For each data center dc_i, there is a single VM_i associated with it.

$$\mathrm{dc_i} = \{\mathrm{VM_i}\} \tag{7.5}$$

This equation introduces *i* as the index for VMs within each data center dc_i. It is coherent because it specifies that dci has exactly one VM_i, ensuring clarity and specificity in indexing.

Eq. (7.6) DCs_L represents the load of each VM_i in the data centers.

$$\mathrm{DCs_L} = \{\mathrm{VM_1\ L}\ , \mathrm{VM_2 L}\ , \mathrm{VM_3\ L}\ , \ldots , \mathrm{VM_k\ L}\} \tag{7.6}$$

This equation uses *i* from *1* to *k* to denote each VM_i and its associated load L. The indexing is coherent as it sequentially lists VM_iL for each VM within the data centers.

Eq. (7.7) This equation indicates that the load L of each VM_i in the data centers 1,2,…,k is approximately equal. It uses i from 1 to k to represent each VM_i.

$$\mathrm{VM_1\ L} \approx \mathrm{VM_2\ L} \approx \mathrm{VM_3\ L}, \ldots , \mathrm{VM_k\ L} \tag{7.7}$$

Eq. (7.8) t_0 calculates the time required to allocate all tasks L to each VM_i, where τ_{0i}, represents the time τ_0 required to execute each task L_i.

$$\mathrm{t_0} = \sum_{i=1}^{n} \tau_{0i} \tag{7.8}$$

Where:

i: Represents the index for tasks, consistent with Equation (7.3) where L_i denotes each task or load.

Eq. (7.9) This equation defines VM as a set containing k VMs within a specific data center. It describes how, when multiple VMs are available (denoted by k), all tasks can be evenly distributed among them for execution. This equation clarifies the method of task distribution across multiple VMs, highlighting the shared allocation approach in cloud computing environments.

$$\mathrm{VM} = (\mathrm{VM_1}\ , \mathrm{VM_2}_{,} \mathrm{VM_3}_{,} \ldots , \mathrm{VM_k}_{,}) \tag{7.9}$$

Eq. (7.10) shows that the total execution time T_0 is the sum of the execution times T_i for each task *i* executed on the total number of VMs *n* in the data center:

$$\mathrm{T_0} = \sum_{i=1}^{n} \mathrm{T}_i \tag{7.10}$$

This equation indicates that T_0 represents the cumulative execution time across all tasks executed on *n* VMs within the specific data center.

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.End
19.IF(V>PV4 and V<=PV5)
20.MF2 ←1
21.Output ← (Fair, MF2)
22.Output ← ((Poor, Good, V. Good, Excellent),0)
23.End
24.IF(V>PV5 and V<=PV6)
*25.MF2 ← (((-1/(PV6-PV5)) *((V-PV5))) +1)*
26.Outputç ←(Fair, MF2)
27.Output ← ((Poor, V. Good, Excellent),0)
*28.MF3 ← (((-1/PV5-PV6)) *((V-PV6))) +1)*
29.Output ← (Good, MF3)
30.Output ← ((Poor, V. Good, Excellent),0)
31.End
32.IF (V>PV6 and V<=PV7)
33.MF3 ←1
34.Output ←(Good, MF3)
35.Output ← ((Poor, Fair, V.Good, Excellent),0)
36.End
37.IF (V>PV7 and V <=PV8)
*38.MF3 ← (((-1/(PV8-PV7)) *((V-PV7))) +1)*
39.Output ← (Good, MF3)
40.Output ← (Poor, Fair, Excellent),0)
*41.MF4 ← (((-1/(PV7-PV8)) *((V-PV8))) +1)*
42.Output ← (V. Good, MF4)
43.Output ←(Poor, Fair, Excellent,0)
44.End

```
45. IF (V>PV8 and V<=PV9)
46. MF4←1
47.Output←(V. Good, MF4)
48.Output← ((Poor, Fair, Good, Excellent),0)
49.End
50.IF (V>PV9 and V<=PV10)
51.MF4← (((-1/(PV10-PV9)) *((V-PV9))) +1)
52.Output← (V. Good, MF4)
53.Output← ((Poor, Fair, Good),0)
54.MF5← (((-1/(PV9-PV10)) *((V-PV10))) +1)
55.Output← (Excellent, MF5)
56.Output← ((Poor, Fair, Good),0)
57.End
58.IF (V>PV10 and V<=PV11)
59.MF5←1
60.Output← (Excellent, MF5)
61.Output← (Poor, Fair, Good, V.Good),0)
62.End
63.End
```

Matching Algorithm

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

27.End
28.IF Output (Excellent, PV10)
29.Assign: User base request (App10) ← M6i.metal
30.End
31.IF Output (Excellent, PV11)
32.Assign: User base request (App11) ← M6a.metal
33.End

7.3.3.3 Cloud Analyst Simulation Framework

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 BW. 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 [193]. Figure 7.3 depicts the cloud analyst model.

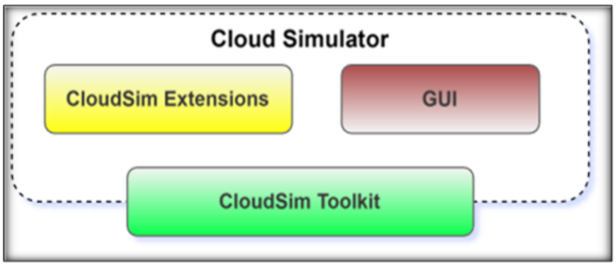


Figure 7.3 Cloud Analyst Model.

7.3.3.4 Round Robin Algorithm

The RR 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 ($1/n$), where n represents the number of processes in the queue. Thus, the wait time for any process is limited to ($n-1$) times the quantum length, q, ensuring a fair and efficient distribution of processing time [194] [195].

7.3.3.5 Service Brokering Strategies

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 [196]. This intermediary position enhances the connection between customers and CSPs [197]. Through the (SBP), services are dynamically distributed between the cloud's infrastructure and its service providers [198], effectively guiding the selection of data centers [196]. The assignment of VMs to physical hardware in data centers, a process critical to the data center broker known as VM deployment, underscores the importance of the SBP [199]. 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 [196]. Our analysis involved adopting two foundational broker strategies and examining and contrasting their effectiveness.[200]. The primary policy focuses on optimizing response time, where the service broker evaluates essential attributes of data centers to gauge their performance [189]. This approach ensures the quickest possible response times for end-users during queries [201]. 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 [202]. VMs are utilized to handle customer requests swiftly, enhancing point-to-point communication [203]. This policy assumes uniform processing requirements and execution times for all requests [204]. The secondary policy involves dynamic reconfiguration based on load, where the service broker manages scalability for cloud applications [189]. This involves the service broker dynamically reconfiguring and altering the VMs within data centers to match demand [201]. 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 [178]. This method calculates retention times to achieve the longest cycle time recorded, addressing both cost and performance expectations of users [204] and adjusting the number of VMs as needed [205].

7.4 Experimentation and analysis

7.4.1 Simulation the proposed system

To test our proposed policy, 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 VM, 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, standardized the network delay metrics from AWS latency monitoring(7), shown in Appendix 7 (Table 3), and set advanced data center configurations for all tests, as detailed below. Table 7.4 displays data related to a Single User Base, which becomes pertinent in Table 7.5 as our research encompasses 11 analogous instances derived from this single-user base, varying according to the magnitude of user requests, 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, measured hourly (60.0). It's posited that the upper limit of users from each user base cluster

(7) (https://www.cloudping.co/grid).

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:

$$\text{Avg peak users} = \frac{\text{Total User Count}}{\text{10 UB}} \quad (7.11)$$

$$\text{Avg Off} - \text{ peak users} = \frac{\text{Avg Peak users}}{10} \quad (7.12)$$

The data size per request (in bytes) and the instruction length per request (in bytes) were determined by applying mathematical formulas No. 7.12 and No. 7.13, 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 VMs within a single data center.

$$\text{Data size per request} = \frac{\text{Total UB request}}{\text{Avg peak users}} \quad (7.13)$$

$$\text{Executable length} = \frac{\text{Total UB request}}{\text{10 UBs}} \quad (7.14)$$

Appendix 7 (Table 4) 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 Appendix 7 (Table 5). The pricing is based on data transferred "in" to and "out" of Amazon EC2[(8)]. In our study, contrasted the proposed (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 (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 study provides a comprehensive description of these traditional allocation strategies in Appendix 7 (Table 6). It is critical to note that the specifications of the EC2 instances utilized in these traditional methods are identical to those employed in the (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, apply this distribution in 11 different scenarios, tailored according to the number of user request sizes identified in this study.

Table 7.3 Results of the Proposed Algorithm.

No.	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

(8) https://aws.amazon.com/ec2/pricing/on-demand/.

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

Table 7.4 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 7.5 (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

7.4.2 Results and Comparative Analysis

7.4.2.1 Implementation of IVCBS with two Service Broker Policies

In the proposed methodology, IVCBS utilizes either the Optimized Response Time SBP or the Dynamic Reconfiguration with LB approach, both supported by the Cloud Analyst simulator. IVCBS employs these policies to route user requests from User Bases (UBs) to AWS 31 data centers worldwide. This router ensures that each data center adheres to predefined parameters tailored to the request volumes of each UB user group, by IVCBS, as detailed in Appendix 7, Table 5. Specifically, resources such as EC2-M6a.metal are optimized for handling high-volume user requests effectively. For instance, the allocation of VM-Cost is optimized to effectively address user requirements, with resources like EC2-M6a.metal specifically designated for handling high-volume user requests. 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 VM 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. To provide a clear comparison, Table 7.6 showcases the results of implementing the IVCBS method with the Optimized Response Time Service Broker Policy, while Table 7.7 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. Appendix 7 (Figure 3), Presents the results of the IVCBS response time by region under the Optimize Response Time policy. The study explores the implementation of IVCBS with two distinct Service Broker Policies: The Optimized Response Time SBP and the Dynamic Reconfiguration with LB approach. It assesses regional average response times for ten user bases, emphasizing the effectiveness of IVCBS's Optimized Response Time Policy. This policy ensures even distribution of user requests across AWS data centers globally, irrespective of geographic proximity, consistently achieving reduced response times compared to the Dynamic Reconfiguration Policy. Appendix 7 (Figure 4) details the outcomes of the Dynamic Reconfiguration Policy, which directs user requests to data centers located in the same geographic region as the users, aiming to minimize latency under the IVCBS framework. 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 [206] [207]. Our observations indicate that the (IVCBS), when implemented with an optimized response time policy, significantly outperforms the dynamic reconfiguration policy. This superiority is clearly demonstrated through the comparative analysis presented in Appendix 7 (Figures 5 and 6). These results illustrate the superior performance of the optimized response time policy in managing the Average Data Center Request Servicing Time, thereby enhancing energy efficiency. Earlier findings indicated that systems employing IVCBS with a dynamically reconfigured load-balancing broker policy, as shown in Appendix 7 (Figure 7), exhibit different performance characteristics compared to those using IVCBS optimized specifically for response times. Appendix 7 (Figure 8) presents the routing strategy implemented under the optimized response time policy. This performance variance primarily arises from the dynamics introduced by the reconfiguration process 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 available VMs 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 RTT and more efficient processing. Moreover, our analysis is grounded in Amazon's real-world distribution of data center locations globally, utilizing eight (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 larger requests of 3 GB are directed to more robust machines like the M6a.metal.

Table 7.6 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

Table 7.7 Implementing IVCBS with Dynamic Reconfiguration 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

7.4.2.2 Traditional methods

This approach starkly contrasts with the intelligent methodology implemented by IVCBS. In both the Optimize Response Time - SBP and the Dynamic Reconfiguration with Load Balancing, user requests of varying sizes are randomly distributed across the 31 data centers without consideration for the specific type and specifications of the EC2 VMs. There is no structured allocation across all DC-VMs. DC-VMs process requests with diverse parameters that lack uniformity and fail to align with the request volumes of each user group (UBs), as detailed in Appendix 7, Table 6. For instance, the high VM cost is expensive for users whose task requirements are minor, thus failing to meet their basic needs adequately. Additionally, resources like EC2-M6a.metal are allocated to execute small user requests that EC2-M6g.medium could more efficiently handle. The setup and configuration of DCs for both methodologies are facilitated by the CloudAnalyst simulation environment, outlined in Appendix 7 (Table 4). This environment allows for configuring AWS-31 DC metrics, which differ between the proposed and traditional methods. These metrics include VM cost, vCPUs count, storage, RAM, and BW. There are 11 scenarios in both methods, similar in setup but differing in the numerical configuration of metrics for each EC2 instance. Employing the optimized response time policy resulted in a higher average overall response time, average data center processing time, and total VM cost than our proposed IVCBS method. However, it was 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 7.8. When evaluating the results from applying the dynamic reconfiguration policy with traditional methods, as detailed in Table 7.9, 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 than the IVCBS approach. Additionally, Appendix 7 (Figure 9) displays the regional average response times for the 10 user bases, showcasing the performance of the traditional Optimized Response Time Policy. Meanwhile, Appendix 7 (Figure 10) visualizes the regional average response times under the dynamic reconfiguration with load policy. Both figures highlight that these traditional methods were less effective than the results of the proposed IVCBS method. Furthermore, Appendix 7 (Figure 12) 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. This enhances the IVCBS's effectiveness, demonstrating its potential to accommodate future growth in cloud systems while ensuring efficient and cost-effective user request processing within the cloud computing environment. Simultaneously, Appendix 7 (Figure 11) displays the average Data Center Request Servicing Time results across the 31 data centers in our study, applied in 11 different scenarios using the traditional Optimized Response Time Policy.

Table 7. 8 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 7.9 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

7.5 Summary

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 (VM) attributes to match user request sizes and evenly distributing workloads as per SLA standards. This approach considers the current and future workloads and the available resources on each AWS-EC2 instance, aiming to distribute user request across VM uniformly to ensure balanced system utilization and avoid over- or underutilization. A significant part of the study introduces the (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 BW, 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. The (IVCBS), evaluated using the Cloud Analyst simulator, demonstrated notable improvements compared to existing policies. The adoption of a throttled LB policy could further enhance the system's effectiveness, highlighting its potential to support future growth in cloud systems while ensuring the efficient and cost-effective processing of user requests within the cloud computing environment. This approach can be expanded upon in the next contribution of this thesis. Specifically, incorporating job size and classifying the workload into performance-optimized routing policies lead to significant improvements in both response and processing times in cloud systems. This addition provides a critical layer of optimization that directly impacts key performance metrics, including response and processing times, which are integral to cloud system efficiency. Furthermore, the introduction of the throttled LB policy serves as a natural extension of the proposed approach, facilitating more efficient workload management and distribution, particularly during peak demand periods.

Chapter 8 A Broker-Driven Approach Integrating Fuzzy Logic for Optimizing Virtual Machine Allocation

Chapter 8 introduces a broker-driven approach integrating fuzzy logic to optimize (VM) allocation in cloud environments. This method dynamically adjusts VM distribution based on incoming request packet sizes and CPU utilization. It utilizes Google's General-purpose machine family for Compute Engine - T2D standard machine types, configured with specifications including VCPU, RAM (GB), Storage (GB), BW (GBPS), and Price per hour ($), as applied in this study. Employing fuzzy logic, this system intelligently assigns VMs to user requests within the user base, ensuring alignment with appropriate sizes and cost considerations for the allocated VMs. In contrast, the traditional method relies on random VM allocation, disregarding user request sizes and assigning available VMs arbitrarily to execute tasks.

8.1 Advancements in Packet Size Optimizations Cloud Service Delivery

In the realm of cloud computing, the efficient allocation of (VMs) is paramount for optimizing resource utilization and ensuring high performance. The rapid proliferation of cloud services has necessitated sophisticated strategies to manage the dynamic and heterogeneous nature of cloud workloads. Traditional methods, which often prioritize metrics such as CPU, memory, and storage capacities, frequently overlook the varying sizes of request packets. This oversight can lead to suboptimal resource usage and potential performance bottlenecks, thereby hindering the overall efficiency and responsiveness of cloud services [208][209]. The complexity of cloud environments requires innovative approaches to VM allocation that can adapt to fluctuating workloads and diverse user demands. Recent advancements in cloud resource management have emphasized the need for intelligent and adaptive systems capable of making real-time decisions based on workload characteristics [210][211]. In this field, one promising direction is dynamically optimizing resource distribution by analyzing the size and nature of incoming request packets [212][213], approach leverages a centralized broker to monitor, analyze, and direct network traffic to the appropriate VMs based on the size of the request packets. This method not only enhances VM efficiency but also reduces latency and improves overall system performance. By incorporating a fuzzy logic system that uses imprecise inputs to make informed decisions, the broker can dynamically adjust VM allocation better to match the real-time demands of the cloud environment [214][76]. The Cloud Analyst tool provides a robust platform for implementing and simulating broker driven VM allocation strategies. It allows for detailed modeling and analysis of cloud computing environments, facilitating the evaluation of various allocation methods under different scenarios. The Cloud Analyst tool integrates fuzzy logic [215], [216], and [217]. As discussed in the previous contribution, propose a novel approach to (VM) allocation that optimizes resource utilization, reduces latency, and enhances overall system performance. This research aims to advance the field of cloud resource management by addressing the limitations inherent in traditional VM allocation strategies. By focusing on the dynamic optimization of VM allocation based on request packet size and workload classification, the proposed broker-driven approach seeks to provide high-quality cloud services while ensuring efficient resource use.

8.2 Current Issues and Challenges

Research on advanced VM allocation strategies aims to optimize resource utilization and performance in cloud computing, addressing the limitations of traditional strategies that often overlook the impact of varying request packet sizes. Sangaiah, Arun Kumar, et al. (2023) propose an intelligent dynamic resource allocation method that integrates TSK neural-fuzzy systems with ACO techniques to reduce energy consumption in cloud networks. This method, which uses real-time data, significantly enhances efficiency and performance in VM migration [218]. However, existing methods often fail to consider the varying sizes of request packets, which can significantly impact network performance. In contrast, broker-driven approaches enhance network performance by dynamically allocating (VMs) based on request packet sizes. This allows for real-time optimization of resource distribution and reduces latency, effectively addressing the limitations of traditional methods.[219] proposes a broker-based mechanism to connect CSPs with customers, analyzing task tendencies and assigning resources. This model uses multi-criteria decision-making to maximize profits, ensure customer satisfaction, and reduce energy consumption in cloud data centers. [220] highlights the increasing demand for cloud services, which necessitates a flexible and dynamic design for data center deployment. Traditional traffic engineering approaches are inadequate for efficiently utilizing IT and network resources. The study suggests two fuzzy logic controllers for efficient VM allocation. These controllers are based on the Mamdani and Sugeno inference processes. Preliminary simulation tests validate the effectiveness of the proposed approach. The Cloud Analyst tool simulates cloud computing environments, evaluates VM allocation strategies, and simulates broker-driven approaches. It is used in a study [221], which discusses the widespread adoption of cloud computing for web applications. The study uses virtualization concepts and resource allocation policies to manage resources in a cloud computing environment. They use a GUI tool called Cloud Analyst to simulate the cloud environment, focusing on energy consumption minimization and class diagram design. Furthermore, integrating advanced algorithms with broker-driven approaches has shown significant promise for optimizing VM allocation. [222] proposes DeepBS, a DRL-based scheduler, to address the inherent uncertainties in cloud broker VM scheduling due to on-demand IaaS VMs. Their study demonstrates that DeepBS improves cost optimization by learning from experience and enhancing scheduling strategies in unpredictable environments, showcasing its potential in dynamic cloud computing. Several recent studies have further expanded on these concepts. For instance, [223] emphasizes the significance of mobile terminal cloud computing migration technology in addressing evolving computer and cloud computing demands. They highlight the necessity for efficient data access, storage, and minimal time delays. They also introduce ML-based VM migration optimization and dynamic resource allocation as key research directions in cloud computing. Similarly, [224] introduces a resource allocation model called IMARM, which uses an intelligent multi-agent system and reinforcement learning. Combining multi-agent characteristics and Q-learning, IMARM dynamically allocates resources based on changing consumer demands and optimizes VM placement. Experimental results indicate that IMARM outperforms other algorithms in energy consumption, fault tolerance, load balancing, and execution time.[225] reviews resource allocation and service provisioning in multi-agent cloud robotics. They provide a taxonomy of resource allocation strategies, covering resource pooling, computation offloading, and task scheduling. The work discusses challenges such as heterogeneous energy consumption rates and data transmission delays and suggests future research directions to advance the field. The authors emphasize addressing research gaps and mitigating data transmission delays for efficient service provisioning. [226] notes that cloud computing has revolutionized resource management, but challenges remain due to scalability, heterogeneity, and dynamic environments. (AI) technology has emerged as a solution to improve efficiency. This work reviews AI techniques for resource management, including ML, reinforcement learning, predictive analytics, natural language processing, and genetic algorithms. It discusses AI-based strategies for efficient resource management, including automated resource provisioning, intelligent workload planning, predictive maintenance, and energy-efficient management. The work also discusses evaluation metrics, performance analysis techniques, ethical considerations, and future directions for AI integration. VM allocation research has also focused on energy efficiency. [227] explores energy-efficient resource allocation using a hybrid heuristic algorithm, showing substantial improvements in energy consumption. Finally,[228] reviews the state-of-the-art and research challenges in cloud computing, providing a comprehensive overview of current trends and future directions in VM allocation and resource management.

8.3 Broker-Driven Methodology in Cloud Computing

The proposed methodology for optimizing (VM) allocation in cloud computing environments leverages a broker-driven approach, enhanced with a fuzzy logic system, to dynamically optimize resource distribution based on the size of incoming request packets. This method is designed to improve VM efficiency, reduce latency, and enhance overall system performance. The following sections detail the key components of the methodology: broker design, fuzzy logic system, integration with the Cloud Analyst tool, and evaluation metrics. Table 8.1, shows the Workload Sizes alongside the specifications for the Google Cloud Platform's t2d-standard machine type, using data from the Google Cloud Compute Engine Pricing. The system leverages real-time data for smart VM allocation, demonstrating its adaptability by adjusting resource distribution in response to changes in network conditions and workload demands.

Table 8.1 workload size machine series specifications.

Workload Size	Machine type Series	VCPU	RAM (GB)	Storage (GB)	BW (GBPS)	Price per hour ($)
Small (<1 GB)	t2d-Standard-1	1	4	2	2	0.054427
Medium (1-10 GB)	t2d-Standard-2	2	8	10	4	0.108854
Large (10-100 GB)	t2d-Standard-4	4	16	16	8	0.217708
Very Large (>100 GB)	t2d-Standard-8	8	32	32	10	0.435416
Massive (Big Data Processing)	t2d-Standard-16	16	64	100	14	0.870832

8.3.1 Design and Architecture of the Broker System

Design and Architecture of the Broker System, Integrating Traffic Monitoring, Data Analysis, and Traffic Routing. The proposed methodology utilizes the Optimized Response Time SBP with a LB approach, facilitated by the Cloud Analyst simulator. The broker acts as a mediator that monitors and analyzes incoming request packets. Its primary functions include:

- Traffic Monitoring: continuously monitoring network traffic to collect data on packet sizes and associated metrics.
- Data Analysis: analyzing the collected data in real-time to identify patterns and trends in request packet sizes.
- Traffic Routing: directing traffic to the appropriate VMs based on the analysis, ensuring optimal resource allocation [229][230].

The broker features advanced data analytics to manage the varied and dynamic cloud workloads effectively.

8.3.2 Implementation of Fuzzy Logic

The Fuzzy Logic system is integrated into the broker to handle the uncertainty and variability inherent in cloud environments [76][231]. The model's input parameters were crafted using the Fuzzy Logic Designer, adhering to the methodological framework introduced in Chapter 4. However, for this chapter, adjustments were made to the division of the universe of discourse to align with the specific primitives and structural prerequisites of the developed model. This chapter focuses on utilizing two primary inputs and single outputs, categorized as VM categories. Five defined triangular membership functions characterize each input.

First input (Workload- Request Packet Size)
Represented by the size of incoming request packets.

Small: [0 0.9 5]; Medium: [1 10 50]; Large: [10 100 150]; V.Large: [100 150 200]; Massive: [150 200 250]

i. Second input (CPU Utilization)

Current utilization levels of the available VMs.

Poor: [10 30 40]; Fair: [30 50 60]; High: [50 70 80]; V.High: [70 85 90]; Excellent: [85 100 100]

ii. Output (T2D standard machine types-Levels)

Simple: [0 0.1 0.2]; Moderate: [0.2 0.3 0.4]; Good: [0.4 0.5 0.6]; V.Good: [0.6 0.7 0.8]
High-Performance: [0.8 1 1]

These functions allow the system to evaluate the inputs and produce a set of fuzzy rules, illustrated in Appendix 8 (Figure 1), that determine the optimal VM allocation strategy. The outputs of the Fuzzy Logic system include VM classes, which categorize VMs based on their suitability for handling the current workload and CPU utilization levels [232]. Table 8.2. Illustrated the fuzzy logic output – Decision making.

Table 8.2 Rules – Decision making.

CPU Utilization	Poor	Fair	High	V.High	Excellent
Request Packet Size	Output (T2D standard machine types-Levels)				
Small	Simple	Simple	Simple	Moderate	Moderate

Moderate	Moderate	Simple	Moderate	Moderate	Good
Large	Moderate	Moderate	Good	Good	V. Good
V.Large	Good	Good	V. Good	V. Good	H.Perf.
Massive	V.Good	V.Good	H.Perf.	H.Perf.	H.Perf.

8.3.3 Integration with Cloud Analyst Tool

The Cloud Analyst tool is employed to simulate and evaluate the proposed broker-driven approach. This tool provides a robust platform for modelling cloud computing environments and testing various VM allocation strategies [233]. The integration process involves:

8.3.3.1 Cloud Environment Modeling

Configuring a simulated cloud environment in Cloud Analyst involves setting up data centers with single VMs and associated user bases. This setup is tested across five scenarios, each employing the proposed broker technique to assess performance and efficiency. The process is illustrated in Appendix 8 (Tables 1 and 2).

8.3.3.2 Throttling Algorithm

In cloud computing, throttling plays a pivotal role in managing system loads and sustaining service quality while also keeping operational costs in check. This process is vital for scaling computing resources efficiently. Through the application of diverse algorithms, throttling ensures that cloud services remain scalable, dependable, and fair. Specifically, it regulates the allocation of critical computing resources such as CPU, BW, and memory. This control helps prevent any single user or application from monopolizing resources, thereby avoiding system overloads and ensuring equitable performance across all users [234].

8.3.3.3 Broker Policy for Response Time

In cloud environments typically involves strategically managing resource allocation to minimize latency. This policy ensures that the broker prioritizes tasks or requests that are critical for performance, dynamically adjusting resource distribution based on real-time demands. Doing so effectively reduces waiting times for resource-intensive operations, ensuring that all processes are executed as swiftly as possible, thus enhancing overall system efficiency and user satisfaction [200].

- Implementing Broker Logic: embedding the broker’s traffic monitoring, analysis, and direction functionalities into the Cloud Analyst simulation.
- Incorporating Fuzzy Logic: integrating the Fuzzy Logic system with the broker within Cloud Analyst to dynamically adjust VM allocation based on real-time data.

8.4 Simulation and Evaluation of Results and Discussion

The proposed methodology was rigorously evaluated through extensive simulations conducted using the Cloud Analyst tool [235]. In the proposed methodology, five distinct scenarios were executed. Each scenario involved deploying ten distinct user bases, consistent with the configuration described previously in this study. In the initial scenario, the user's request was within this amount. (500,000,000) Bytes were processed using t2d-Standard-1. Moving to the second scenario, requests within this amount of a workload of 1,000,000,000 bytes were allocated to t2d-Standard-2. The third scenario handled requests within the workload of 10,000,000,000 bytes assigned to t2d-Standard-4. Subsequently, requests amounting to 150,000,000,000 bytes in the fourth scenario were managed using t2d-Standard-8. Finally, in the fifth scenario, where requests amounted to 200,000,000,000 bytes, t2d-Standard-16 was allocated for execution. Similar parameters were utilized when implementing the traditional method scenarios, as in the proposed method concerning user base logins to the computing environment, defined by Peak hours Start-End and Avg. Peak Users On-Off. However, the traditional approach diverges from the proposed method in how it distributes and processes user requests and workloads, as detailed in Table 8.3.

Table 8.3 Basics of applying the traditional method.

Scenario number	User Bases	Request Packet Size (Byte)	Machine type Series	Price per hour($)	Load balance Algorithm	Broker policy
1	[UB1 UB10]	[500,000,000 200,000,000,000]	t2d-Standard-1	0.054427	Throttling algorithm.	Optimize response time.
2	[UB1 UB10]	[500,000,000 200,000,000,000]	t2d-Standard-1	0.108854	Throttling algorithm.	Optimize response time.
3	[UB1 UB10]	[500,000,000 200,000,000,000]	t2d-Standard-4	0.217708	Throttling algorithm.	Optimize response time.
4	[UB1 UB10]	[500,000,000 200,000,000,000]	t2d-Standard-8	0.435416	Throttling algorithm.	Optimize response time.
5	[UB1 UB10]	[500,000,000 200,000,000,000]	t2d-Standard-16	0.870832	Throttling algorithm.	Optimize response time.

A variety of workload scenarios were implemented, each featuring distinct request packet sizes and VM resource demands. These simulations were designed to assess the robustness, adaptability, and practical viability of the broker-driven approach, particularly in comparison to traditional VM allocation strategies. The experimental setup modeled a realistic cloud environment where the dynamic nature of cloud workloads was replicated to test how effectively the system responds under varying operating conditions. The broker-driven system incorporates a fuzzy logic mechanism that utilizes workload packet size and CPU utilization as key input parameters to dynamically allocate (VMs) based on their classification across five levels of workload intensity. Appendix 8 (Figure 2) visually demonstrates the simulation execution process, while Appendix 8 (Figure 3) illustrates the decision outcomes produced by the fuzzy logic system. Quantitative performance metrics were collected, including overall response time, data center processing time, request serving time, total VM costs, and total data transfer costs. The comparison between the traditional VM allocation approach (summarized in Table 8.4) and the proposed method (detailed in Table 8.5) clearly demonstrates significant improvements across all critical metrics. Specifically, the proposed broker-driven system reduced response time by up to 68%, decreased processing and serving times by an average of 20% and achieved substantial reductions in cost—most notably in data transfer and VM provisioning. The novelty of this research lies in the introduction of a broker-driven VM allocation model that uniquely integrates fuzzy logic with packet size classification—an aspect widely neglected in conventional allocation approaches. Traditional methods largely emphasize Resource scalability capabilities, yet they often fail to account for the heterogeneity and variability of incoming packet sizes, which are essential determinants of workload behavior. By incorporating packet size as a classification factor alongside real-time CPU utilization, the proposed approach ensures a more granular and intelligent allocation of cloud resources. Moreover, the integration of fuzzy logic contributes significant adaptability to the decision-making process. The fuzzy inference engine enables the system to handle uncertainty and imprecision, aligning resource allocation with dynamic demand patterns more effectively than static rule-based methods. This enables the system not only to allocate resources optimally but also to proactively prevent bottlenecks and reduce energy consumption through more efficient VM utilization. The methodological innovation also includes a well-defined classification scheme that translates request sizes and CPU usage into actionable VM categories. This classification is mapped through triangular membership functions that support interpretability and computational efficiency—key features for scalable cloud infrastructure. The proposed approach has substantial practical implications. By dynamically aligning VM allocations with workload characteristics, cloud providers can achieve better energy efficiency, improve system responsiveness, and reduce operational costs. The ability to manage workloads based on packet size and CPU load allows for a more equitable and efficient distribution of cloud resources, enhancing the performance and reliability of services across heterogeneous and high-demand environments. This study contributes to the advancement of intelligent cloud resource management by offering a scalable, cost-effective, and energy-aware alternative to traditional VM allocation. The results validate the theoretical principles underpinning this model and position it as a promising solution for next-generation cloud systems where adaptability and performance optimization are paramount.

Table 8.4 Summary of the results of the traditional method.

Scenario	Overall response time Avg(ms)	Datacenter processing time Avg(ms)	Datacenter request serving times Avg(ms)	Total data transfer cost ($)
1	571309,86	58,06	58,06	33959999,08
2	548272,30	59,31	59,31	30557098,39
3	565510,88	60,39	60,386	33791313,17
4	558790,62	58,03	58,026	33726768,49
5	574401,10	59,35	59,348	32435417,18

Table 8.5 Summary of the results of the proposed Method.

Scenario Number	Overall response time Avg(ms)	Datacenter processing time Avg(ms)	Datacenter request serving times Avg(ms)	Total data transfer cost ($)
1	333748,21	56,41	56,141	4186420,44
2	278151,12	49,88	49,875	6354904,17
3	183111	44,30	44,297	9916305,54
4	0	39,32	39,323	4909515,38
5	0	40,26	40,264	4531860,35

8.5 Summary

This chapter introduced and validated a broker-driven approach enhanced by fuzzy logic for intelligent (VM) allocation in cloud computing. While the simulation results confirmed significant improvements in response times, resource utilization, and cost efficiency compared to traditional methods, the broader implications of this work extend beyond the numerical gains. The proposed methodology demonstrates how integrating fuzzy logic with packet size classification enables cloud systems to respond dynamically to heterogeneous workloads, addressing a critical gap in conventional resource allocation strategies. This adaptability is essential for future cloud environments that must support diverse and evolving service demands while maintaining energy efficiency and cost-effectiveness. Looking ahead, this research opens avenues for developing cloud brokerage systems capable of real-time service selection based on user requirements, network dynamics, and workload characteristics. Future contributions will focus on extending this approach to scenarios involving user mobility, service migration, and the complex interplay between performance and cost in dynamic cloud ecosystems. These developments aim to transform cloud resource management into a more intelligent, context-aware process, aligning technical innovation with practical deployment needs.

Chapter 9 Reliable and Cost-Effective Fuzzy-based Cloud Broker

Due to the rapid increase in CSPs, users find it challenging to select a cloud service that suits their needs and budget. Thus, having an intermediate entity between the two in cloud broking services is more crucial than ever. Chapter 9 Proposes a cloud broker that uses fuzzy logic to rank service instances and users, aiming to balance user needs and service provider interests. It investigates the impact of user mobility on service quality by analyzing scenarios involving stationary and mobile users. The study also examines the impact of service migration on performance and cost, highlighting the benefits of dynamic resource management. The proposed broker ensures reliable service delivery with stable performance and cost-efficient resource usage, outperforming traditional methods in mobility and service migration scenarios.

9.1 Cloud Brokerage Systems and Cost Optimization Using Fuzzy Logic

Remote processing has become increasingly popular in recent years with the rise of cloud computing [236], MEC [237], and fog computing platforms [238]. These paradigms are considered the main enablers for Ultra-Reliable Low Latency Communications (URLLC), Enhanced Mobile Broadband (eMBB), and Massive Machine-Type Communications (mMTC) services [239] that are promised for beyond 5G networks. These kinds of services are more strict in Key Performance Indicators (KPIs), which can only be achieved by overcoming the limitations of users' equipment resources and exploiting the unlimited cloud resources via remote processing. Notwithstanding the indisputable advantages of these platforms, they also pose novel challenges for (CSPs) and their customers. For example, the user who needs a certain service will have difficulty choosing from the abundance of alternatives offered by the (CSPs). On the other hand, CSP may also have difficulty promoting their services and efficiently allocating their resources to accommodate more users. Therefore, mentioned in the previous chapters, focusing on representing a third party is usually recommended in the form of a cloud broker, which is an entity that acts as middleware between potential customers and CSP. The presence of such an entity can help not only offer efficient and affordable services for users but also help with resource management and LB cross-cloud or between different instances of the service in the same cloud. Driven by the importance of having a broking service that takes into account the customers' needs and the CSP's interests, present this study with several contributions in mind.

9.2 Review of Existing Cloud Brokers and Analysis of Intelligent Cloud Brokerage

Cloud brokerage services have been widely discussed in academia, where numerous studies have been conducted in search of the optimal broker. Focus-wise, some studies were customer-centric, where the interest of the clients was considered the priority in terms of focusing on improving the Quality of Service (QoS) provided for the users. Examples of these studies are [240–244]. Other approaches were more focused on the broker profit [245–247]. This profit can mainly be acquired by wisely managing the cloud's resources or by exploiting the difference in prices between on-demand and reserved service instances [247]. Some studies, however, tried to find a balance between the broker's and user's interests [248, 249]. The brokerage problem is viewed in some research studies as a resource provisioning and management problem, which can be summed up as deciding which resources should be set aside for the user and then distributing the load among the resources that the service provider has available [250]. Thus, numerous studies focused on LB and efficient resource allocation such as [251–254], Methodology-wise, many techniques were employed for the brokerage service, such as game theory [255], reinforcement learning [256, 222], weighted algorithm [257, 258], ontology [259], Analytic Hierarchy Process (AHP) in combination with Technique for Order Preference by Similarity to Ideal Solution (TOPSIS) [260] and fuzzy logic [261–263]. The main issue in game theory approaches is that the negotiating process becomes lengthy when the number of SLA parameters rises [264]. Similarly, the primary disadvantage of reinforcement learning approaches is their lengthy execution time to reach a stable model, which leads to a long learning phase in which the broker is not functioning. On the other hand, weighted algorithms need predefined weights and criteria to select the service efficiently. Setting a fixed value for these weights for all users may be unsatisfactory for some users. Meanwhile, defining values that correspond to each user takes a lot of effort and time. In AHP combined with TOPSIS approaches, the broker employs a multi-criteria decision-making technique to choose a suitable cloud provider after evaluating each provider's quality and ranking each one according to the customer's needs. Therefore, these approaches can be confusing for nonprofessional users since they are forced to specifically define their priorities and preferences. [250, 264]. Employing fuzzy logic systems can yield good results. However, two problems will surface when many input parameters are taken into account. The first issue is when the number of customers grows and online service selection is required, collecting this data can become more challenging if not impossible. Additionally, some service providers might be reluctant to divulge some parameters since doing so could reveal security flaws and compromise the service provider's integrity. The second problem is that as the number of rules increases dramatically with the increase of input parameters, setting up the inference engine will become more difficult and time-consuming. These problems can be identified in studies such as the fuzzy-based brokers proposed in [261–263]. In our approach, combine two different techniques for our cloud brokerage system. They are fuzzy logic and a modified version of TOPSIS. In the study, various data centers from AWS, Google Cloud (GC), and Azure Cloud Services (AZURE) are distributed across different geographical regions. These (CSPs) offer a range of VM types, including general-purpose, compute-optimized, memory-optimized, and accelerator-optimized instances. Our approach uses fuzzy logic to classify and rank the service instance and the user, trying to satisfy users' and service providers' interests and needs. Moreover, we only consider two easily acquired parameters for each fuzzy system, reducing the rules required in the engine and making the broker incorporation in the cloud environment more feasible. We associate the user with an appropriate service instance based on this ranking. Further details on our proposed brokerage system design are elaborated in the subsequent section.

9.3 System Design

The proposed system considers the user requirements as well as the service specifications offered by different cloud providers. The proposed system architecture is illustrated in Figure 9.1, made an effort to build the system so that both novice and expert users could utilize the broker with ease since the user interface is thought to be one of the most common problems with commercial brokers [265].

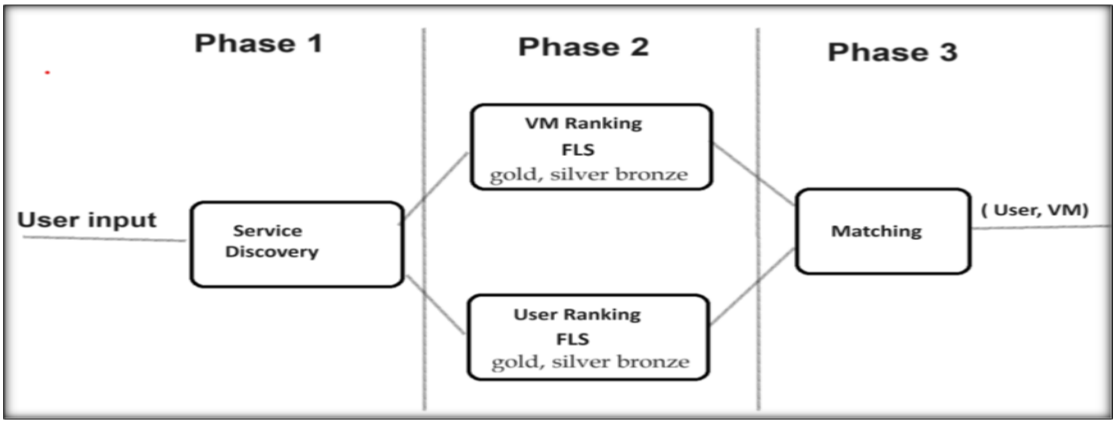


Figure 9.1 Proposed System Architecture.

i. Clarification and Detailed Explanation of the Matching Process:
In the proposed fuzzy-based cloud brokerage system, the "Matching" phase constitutes a critical step in the overall service allocation process. The matching procedure occurs after two crucial prior stages, which are clearly described:

1. Service Discovery: Users specify their service requirements (type, budget, desired quality), and the broker identifies relevant cloud service instances from available (CSPs).
2. Ranking (Classification):
   - A fuzzy logic system is employed to independently classify (VMs) and users into distinct ranks: Gold, Silver, and Bronze.
   - VM ranking considers CPU availability and cost; user ranking considers task size and budget constraints.

Once these classifications are established, the "Matching" process explicitly associates users with suitable VM service instances according to their respective ranks (Gold, Silver, Bronze). This step ensures alignment between user expectations and VM capabilities.

ii. Detailed Explanation and Steps of the Matching Phase:
The matching operation specifically follows these structured steps:
Step 1: Rank-Based Matching: The system pairs users and VM instances according to their corresponding ranks:

   - Gold-ranked users are matched to Gold-ranked VM instances to ensure high-quality service and resource availability.
   - Silver-ranked users are matched to Silver-ranked VM instances, providing a balanced trade-off between performance and affordability.
   - Bronze-ranked users are matched to Bronze-ranked VM instances, satisfying basic service requirements economically.

Step 2: Final Allocation: Once the matching pairs are established, the broker executes resource allocation, ensuring optimal performance, service quality, and cost-effectiveness for users and efficient resource utilization for providers.

iii. Reasoning for the Matching Process: The rank-based matching approach achieves several key objectives:

- Optimal Compatibility: It ensures users receive appropriate resource types matching their service quality and budget constraints.
- Balanced Load Distribution: Aligning user demands and VM capabilities helps maintain balanced resource utilization.
- Enhanced User Satisfaction: The systematic matching ensures user needs are accurately met, enhancing overall satisfaction.
- Efficiency in Decision Making: Utilizing predefined rankings simplifies the decision-making process, enabling efficient real-time service allocation.

9.3.1 The broker's Fuzzy-logic systems

In the proposed cloud broker, we used two fuzzy logic systems. One is designated to rank the service, and the other is to rank the users. These two systems are detailed in the following subsections.

9.3.1.1 VM ranking Fuzzy logic system

The Fuzzy Logic System (FLS) system used for VM ranking is illustrated in phase 2 in Figure 9.1. The input parameters for this system are the percentage of available Central Processing Unit (CPU) on the VM, and the cost of the VM. These parameters go into the fuzzification phase to be mapped into the linguistic values (low, medium, and high) according to the membership functions illustrated in Figure 9.2 and Figure 9.3, used trapezoidal and triangular fuzzy membership functions to map the crisp input variables into multivalued logic. After the fuzzification phase, these resulting linguistic values will go through the inference engine. To assess the fuzzy output variable indicating the VM ranking, the engine uses simple IF-THEN rules with a condition and conclusion. For instance:
IF VM's available CPU capacity is (Low)AND the VM cost per month is (Low) Then the VM has a (Silver) ranking.
The VM will be classified as Gold, Silver, or Bronze according to its specification, Figure 9.4, illustrate the VM's ranking membership function. This rank is subjective and a typical user's assessment served as the basis for this classification. The set of fuzzy rules used in the inference engine is depicted in Table 9.1. The resulting ranking is then converted to a crisp value using the COG technique.

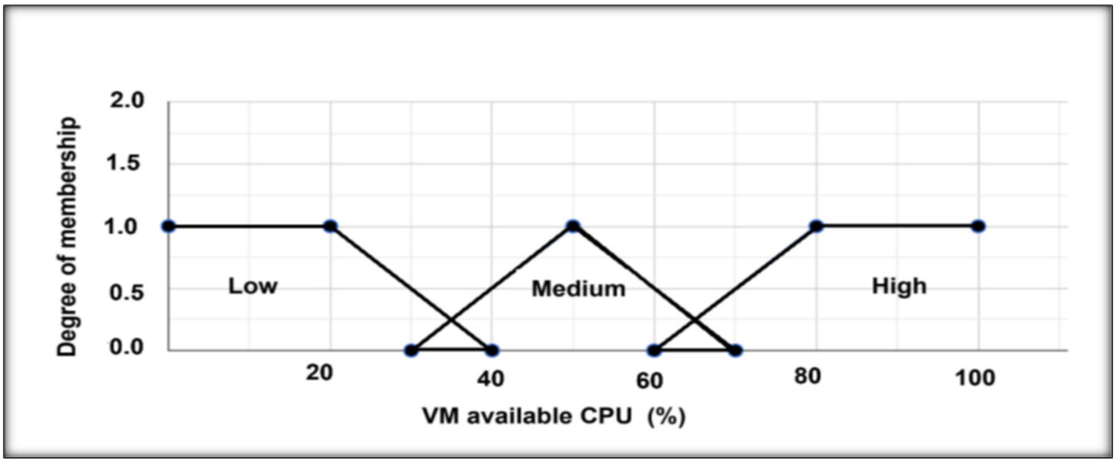


Figure 9.2 The VM's availability membership function.

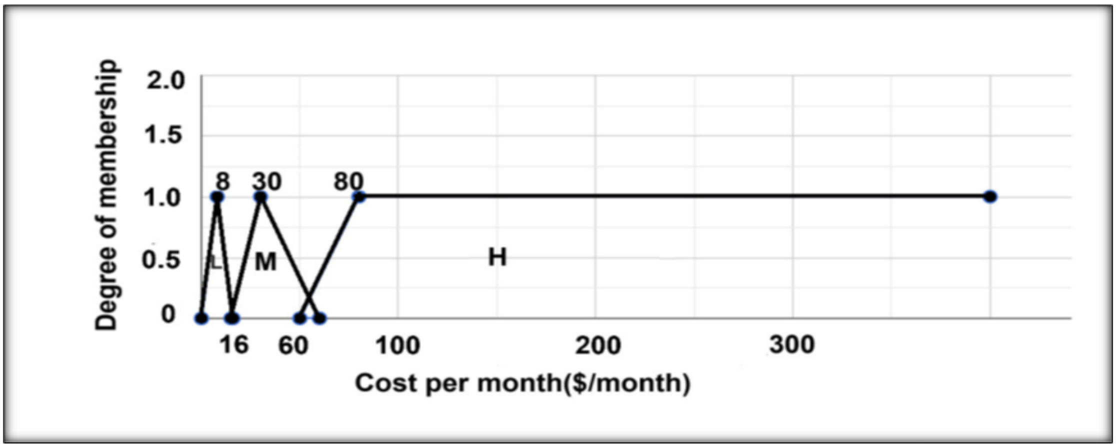


Figure 9.3 The VM's Cost membership function.

Table 9.1 VM ranking FLS.

Available CPU / Cost per month	Low	Medium	High
	Service classification		
Low	Silver	Gold	Gold
Medium	Bronze	Gold	Gold
High	Bronze	Silver	Silver

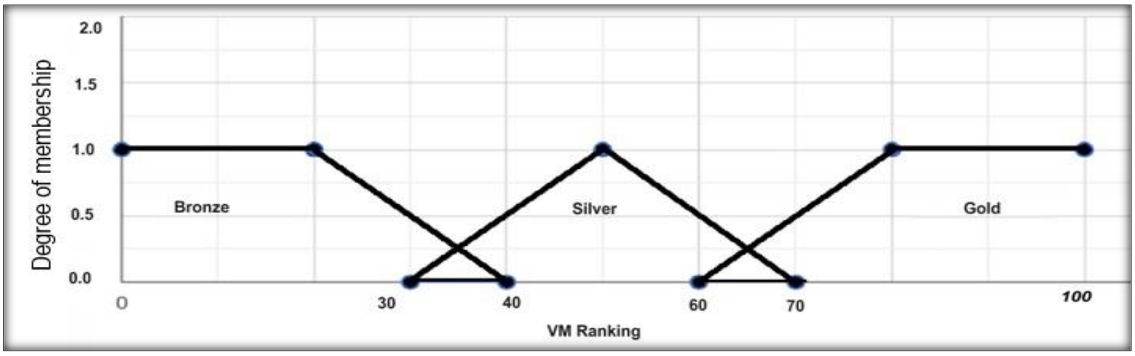


Figure 9.4 VM's ranking membership function.

9.3.1.2 User ranking Fuzzy logic system

These parameters include the client's budget and the task length, measured in the number of instructions required. These fuzzy logic inputs are translated into Low, Medium, and High linguistic values. Triangular and trapezoidal membership functions were employed to convert the user budget and task length into fuzzy sets, depicted in Figure 9.5 and Figure 9.6, respectively. Based on their requirements and financial constraints, the user type will be classified as Gold, Silver, or Bronze.This rating is based on our estimation of what the service provider would assign to that user. To compute the user ranking, which is the output parameter, an IF-Then inference engine is used, with a set of rules summarized in Table 9.2. In the defuzzification stage, the linguistic value representing the user's rank and derived from the inference engine is then mapped into a crisp value using the COG method for defuzzification. The membership function used for the user rank is depicted in figure 9.7.

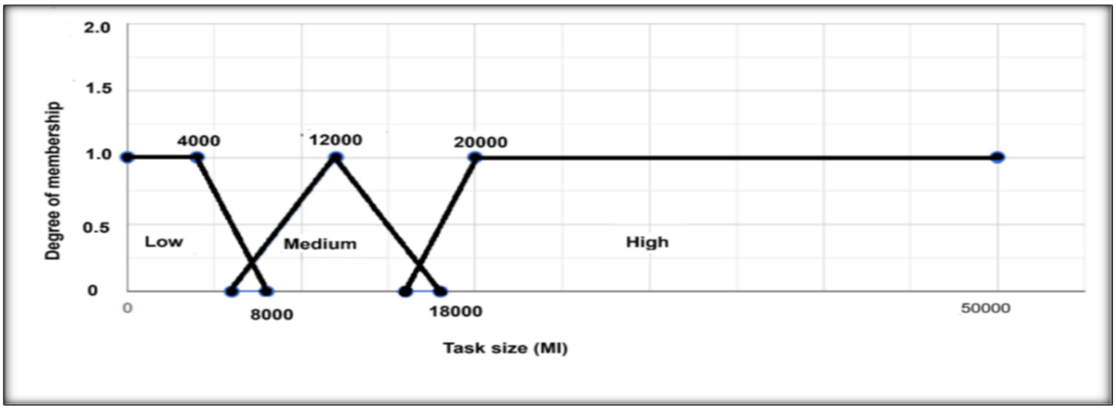


Figure 9.5 Task size membership function.

Table 9.2 User ranking FLS.

Task size / Cost per month	Low	Medium	High
	Service classification		
Low	Silver	Bronze	Bronze
Medium	Gold	Silver	Bronze
High	Gold	Gold	Gold

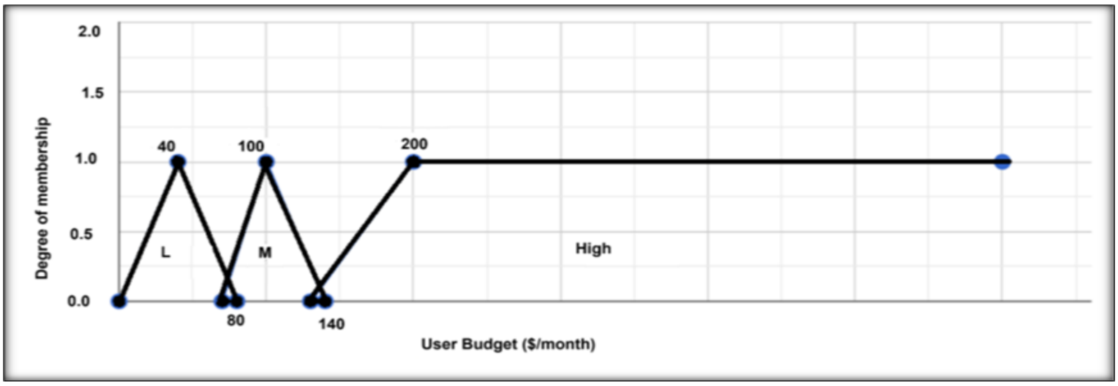


Figure 9.6 User budget membership function.

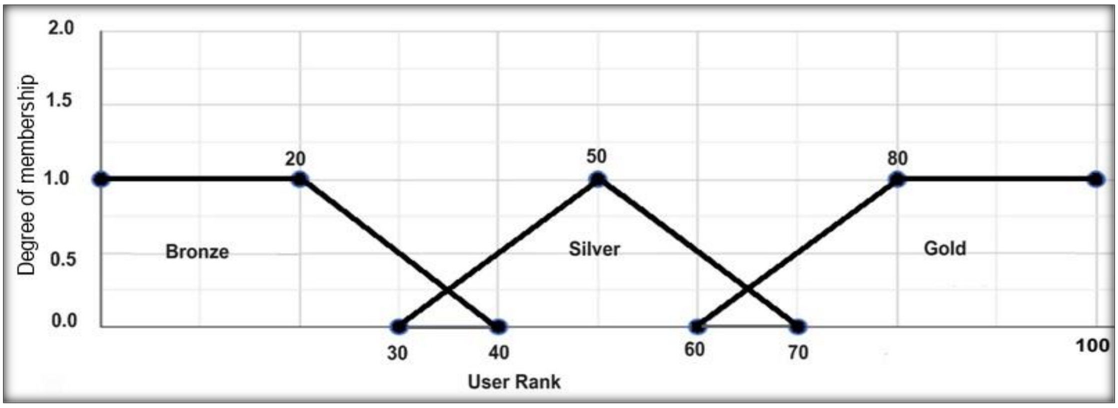


Figure 9.7 User rank membership function.

9.4 Scenario Description

Used Edge CloudSim [266–269]. Simulator to implement the proposed cloud broker on (MEC) paradigm, made this choice as the services running on the virtualized edge are more sensitive to delay and the broker selection of the appropriate service instance will have a more significant impact in this kind of setting. In the scenario, have different data centers belonging to AWS, Google Cloud (GC), and Azure Cloud Services (AZURE) and placed in different regions, namely: United State of America (USA), western Europe and Southeast Asia and the data centers located in different regions are connected via (WAN) and the datacenters located in the same region are connected by MAN network. Giant CSP have different types of VM, such as general purpose, compute-optimized, memory-optimized, and accelerator-optimized instances. Thus, tried to make the scenario more realistic by choosing one or more instances from these different types. The chosen instances are detailed in Table 9.3. All the values in this table are taken from the official websites of the three cloud providers. Four types of delay-intolerant services are used in the simulation setup, with them specifications in terms of the generated traffic characteristics mentioned in Table 9.4. The delay sensitivity is a value between 0 to 1 where the value 1 indicates the application with the highest delay sensitivity. Each user requests a specific type of service identifying his budget and his needs will be determined by his traffic profile and more specifically his average tasks' length measured in millions of instructions (MI). This value is usually estimated based on the application he requested. Based on these parameters, the cloud broker will identify the most appropriate service instance in the region where the user is currently located. The user communicates with the datacenter where the service is placed via a wireless local area network. This network is modeled as M/M/1 Queue. EdgeCloudSim includes realistic network measurements. For modeling WLAN delay, it examines the performance of an 802.11-family access point, while WAN delays are calculated using measurements from a fiber internet connection in Istanbul. The results of the empirical network delay analysis are detailed in [266].

Table 9.3 Official Application Specifications from the Three Cloud Providers' Websites.

Name	CSP	Type	Number of vcpu	Memory
T2A	GC	General purpose	2	4

E2	GC	Cost optimized	2	1
M1	GC	Memory optimized	40	961
C2	GC	Compute optimized	4	6
A2	GC	Accelerator optimized	12	85
t2. small	AWS	General purpose	1	2
i4i.large	AWS	Storage optimized	2	16
r7a.medium	AWS	Memory optimized	1	8
r7a.large	AWS	Memory optimized	2	16
c7a.medium	AWS	Compute optimized	1	2
c7a.large	AWS	Compute optimized	2	4
p3.2xlarge	AWS	Accelerator optimized	8	61
hpc7g.4xlarge	AWS	HPC optimized	16	128
B2ls v2	AZURE	General purpose	2	4
F2s v2	AZURE	Compute optimized	2	4
E2as v5	AZURE	Memory optimized	2	16
L8as v3	AZURE	Storage optimized	8	64
NC6	AZURE	GPU optimized	6	56
H8	AZURE	High performance compute	8	56

Table 9.4 Types and Specifications of Delay-Intolerant Services in the Simulation Setup.

Type	Average of upload data	Average of download data	Task Length	Delay sensitivity
Health App	1500	25	9000	0.7
Augmented Reality	20	1250	3000	0.9
Heavy Computing	2500	200	45000	0.1
Infotainment	25	1000	15000	0.3

9.5 Results analysis

Compare the proposed system with two different approaches. They are, a random approach where the user randomly chooses the service instance, and the second approach is when the broker chooses the service instance with the highest capability in terms of processing power available to associate the user with, compare these approaches focusing on two main metrics which are the service delay experienced by the users and the cost the user needs to pay per month, make this comparison in four distinct scenarios. They are:

- First scenario: the users are motionless. Upon selecting a service instance from a certain CSP, the user establishes and maintains the association until the simulation time expires. This represents the policy of reserved VM.
- Second scenario: the users are mobile and move around following a nomadic mobility, spending a specific duration on one site before moving on to the next. In this scenario, the service instance stays in the original data center with which it was associated and is not migrated. The payment policy here is also a reserved instance policy.
- The third scenario involves clients moving around following a nomadic mobility model. In this scenario, test a cross-cloud migration, where the broker seamlessly migrates the service across multiple cloud providers ensuring the satisfaction of SLA requirements defined by the user. The payment policy in this scenario is pay-as-you-go policy (PAYG). Where the user rents resources on-demand and only pays for his usage.

For the first scenario, compare the proposed approach with two approaches. They are the Least Loaded (LL), in which the VM that is least loaded and within the budget of the user is chosen as a service instance. The second algorithm is a random selection, where the service instance is chosen randomly. The simulation is performed for five runs and the average results for service delay and the client's budget savings are illustrated in figures Figure 9.8, and Figure 9.9. As shown in these figures, by employing our fuzzy logic approach, were able to achieve better results regarding the average service delay. The increase in the delay in accordance to the increase of the number of clients is normal due to the limited number of service instances in the scenario.

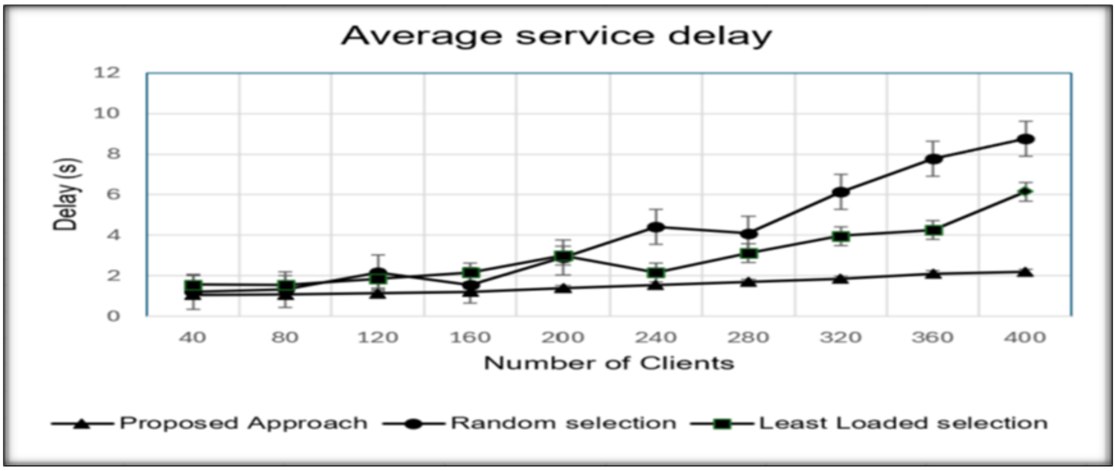


Figure 9.8 Average service delay for immobile users.

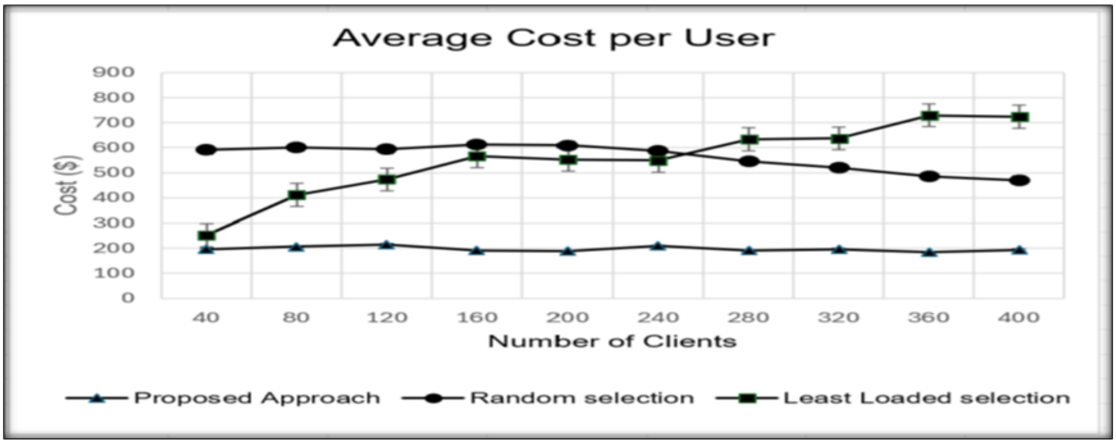


Figure 9.9 The average of monthly client payment.

However, noted that our approach exhibits a more stable performance than both random and least-loaded approaches, where the variation in the delay is unnoticeable compared to the other two. This is a very important aspect from the service provider's perspective as he is obligated to respect certain QoS limits defined in the SLA. Thus, employing our approach can guarantee more stable performance and prevent the violation of the SLA terms. The main reason why the LL approach failed to perform well is because service migration and dynamic task offloading are not supported in this scenario. Since each user is maintaining the association with the same service instance for the whole time, the effectiveness of choosing the least loaded instance is diminished. When comparing the proposed approach with the other two approaches regarding the average cost each customer has to pay, noticed LL and random approaches forced the clients to pay more as the number of clients increased. This is basically due to their imbalanced policies where the cost was not considered, and more users were associated with more expensive service instances. On the other hand, our approach surpassed both approaches and the customer were still able to get the service with the same quality while maintaining the same payment.

9.5.1 The effects of Client's mobility

In the second scenario, we tested the three approaches on mobile clients. The clients follow a nomadic mobility model, mimicking a normal person's daily routine, where he goes to certain points of interest such as the workplace, university, or home, spends some time there, and then moves to other places. In this scenario, once the user is associated with a service instance, he maintains his association regardless of his current location. This scenario reflects the operational policy of certain cloud brokers that do not support service migration, meaning that once a user is associated with a particular service instance in a specific data center, the connection remains fixed regardless of the user's subsequent movements. As a result, the service continues to be delivered from the original data center even if the user relocates to another geographical region, potentially increasing communication delays and impacting overall service quality. The results are illustrated in Fig. 10. All three approaches were significantly affected by the client's mobility as shown in Fig. 10. This is mainly because the communication delay started to play a significant part in the overall delay as none of the three approaches was able to mitigate the impact of the user's getting further away from the service instance. Our approach was not able to get notably better results in terms of the average service delay. However, it was able to maintain a certain stability in the performance, with less delay variation than both random and LL approaches. This is quite important for preventing SLA breaches.

Figure 9.10 Average service delay for mobile users.

9.5.2 Effects of Service Migration on SLA Compliance

In the third scenario, examined the implementation of the three brokerage approaches on mobile users with the support of service migration. As the service instance associated with the user is changing in accordance with the user's location, considered a pay-as-you-go pricing policy in each location, where the minimum reservation time is one hour. The resulting average service delay experienced by the clients as well as the average cost per user are illustrated in Figure 9.11 and Figure 9.12. Our approach and LL selection-based broker gave a very close performance in terms of service delay experienced by clients. The main advantage of our approach was in having the clients maintain the same quality of service while paying the same amount regardless of the number of users demanding the same service.

9.6 Real-World Implementation and Practical Implications

Estimate that our model can be integrated into the cloud computing environment easily. Using fuzzy logic for ranking can facilitate the use of this broker for unprofessional users. Nevertheless, several issues can arise. First, observed a significant amount of computation when the number of users increased. This resulted in a longer simulation time than other approaches such as the random and the LL service selection. When used in practice, this may have an impact on scalability. However, when sufficient resources are allotted for the broker to carry out fuzzy-logic-based ranking, significant computation time can be avoided.

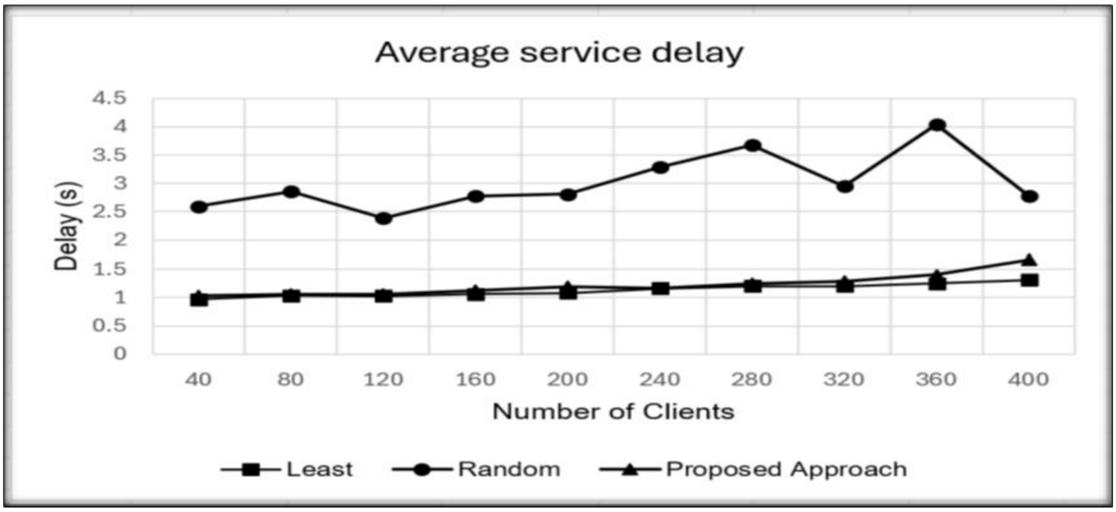


Figure 9.11 Average service delay with mobile users and service migration.

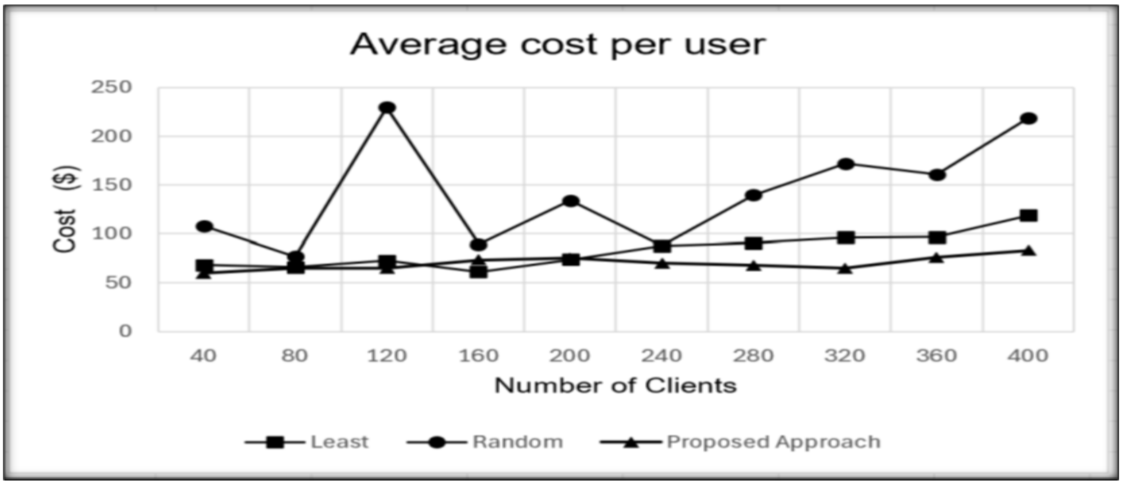


Figure 9.12 Average monthly payment in case of service migration.

To mitigate the computational requirements of the proposed system, we have developed several strategies aimed at improving efficiency. Users can be clustered and ranked as a single cluster to assist cut down on the amount of processing required for ranking. One of our model's primary input parameters for ranking a user is the average task size of the application he utilizes. When multiple people use the same application, both group-based and flow-based ranking are possible. For example, a group of video gamers at the same location or a group of employees in a firm using the same application can be ranked as a cluster using the aggregated flow specifications. Subsequently, a single service instance can be assigned to this group instead of allocating an instance for each user. Computation can also be minimized by employing user profiling and assigning a fixed rank for some clients based on the sensitivity of their services. For instance, users of health applications can be assigned the highest rank (Gold) due to the sensitivity and importance of the data transmitted.

9.7 Summary

In this contribution, introduce a novel fuzzy logic-based broker that considers both the interests of the client and the service provider, analyze various scenarios, demonstrating the feasibility of our approach. For future work, aim to enhance the design of the proposed broker by incorporating additional parameters into the decision-making process, such as the delay sensitivity of applications and the client's mobility profile. Our observations revealed that network delay plays a significant role, especially in the absence of service migration support for mobile users. To address this, plan to implement a new mechanism within the broker to mitigate the impact of mobility on service quality. As discussed in previous chapters, utilizing a third-party intermediary, typically in the form of a cloud broker, is widely recommended. A cloud broker acts as middleware between potential customers and (CSPs). The inclusion of such an entity facilitates the provision of efficient and cost-effective services for users while also assisting with resource management and LB across multiple clouds or between instances within the same cloud. Cloud broking is a rapidly growing field driven by the increasing adoption of cloud computing. The cloud services broking (CSB) market is expected to continue its expansion in the coming years. CSBs are instrumental in managing multi-cloud and hybrid cloud environments, optimizing cloud expenditures, and integrating advanced technologies such as (AI), big data, and the Internet of Things (IoT). Future advancements in cloud broking are expected to focus on deeper AI integration, enhanced security measures, expansion into emerging markets, and greater automation. This positions cloud broking as a dynamic and promising area of growth and innovation in the future.

Chapter 10 Theses

Cloud computing has become a cornerstone of contemporary IT infrastructure, delivering scalable and flexible access to computing resources through (SLAs) that define performance guarantees. Despite its advantages, several challenges persist. Compliance issues, vendor lock-in, and variability in Quality of Service (QoS) hinder efficient decision-making and operational management. Moreover, the rapid expansion of cloud data centers has escalated concerns about energy consumption, emphasizing the need for sustainable and energy-efficient management strategies. Geographical distances between data centers contribute to fluctuations in RTT and service reliability, compounded by the fact that (CSPs) often provide predominantly qualitative rather than quantitative network performance data. Efficient management of cloud-to-user latency and network optimization is, therefore, critical for ensuring global service reliability. Furthermore, distributed transaction management continues to face the ongoing challenge of maintaining both reliability and consistency in the face of hardware failures, network disruptions, and variable latency. To address these challenges, intelligent and adaptive cloud service management techniques are essential. Advanced resource allocation, SLA optimization, and predictive modeling play crucial roles in enhancing performance, reducing latency, and ensuring scalable, cost-effective, and sustainable cloud services aligned with evolving IT demands. In this context, my doctoral research has contributed three significant systems and methodologies that advance the field of cloud computing through innovative applications of fuzzy logic and decision-making models:

Thesis I: Intelligent SLA Guarantee Model for Cloud Computing

I have developed an **Intelligent SLA Guarantee Model for Cloud Computing**, employing fuzzy logic for the estimation of RTT and the classification of (SLAs). This model transforms complex technical measurements into linguistically interpretable terms, enabling clearer SLA assessments and more user-friendly decision-making processes.

The results of this research have been published in the following conference proceedings:

- Sekhi, I. (2023). *Estimating Cloud Computing RTT Using Fuzzy Logic for Inter-Region Distances*. International Journal on Cybernetics & Informatics (IJCI), 12(12), 95.
- Sekhi, I. (2023). *Selecting the SLA Guarantee by Evaluating the QoS Availability*. Multidiszciplináris Tudományok: A Miskolci Egyetem Közleménye, 13(4), 80–102. https://doi.org/10.35925/j.multi.2023.4.8

Thesis II: Intelligent Validation Cloud Broker System (IVCBS)

I have created the (IVCBS), a fuzzy logic-based framework designed to optimize (VM) allocation and improve cloud computing efficiency. The system dynamically adjusts VM distribution based on the analysis of incoming request packet sizes, enhancing resource utilization, reducing latency, and maintaining consistent service quality.

The outcomes of this research have been documented in the following journals:

- Sekhi, I., & Nehéz, K. (2024). *Intelligent SLA Selection Through the Validation Cloud Broker System*. IEEE Access. DOI: 10.1109/ACCESS.2024.3439617

- Sekhi, I. (Accepted). *Efficient Broker-Driven Request Packet Size*. International Journal on Informatics Visualization.

Additionally, related foundational concepts and fuzzy logic optimization techniques were published in:

- Sekhi, I., Kovács, S., & Nehéz, K. (2025). *Enhancing Decision-Making in Uncertain Domains through Optimized Fuzzy Logic Systems*. Periodica Polytechnica Electrical Engineering and Computer Science, 69(1), 63–78. https://doi.org/10.3311/PPee.38729

Thesis III: Intelligent Cloud Brokerage System

I have designed an **Intelligent Cloud Brokerage System** that combines fuzzy logic with the Technique for Order Preference by Similarity to Ideal Solution (TOPSIS) to optimize cloud service selection and resource management across multiple (CSPs). This intelligent brokerage system serves as an intermediary, aligning user requirements with provider capabilities to improve service quality, cost efficiency, and operational performance.

The findings related to this research are published in:

- Sekhi, I. R., Abdah, H., & Nehéz, K. (2025). *Reliable and Cost-Effective Fuzzy-Based Cloud Broker*. International Journal of Networked and Distributed Computing, 13(1), 1–9. https://doi.org/10.1007/s44227-024-00052-x

These three theses collectively address critical challenges in cloud computing, contributing innovative solutions for enhancing performance, reducing latency, and improving the efficiency of resource management. The integration of fuzzy logic and advanced decision-making techniques in my research provides new pathways for achieving scalable, reliable, and cost-effective cloud services.

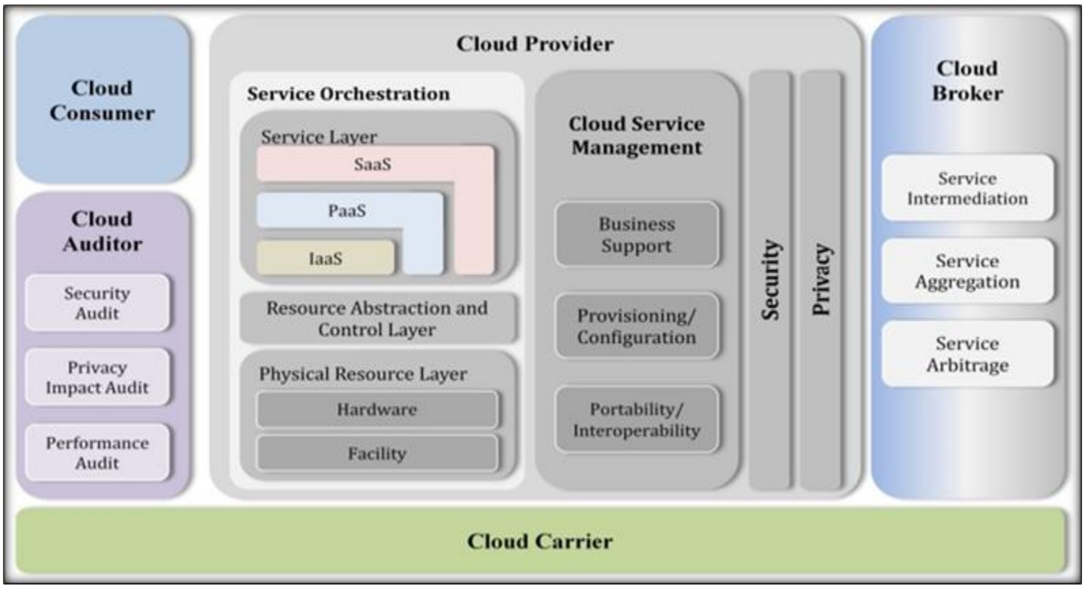
10.1 Future Research Direction

- Future research should focus on integrating IoT, edge computing, and 5G to enhance cloud computing scalability and interoperability. Real-world testing is crucial to evaluate performance, adaptability, and SLA management. Incorporating ML and fuzzy logic can optimize SLA classification and QoS adjustments, improving efficiency and reliability. Additionally, adaptive traffic management should be explored to enhance QoS, resource allocation, and fault recovery. Further research on SLA prioritization will optimize cloud resource utilization and user satisfaction. These advancements will contribute to intelligent, adaptive, and efficient cloud brokerage systems, ensuring better service selection and resource optimization in dynamic cloud environments.
- Enhance cross-cloud compatibility through standardized integration methods, ensuring seamless workload distribution across heterogeneous platforms for individual users and enterprises. This will also improve energy efficiency, reducing data centers' carbon footprint while maintaining high performance. Leveraging ML-driven workload distribution enables real-time optimization, dynamically adapting to service demands and enhancing resource efficiency. Addressing security and compliance challenges is crucial to mitigating vulnerabilities, improving data privacy, and maintaining

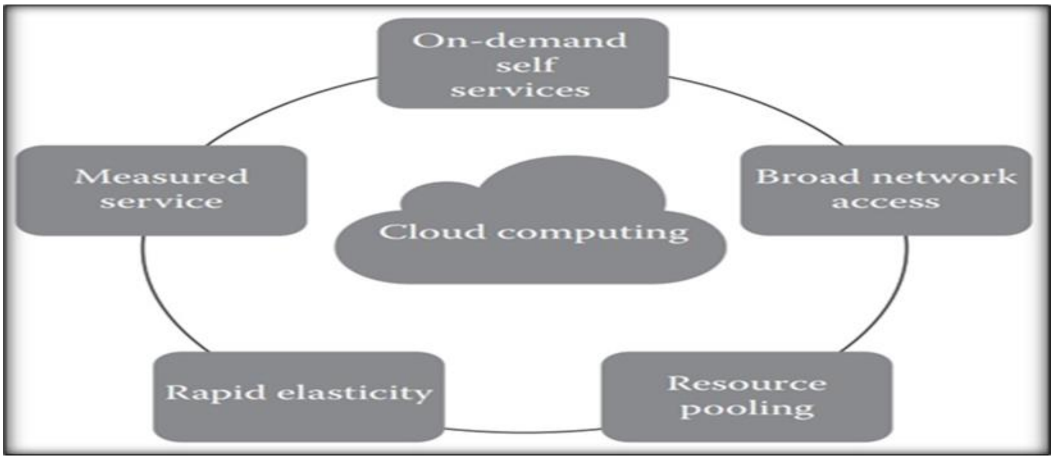
regulatory standards in multi-cloud environments. Additionally, context-aware decision-making in cloud brokerage systems should incorporate application delay sensitivity and client mobility profiles. Developing adaptive mechanisms to adjust resource allocation dynamically will help mitigate network delay, ensuring seamless service quality, minimal latency, and optimal performance in mobile cloud environments.

Appendices

Appendix 1: Cloud Computing

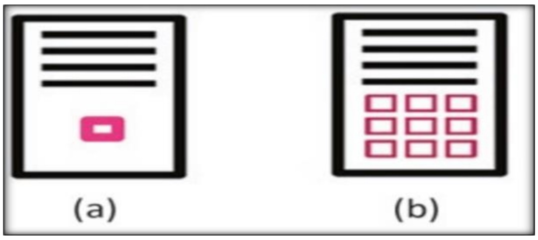


Appendix 1: 0.1 Figure 1. NIST Cloud Computing reference model.

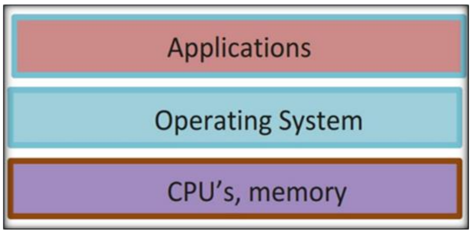


Appendix 1: 0.2 Figure 2. The essential characteristics of cloud computing.

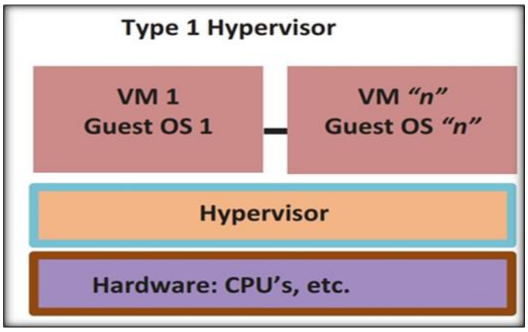
Appendix 2: Adoption and Implementation of Cloud Platforms



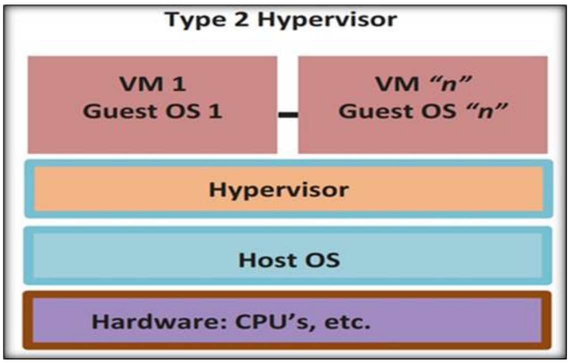
Appendix 2: 0.1 Figure 1. (a) Single application server. (b) Virtualized server.



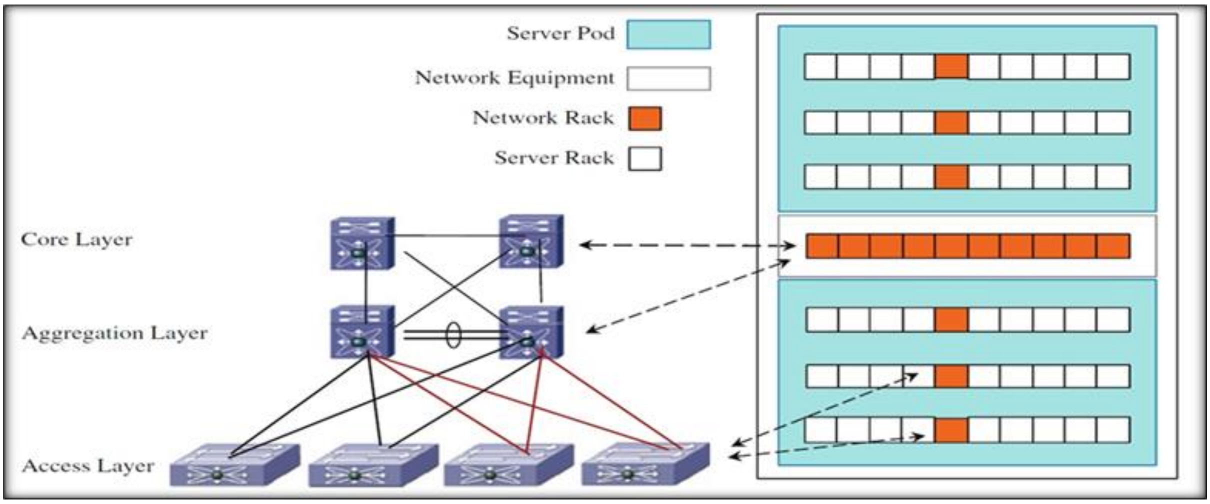
Appendix 2: 0.2 Figure 2. Hardware server components.



Appendix 2: 0.3 Figure 3. Type1 hypervisor.



Appendix 2: 0.4 Figure 4. Type2 hypervisor.



Appendix 2: 0.5 Figure 5. Data center network architecture.

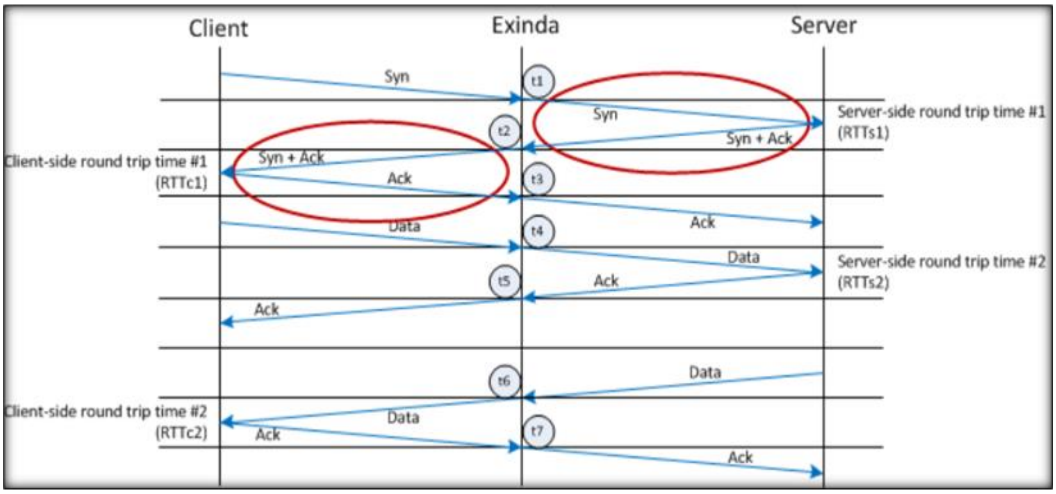
Appendix 2: 0.6 Table 1. Key Contractual Elements of an Infrastructural SLA.

Hardware availability month	99% uptime in a calendar month
Power availability	99.99% of the time in a calendar month
Data center network availability	99.99% of the time in a calendar month
Backbone network availability	99.999% of the time in a calendar month
Service credit for unavailability	Refund of service credit prorated on downtime period
Outage notification guarantee	Notification of customer within 1 hr. of complete downtime
Internet latency guarantee	When latency is measured at 5-min intervals to an upstream provider, the average doesn't exceed 60 msec
Packet loss guarantee	Shall not exceed 1% in a calendar month

Appendix 2: 0.7 Table 2. Key contractual components of an application SLA.

Service-level parameter metric	• Web site response time (e.g., max of 3.5 sec per user request) • Latency of web server (WS) (e.g., max of 0.2 sec per request) • Latency of DB (e.g., max of 0.5 sec per query)
Function	• Average latency of WS= (latency of web server 1+latency of web server 2) /2 • Web site response time= Average latency of web server+ latency of database
Measurement directive	• DB latency available via http://mgmtserver/em/latency • WS latency available via http://mgmtserver/ws/instanceno/latency
Service-level objective	Service Assurance
Penalty	• web site latency, 1 sec when concurrent connection, 1000 Penalty. • 1000 USD for every minute while the SLO was breached

Appendix 3: Triangular Membership Function-Based Estimation of Round-Trip Time (RTT) for Optimal SLA Evaluation



Appendix 3: 0.1 Figure 1. RTT process.

The RTT calculation, The ensuing diagram and equations provide a visual representation of how the round-trip time is computed

Server RTT:

- RTTs1 = t2 - t1
- RTTs2 = t5 - t4

Client RTT:

- RTTc1 = t3 - t2
- RTTc2 = t7 - t6

Average Server RTT = (RTTs1 + RTTs2)/2

Average Client RTT = (RTTc1 + RTTc2)/2

Average Total RTT = avRTTs + avRTTc

```
C:\>ping us-east-2.console.aws.amazon.com

Pinging af2049b9c08c62706.awsglobalaccelerator.com [13.248.199.77] with 32 bytes of data
Reply from 13.248.199.77: bytes=32 time=59ms TTL=237
Reply from 13.248.199.77: bytes=32 time=47ms TTL=237
Reply from 13.248.199.77: bytes=32 time=47ms TTL=237
Reply from 13.248.199.77: bytes=32 time=45ms TTL=237

Ping statistics for 13.248.199.77:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 45ms, Maximum = 59ms, Average = 49ms

C:\>ping us-east-1.console.aws.amazon.com

Pinging a508c136b61c3cfc2.awsglobalaccelerator.com [3.3.9.1] with 32 bytes of data:
Reply from 3.3.9.1: bytes=32 time=56ms TTL=237
Reply from 3.3.9.1: bytes=32 time=50ms TTL=237
Reply from 3.3.9.1: bytes=32 time=52ms TTL=237
Reply from 3.3.9.1: bytes=32 time=56ms TTL=237

Ping statistics for 3.3.9.1:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 50ms, Maximum = 56ms, Average = 53ms

C:\>ping us-west-1.console.aws.amazon.com

Pinging acfd892fef6fe535d.awsglobalaccelerator.com [75.2.51.23] with 32 bytes of data:
Reply from 75.2.51.23: bytes=32 time=62ms TTL=238
Reply from 75.2.51.23: bytes=32 time=43ms TTL=238
Reply from 75.2.51.23: bytes=32 time=43ms TTL=238
Reply from 75.2.51.23: bytes=32 time=56ms TTL=238

Ping statistics for 75.2.51.23:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 43ms, Maximum = 62ms, Average = 51ms
```

Appendix 3: 0.2 Figure 2. Ping testing process.

Appendix 3: 0.3 Figure 3. AWS latency test.

Appendix 3: 0.4 Table 1. Distances from Wasit Governorate to all AWS regions.

No	Region name	Distance (KM)	Latitude	Longitude	Endpoint
1	Bahrain	862.94	26.0667	50.5577	ec2.me-south-1.amazonaws.com
2	UAE – Dubai	1234.23	25.276987	55.296249	ec2.me-central-1.amazonaws.com
3	Mumbai	3089.72	19.0760	72.8777	ec2.ap-south-1.amazonaws.com
4	Milan	3428.79	45.4642	9.1900	ec2.eu-south-1.amazonaws.com

This table extends and complements the information presented in Appendix 3: 0.4 Table 1.					
5	Zurich	3525.01	47.3769	8.5417	ec2.eu-central-2.amazonaws.com
6	Frankfurt	3601.23	50.1109	8.6821	ec2.eu-central-1.amazonaws.com
7	Paris	3607.54	48.8566	2.3522	ec2.eu-west-3.amazonaws.com
8	London	4009.87	51.5074	-0.1278	ec2.eu-west-2.amazonaws.com
9	Spain	4202.65	41.6488	-0.8891	ec2.eu-south-2.amazonaws.com
10	Ireland	4238.49	53.3331	-6.2489	ec2.eu-west-1.amazonaws.com
11	Stockholm	4682.33	59.3293	18.0686	ec2.eu-north-1.amazonaws.com
12	Hong Kong	5981.25	22.3193	114.1694	ec2.ap-east-1.amazonaws.com
13	Hyderabad	6012.87	17.3850	78.4867	ec2.ap-south-2.amazonaws.com
14	Osaka	6789.34	34.6937	135.5023	ec2.ap-northeast-3.amazonaws.com
15	Seoul	7056.22	37.5665	126.9780	ec2.ap-northeast-2.amazonaws.com
16	Singapore	7289.64	1.3521	103.8198	ec2.ap-southeast-1.amazonaws.com
17	Tokyo	7435.78	35.6895	139.6917	ec2.ap-northeast-1.amazonaws.com
18	Jakarta	7832.90	-6.2088	106.8456	ec2.ap-southeast-3.amazonaws.com
19	Kuala Lumpur	8053.21	3.1390	101.6869	ec2.ap-southeast-4.amazonaws.com
20	Canada Central – Ottawa	8923.45	45.4215	-75.6972	ec2.ca-central-1.amazonaws.com
21	N. Virginia	10023.67	38.0336	-78.5080	ec2.us-east-1.amazonaws.com
22	Ohio	10289.47	39.9612	-82.9988	ec2.us-east-2.amazonaws.com
23	N. California	12345.89	37.7749	-122.4194	ec2.us-west-1.amazonaws.com
24	Oregon	12678.56	45.5234	-122.6762	ec2.us-west-2.amazonaws.com
25	Melbourne	13756.90	-37.8136	144.9631	ec2.ap-southeast-4.amazonaws.com
26	Sydney	14321.76	-33.8688	151.2093	ec2.ap-southeast-2.amazonaws.com

27	Cape Town	14989.34	-33.9249	18.4241	ec2.af-south-1.amazonaws.com
28	São Paulo	15478.65	-23.5505	-46.6333	ec2.sa-east-1.amazonaws.com

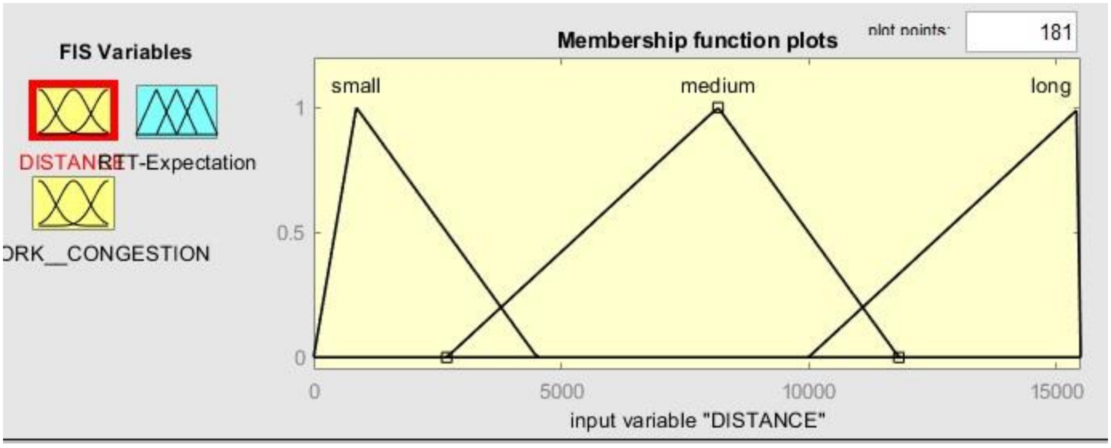
❖ Haversine Formula

The formula to compute the distance d between two points (lat1, lon1) and (lat2, lon2) is:

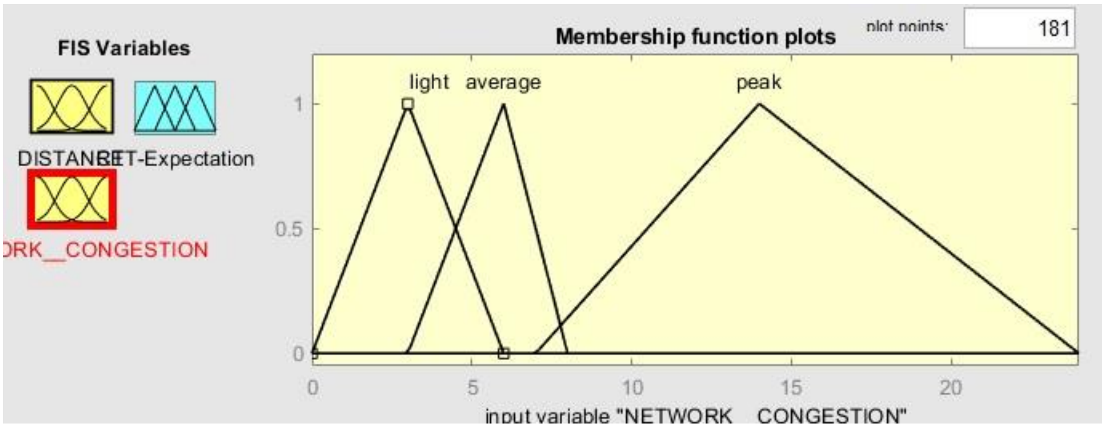
$$d = 2R.\arcsin\left(\sqrt{sin^2\left(\frac{\Delta\varphi}{2}\right) + \cos(\varphi 1).\cos(\varphi 2).sin^2\left(\frac{\Delta\lambda}{2}\right)}\right)$$

Where:

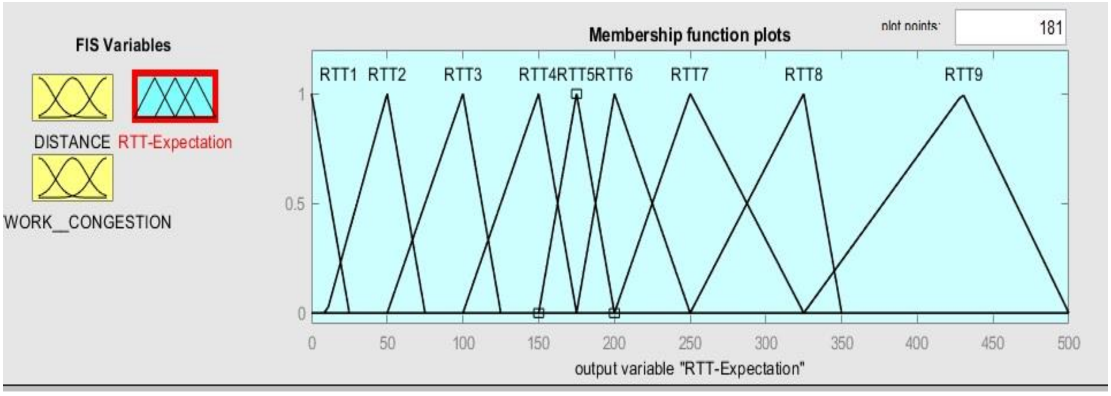
- d = distance between the two points (in kilometers or miles).
- R = Earth's radius (mean radius = 6371 km or 3958.8 miles).
- φ1, φ2 = latitudes of the two points in radians.
- λ1, λ2 = longitudes of the two points in radians.
- Δφ =φ2−φ1 (difference in latitudes).
- Δλ =λ2−λ1 (difference in longitudes).



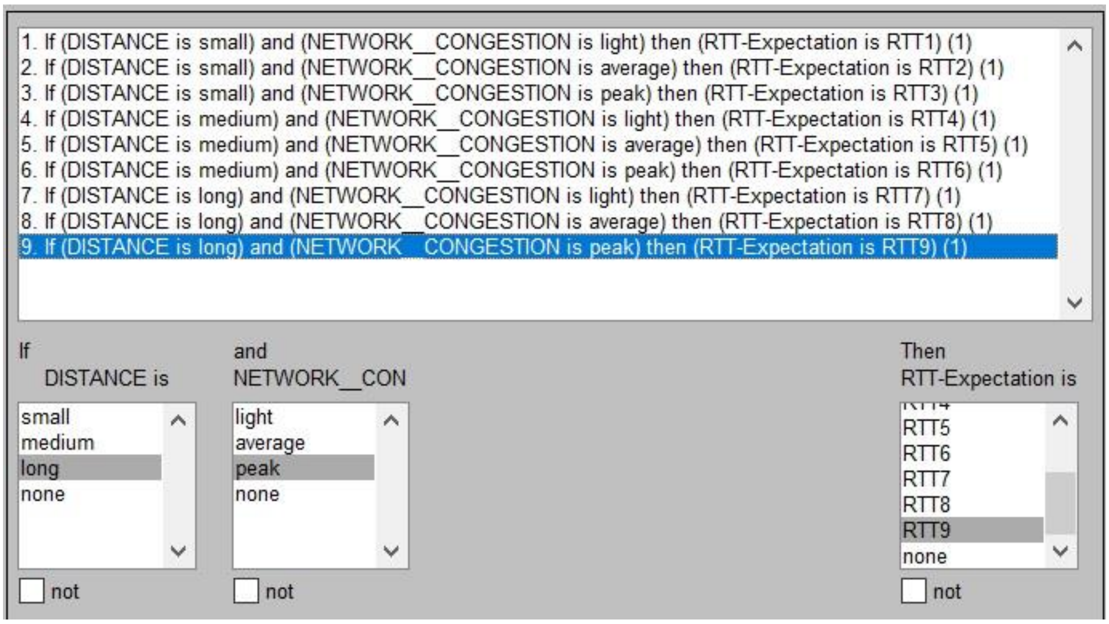
Appendix 3: 0.5 Figure 4. Define first input (Distance).



Appendix 3: 0.6 Figure 5. Define second input (Network-congestion).



Appendix 3: 0.7 Figure 6. Define Output (RTT-Expectation).



Appendix 3: 0.8 Figure 7. Rule base system.

Appendix 4: Quality of Service (QoS) Availability Assessment for Optimal SLA Selection

Appendix 4: 0.1 Table 1. Maximum allowable downtime for different availability levels.

Years of continuous operations	1	2	3
Availability	Maximum allowable downtime		
99.0000% (2–9s)	3 d 15 h 36 min 0 s	7 d 7 h 12 min 0 s	10 d 22 h 48 min 0 s
99.9000% (3–9s)	8 h 45 min 15 s	17 h 31 min 12 s	1 d 2 h 16 min 48 s
99.9900% (4–9s)	52 min 34 s	1 h 45 min 7 s	2 h 37 min 41 s

99.9990% (5–9s)	5 min 15 s	10 min 31 s	15 min 46 s
99.9999% (6–9s)	32 s 1 min 3 s	1 min 3 s	1 min 35 s

Appendix 4: 0.2 Table 2. The universe of discourse for both inputs.

The universe of discourse for both (Computing and networking) inputs		
90	93.39966	96.79932
90.09999	93.49965	96.89931
90.19998	93.59964	96.9993
90.29997	93.69963	97.09929
90.39996	93.79962	97.19928
90.49995	93.89961	97.29927
90.59994	93.9996	97.39926
90.69993	94.09959	97.49925
90.79992	94.19958	97.59924
90.89991	94.29957	97.69923
90.9999	94.39956	97.79922
91.09989	94.49955	97.89921
91.19988	94.59954	97.9992
91.29987	94.69953	98.09919
91.39986	94.79952	98.19918
91.49985	94.89951	98.29917
91.59984	94.9995	98.39916
91.69983	95.09949	98.49915
91.79982	95.19948	98.59914

This table extends and complements the information presented in Appendix 4: 0.2 Table 2.		
91.89981	95.29947	98.69913
91.9998	95.39946	98.79912
92.09979	95.49945	98.89911
92.19978	95.59944	98.9991
92.29977	95.69943	99.09909
92.39976	95.79942	99.19908
92.49975	95.89941	99.29907
92.59974	95.9994	99.39906
92.69973	96.09939	99.49905
92.79972	96.19938	99.59904
92.89971	96.29937	99.69903
92.9997	96.39936	99.79902
93.09969	96.49935	99.89901
93.19968	96.59934	99.999
93.29967	96.69933	

Appendix 4: 0.3 Table 3. Proposed Uptime and downtime.

Uptime%	Day Uptime	Day Downtime	Week Uptime	Week Downtime	Month Uptime	Month Downtime	Year Uptime	Year Downtime
90 %	21:36:00	2:24:00	151:12:00	16:48:00	648:00:00	72:00:00	7884:00:00	876:00:00
91%	22:04:47	1:55:12	154:33:34	13:26:25	662:23:54	57:36:05	8059:10:56	700:49:03

99.999	98%	97%	96%	95%	94%	93%	92%
23:59:59	23:45:35	23:31:11	23:16:47	23:02:23	22:47:59	22:33:35	22:19:11
0:00:00	0:14:24	0:28:48	0:43:12	0:57:36	1:12:00	1:26:24	1:40:48
167:59:53	166:19:06	164:38:19	162:57:31	161:16:44	159:35:56	157:55:09	156:14:22
0:00:06	1:40:53	3:21:40	5:02:28	6:43:15	8:24:03	10:04:50	11:45:37
719:59:34	712:47:36	705:35:39	698:23:41	691:11:44	683:59:47	676:47:49	669:35:52
0:00:25	7:12:23	14:24:20	21:36:18	28:48:15	36:00:12	43:12:10	50:24:07
8759:54:44	8672:19:16	8584:43:47	8497:08:19	8409:32:50	8321:57:22	8234:21:53	8146:46:25
0:05:15	87:40:43	175:16:12	262:51:40	350:27:09	438:02:37	525:38:06	613:13:34

- EX: In equation form for 90% uptime in a single day:

Uptime in seconds:

Uptime=Total Time per day × Uptime percentage; Where:

Total Time per day = 86,400 seconds (for 24 hours),

Uptime percentage = 0.90 for 90%.

Downtime in second:

Downtime=Total Time per day × (1- Uptime percentage); Where:

Downtime percentage=1 - 0.90

Downtime= 0.10

Then In equation form for 90% Uptime in a single day:

Uptime = 86,400 × 0.90 =77,760 seconds

Downtime = 86,400 × (1-0.90)

Downtime = 8,640 seconds

To convert seconds into hours, minutes, and seconds:

- Uptime:77,760 seconds =21 hours,36 minutes.
- Downtime:8,640 seconds = 2 hours,2 minutes.

These equations provide a clear way to calculate uptime and downtime for any percentage of uptime over any given period (e.g., a day, week, month, or year).

Appendix 5: Implementation details of the three proposed algorithms for the system

Appendix 5:0.1 Detailed Analysis of the First Algorithm

- *Maximum value: 67,170*
- *Point1 = Maximum value / 4*
- *Point2 = 2 * Point1*
- *Point3 = 3 * Point1*
- *Point4 = 4 * Point1*
- μ_{small}: *[0 0 point2]*
- $\mu_{medium:}$ *[point1 point2 point3]*
- $\mu_{big:}$ *[point2 point4 point4]*
- When 0 ≤ value ≤point1

Consider input value is 165

Calculate Small Membership function:

μ_{small} (165) *=(- value/point2)+1*

μ_{small} (165) *=(-165/33585)+1*

μ_{small} (165) = *-0.00491+1*

μ_{small} (165) = *0.995087092*

" μ_{medium} (165) *" remains 0 since the input value falls within the 0 to Point1 range.*

" μ_{big} (165) *" remains 0 since the input value falls within the 0 to Point1 range.*

- When point1 ≤ value ≤point2 then:

Consider input value is 20892

Calculate Small Membership function:

μ_{small} (20892) = (- value/point2)+1

μ_{small} (20892) = - 0.6218+1

μ_{small} (20892) = 0.377936579

Calculate α

α= value – point2

α=20892 – 33585

α= - 12693

Calculate medium Membership function:

μ_{medium} (20892) = (-1/point2 - point1). | α |+1

μ_{medium} (20892) = (-1/33585– 16792.5). |12693|+1

μ_{medium}(20892)= (-1/16792.5) . 12693+1

μ_{medium}(20892)= - 0.7560+1

μ_{medium}(20892)=0.244126842

" μ_{big}(20892)" *remains 0 since the input value falls within the Point1 to Point2 range.*

Appendix 5:0.2 Detailed Analysis of the Second Algorithm

- *Maximum value: 67,170*
- *Point1 = Maximum value / 5*
- *Point2 = 2 * Point1*
- *Point3 = 3 * Point1*
- *Point4 = 4 * Point1*
- *Point5 =5 * point1*
- μ_{small}: [0 0 point1 point2]
- $\mu_{medium:}$ [point1 point2 point3 point4]
- $\mu_{big:}$ [point3 point4 point4 point5]
- *When* $0 \leq value \leq point1$ *then:*

μ_{small} (value) = 1

" μ_{medium} (value) " *remains 0 since the input value falls within the 0 to Point1 range.*

" μ_{big}(value) " *remains 0 since the input value falls within the 0 to Point1 range.*

- *When* $point1 \leq value \leq point2$

Consider input value is 17132

Calculate Small Membership function degree:

μ_{small} (value) = (- value/point2)+1

μ_{small} *(17132) = (- 17132/33585)+1*

μ_{small} *(17132) = - 0.6376+1*

μ_{small} *(17132) = 0.362364151*

Calculate α:

α= value – point2

α=17132 – 26868

α= - 9736

Calculate medium Membership function degree:

μ_{medium} *(17132) = (-1/point2 - point1). | α |+1*

μ_{medium} *(17132) = (-1/26868 – 13434). |- 9736 |+1*

μ_{medium} *(17132) = (-1/13434). 9736+1*

μ_{medium} *(17132) = -0.7248+1*

μ_{medium} *(17132) =0.275271699*

" μ_{big} *(17132)" remains 0 since the input value falls within the Point1 to Point2 range.*

Appendix 5:0.3 Detailed Analysis of the Third Algorithm

- *Maximum value: 67,170*
- *Point1=0*
- *Point2=Maximum value/2*
- *Point4=Maximum value*
- *Standard Deviation* σ *=16339*
- *Small center=* $c_{small=point1}$
- μ_{small}*:* $[\sigma \quad point1]$
- *Medium center=* $c_{medium=point2}$

μ_{medium}*:* $[\sigma \quad point2]$

- *Big center=* $c_{big=point4}$

μ_{big}*:* $[\sigma \quad point4]$

Consider input value is 11381

- *Calculate Small membership function degree*

μ_{small} *(11381) =Exp* $(-(11381-0)^2/2.\ (16339)^2)$

Calculate the squared difference:

$(11381-0)^2=129564361$

Compute $2.\ \sigma^2=2.\ (16339)^2$

=533906642

Divide and apply the exponent:

μ_{small} *(11381) =Exp (-129564361/533906642)*

μ_{small} *(11381) =Exp (-0.2426)*
μ_{small}*(11381) =0.784590058*

- *Calculate Medium membership function degree*

μ_{medium} *(11381) =Exp* $(-(11381-33585)^2/2.(16339)^2)$

Calculate the squared difference:

$(11381-33585)^2=494383296$

Divide and apply the exponent:

μ_{medium} *(11381) =Exp (-494383296/533906642)*

μ_{medium} *=Exp(-0.9263)*

μ_{medium} *= 0.397173449*

- *Calculate Big membership function degree*

μ_{big} *(11381) =Exp* $(-(11381-67170)^2/2.\ (16339)^2)$

Calculate the squared difference:

$(11381-67170)^2=3104115681$

Divide and apply the exponent:

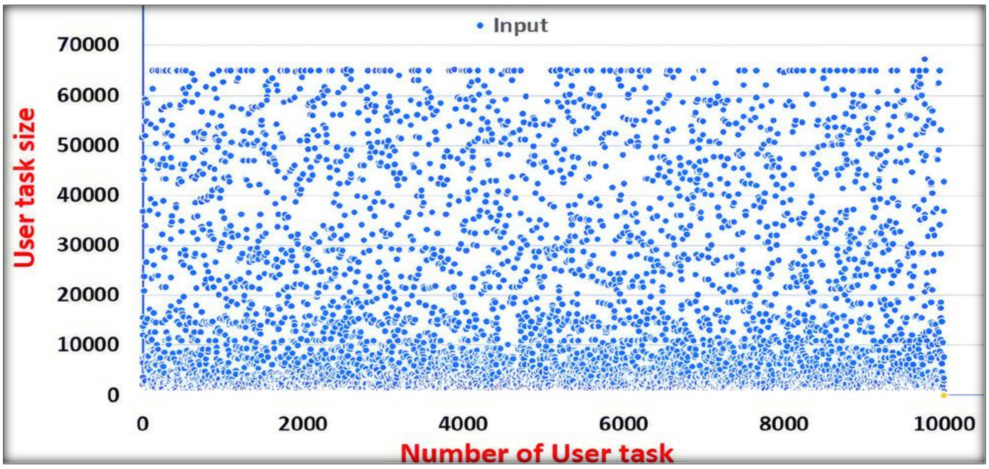
μ_{big} *(11381) =Exp (-3104115681/533906642)*

μ_{big} *(11381) =Exp (-5.8146)*
μ_{big} *(11381) = 0.002940142*

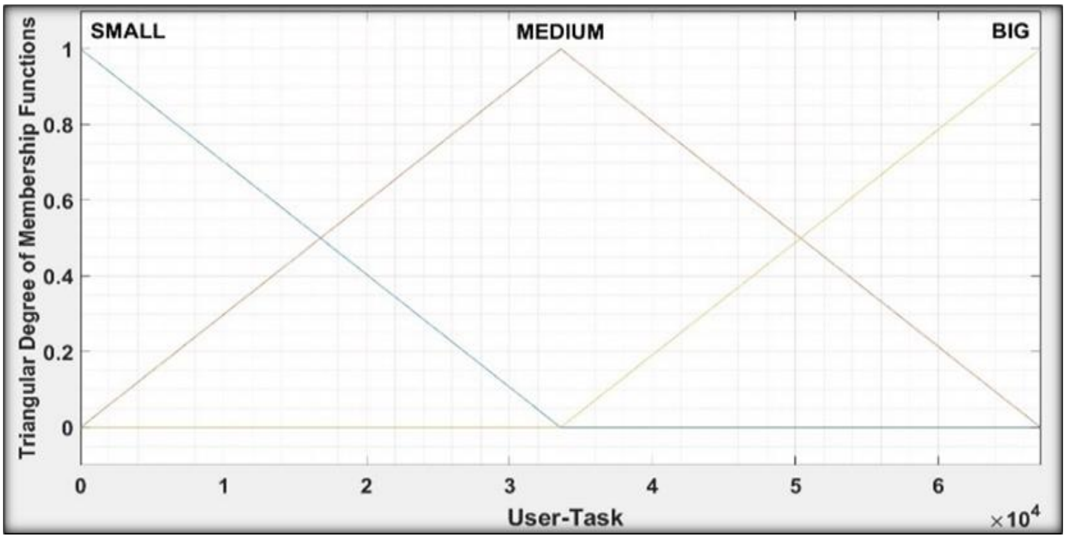
Appendix 6: Optimized Fuzzy Logic Systems for Enhanced Decision-Making in Uncertain Domains

Task	Number of Allocated processors	Average CPU time used	Request Time	User ID	Executable Number

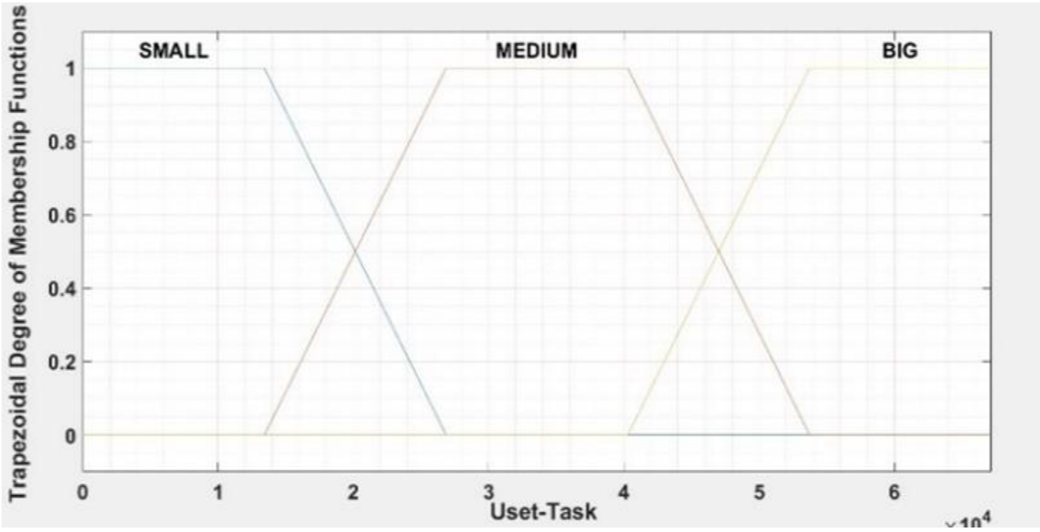
Appendix 6: 0.1 Figure 1. Database Addresses.



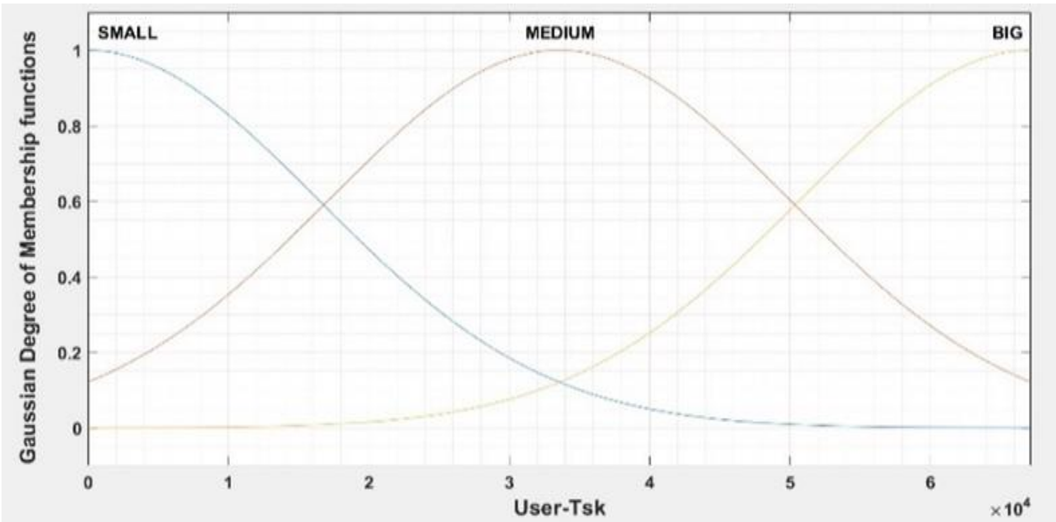
Appendix 6: 0.2 Figure 2. User task before classify.



Appendix 6: 0.3 Figure 3. Mamdani Triangular MF.



Appendix 6: 0.4 Figure 4. Mamdani Trapezoidal MF.



Appendix 6: 0.5 Figure 5. Mamdani Gaussian MF.

Appendix 7: Fuzzy Cloud Broker Validation System for SLA Selection Mechanisms

Appendix 7: 0.1 Table 1. AWS-General-Purpose series Attributes and specs.

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	(EBS)	AMD Transparent Single key Memory Encryption (TSME)

Appendix 7: 0.2 Table 2. AWS data centers and general costs.

6-Geographical (31-Regions)	EC2-General purpose cost										
	M6g.medum	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

This table extends and complements the information presented in Appendix 7: 0.2 Table 2.											
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
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

This table extends and complements the information presented in Appendix 7: 0.2 Table 2.

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
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

Appendix 7: 0.3 Table 3. 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

Appendix 7: 0.4 Table 4. 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- Hong Kong	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

Appendix 7: 0.5 Table 5. 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

Appendix 7: 0.6 Table 6. 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

This MATLAB code serves as a foundational tool for analyzing and improving cloud resource allocation, playing a crucial role in system enhancement, 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 FIS to explore the intelligent features built into the (IVCBS), 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,. This makes sure that the SLA-level classification is correct and reliable. Moreover, to demonstrate the alignment of our mathematical model with the trapezoidal membership function, referenced this approach in the discussion on initializing and depicting the membership function. This MATLAB code is crucial, serving as a foundational tool for the analysis and enhancement of cloud resource allocation.

```
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
```

Appendix 7:0.7 Figure 1. VCPU Classification code.

```
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

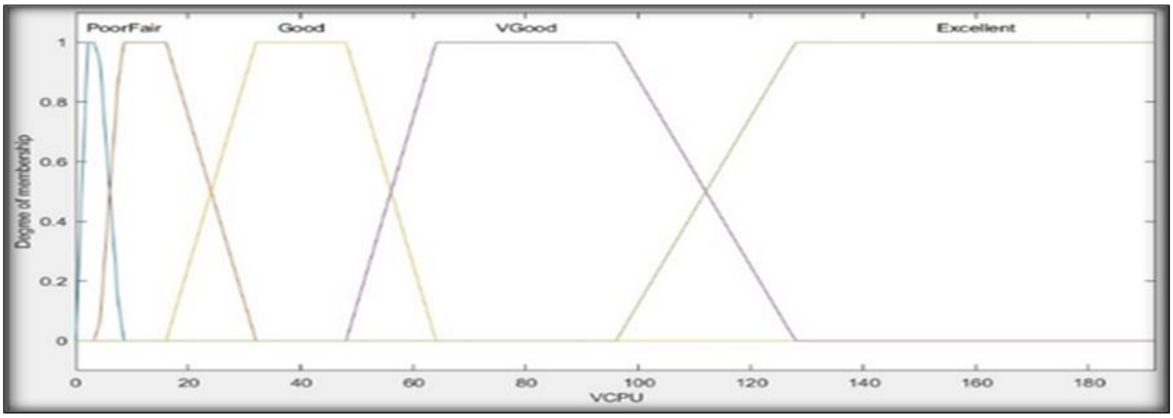
```
pV1 = 1; pV2 = 2; pV3 = 4; pV4 = 8; pV5 = 16;
pV6 = 32; pV7 = 48; pV8 = 64; pV9 = 96; pV10 = 128; pV11 = 192;
fis = addvar(fis, 'input', 'VCPU', [0 MAX]);
fis = addmf(fis, 'input', 1, 'Poor', 'trapmf', [pV1 pV2 pV3 pV4]);
fis = addmf(fis, 'input', 1, 'Fair', 'trapmf', [pV3 pV4 pV5 pV6]);
fis = addmf(fis, '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.

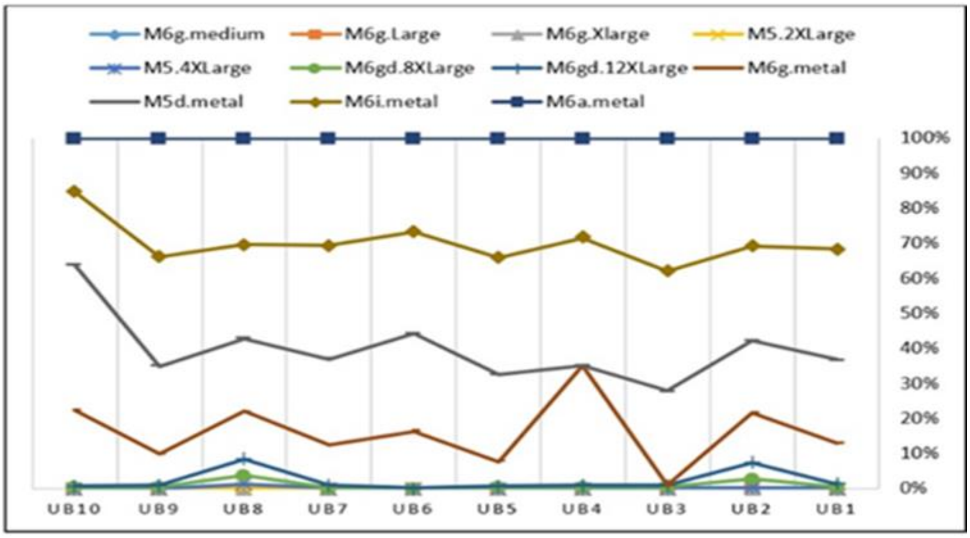
- Visualization

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

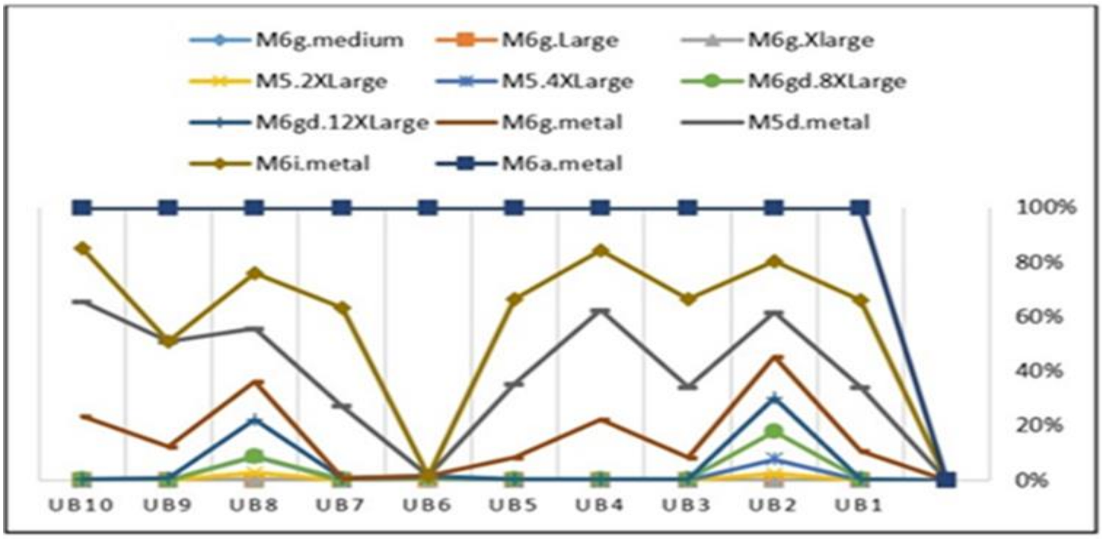
This visualizes the trapezoidal membership functions. Finally, the specific MATLAB software and libraries, along with the parameters and functions examined in the Intelligent Cloud Broker Validation System, were represented. After the broker finalizes the classification of user requests and SLA resources 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.



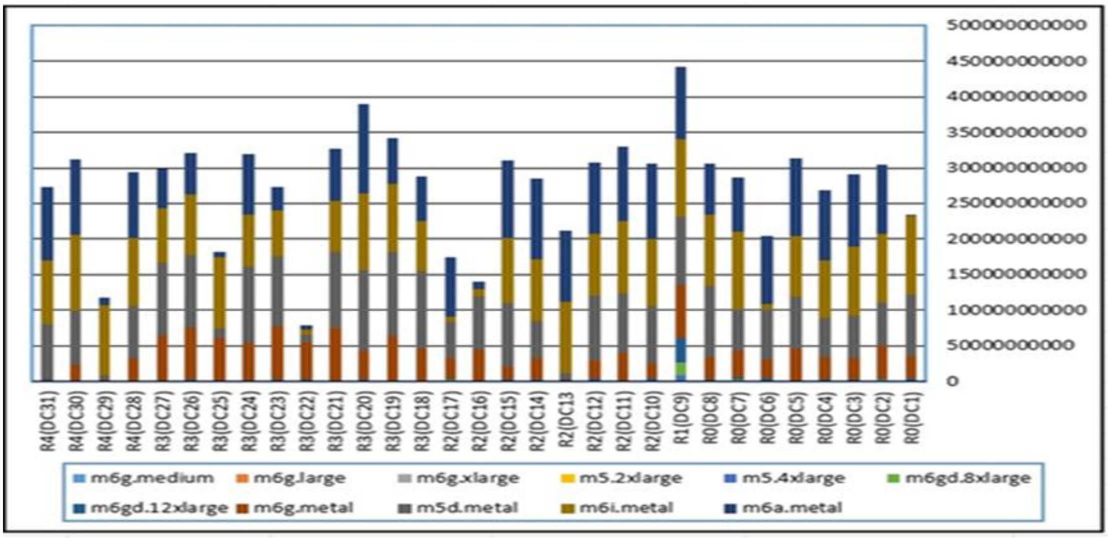
Appendix 7:0.8 Figure 2. Apply the Trapezoidal proposed model of CPU levels.



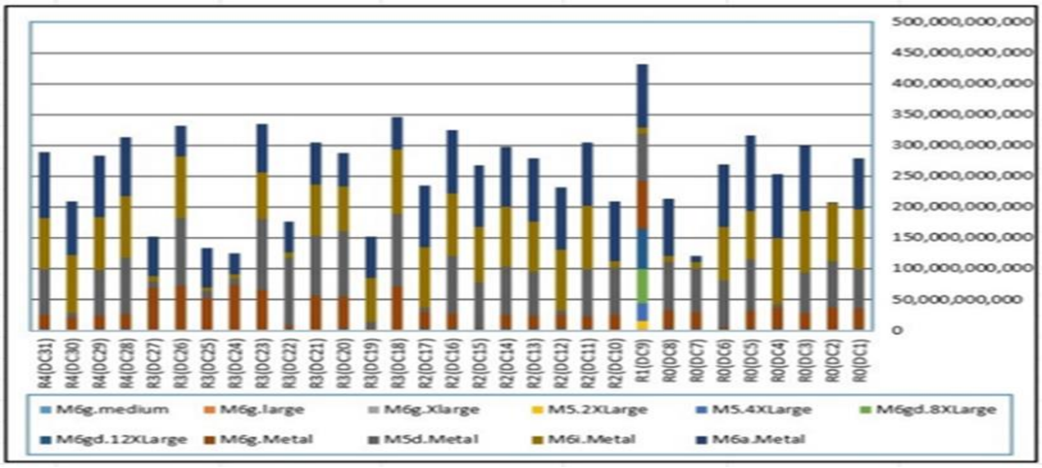
Appendix 7:0.9 Figure 3. IVCBS-Response time by region (optimize response time policy).



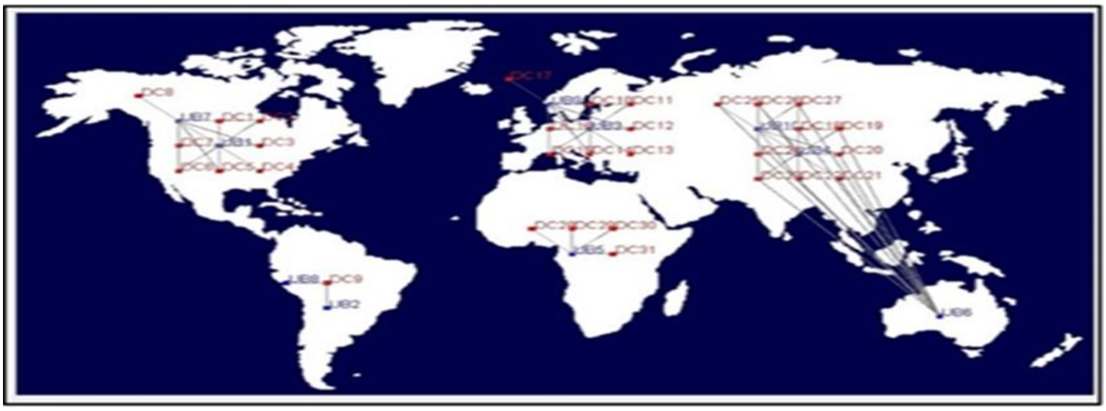
Appendix 7:1.0 Figure 4. IVCBS-Response time by region (reconfigure dynamically policy).



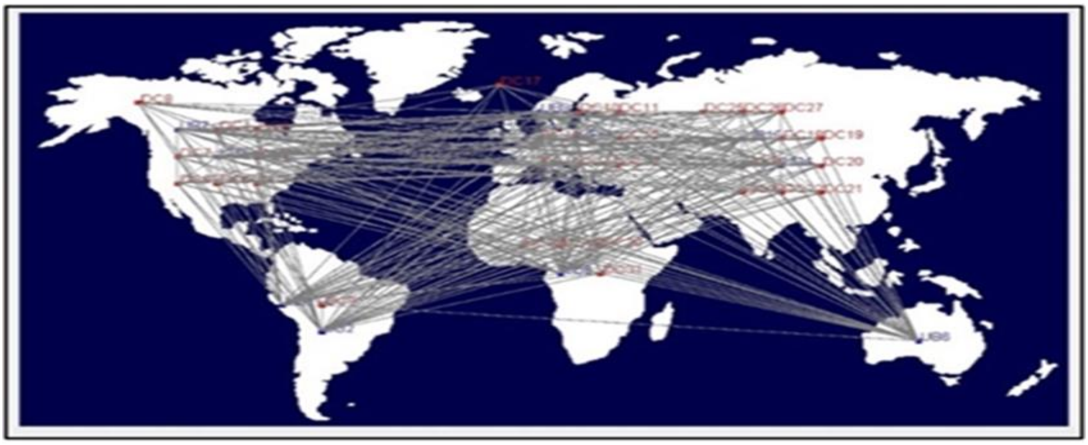
Appendix 7:1.1 Figure 5. IVCBS DC- Request Servicing Time (optimize response time policy).



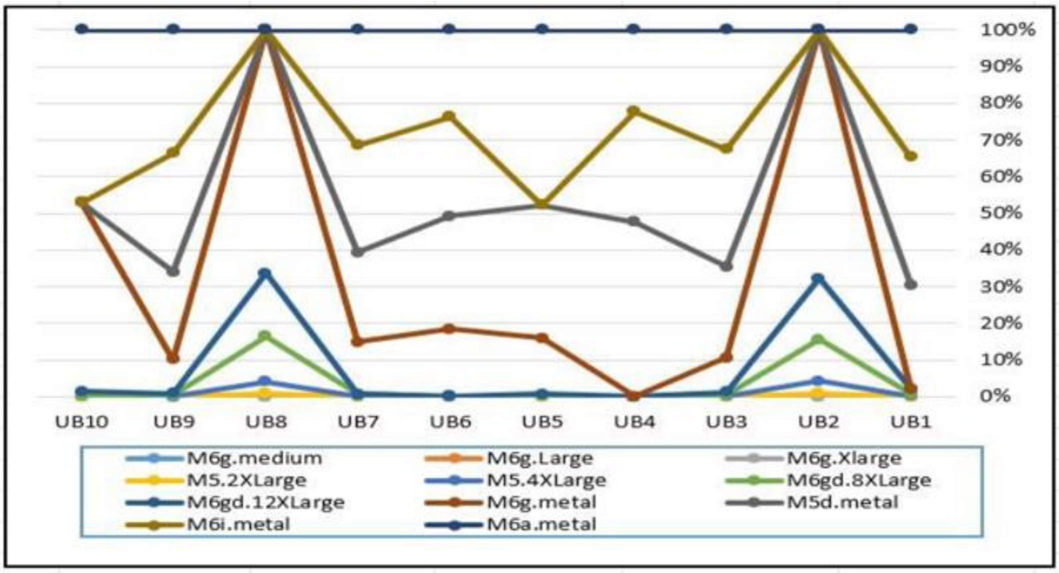
Appendix 7:1.2 Figure 6. IVCBS DC- Request Servicing Time (dynamic reconfiguration policy).



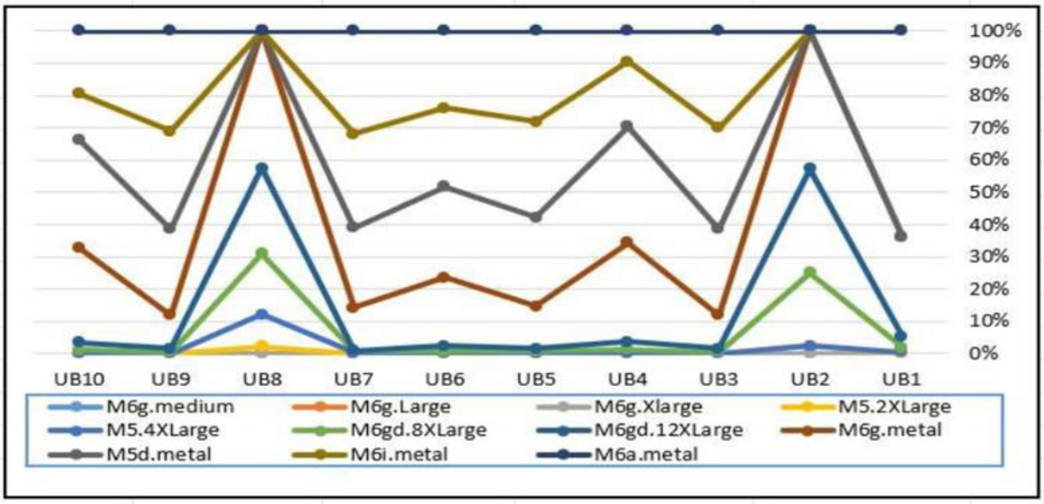
Appendix 7:1.3 Figure 7. Routing strategy by the dynamic reconfigurations policy.



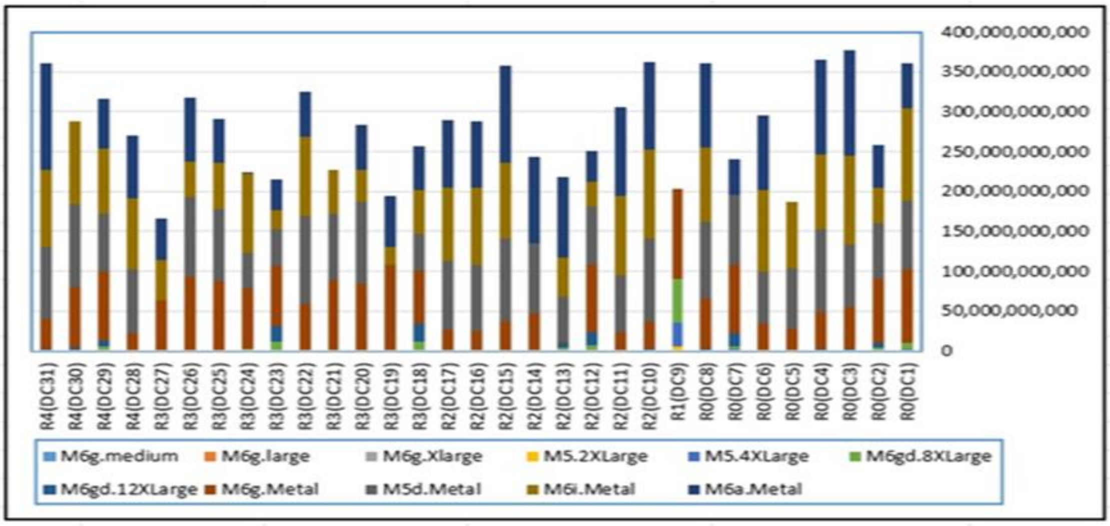
Appendix 7:1.4 Figure 8. Routing strategy by the optimized response time policy.



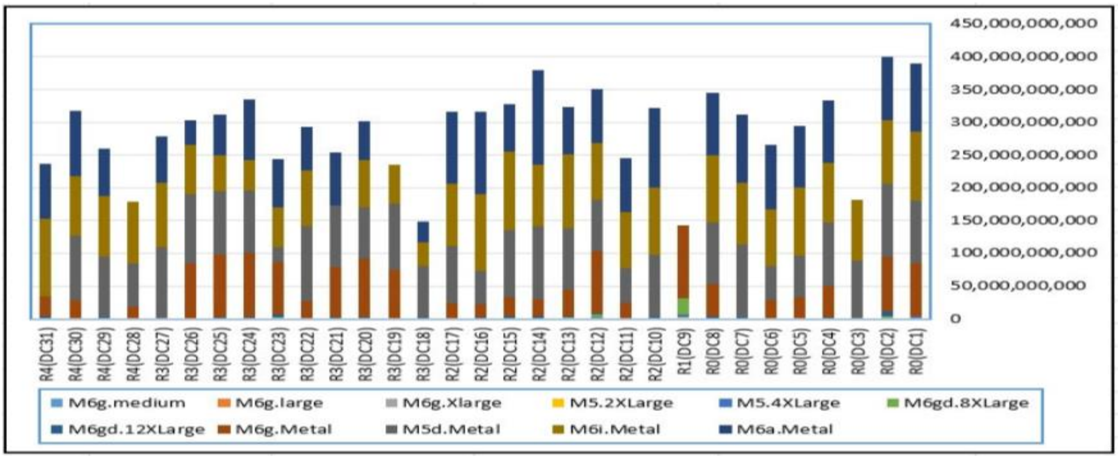
Appendix 7:1.5 Figure 9. Traditional-Response time by region (optimize response time policy).



Appendix 7:1.6 Figure 10. Traditional-Response time by region (reconfigure dynamically policy).

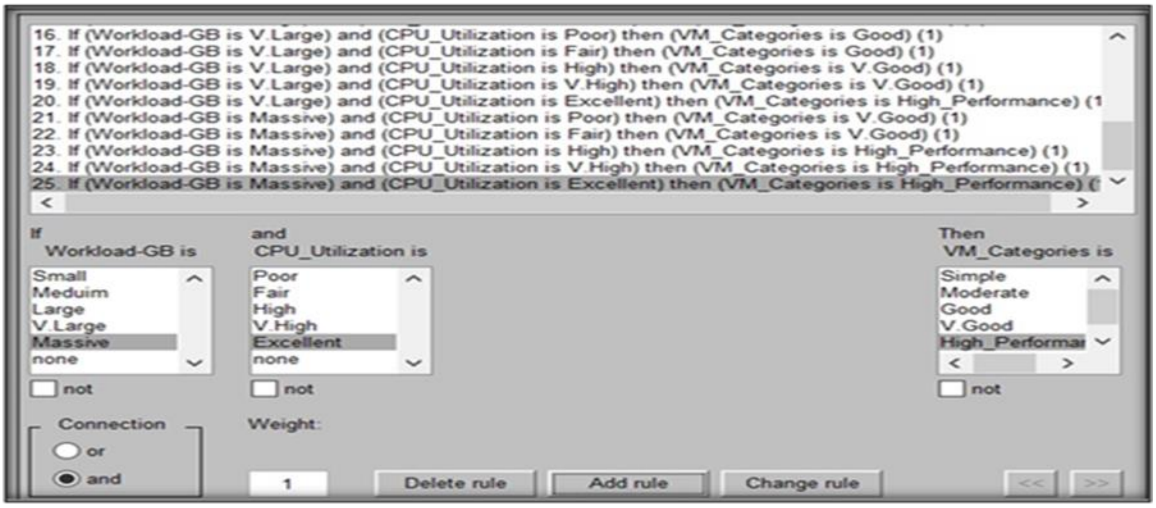


Appendix 7:1.7 Figure 11. Traditional DC- Request Servicing Time (optimize response time policy).



Appendix 7:1.8 Figure 12. Traditional DC- Request Servicing Time (dynamic reconfiguration policy).

Appendix 8: Optimizing Request Packet Size Through an Efficient Broker-Driven Approach



Appendix 8:0.1 Figure 1. Fuzzy rule base.

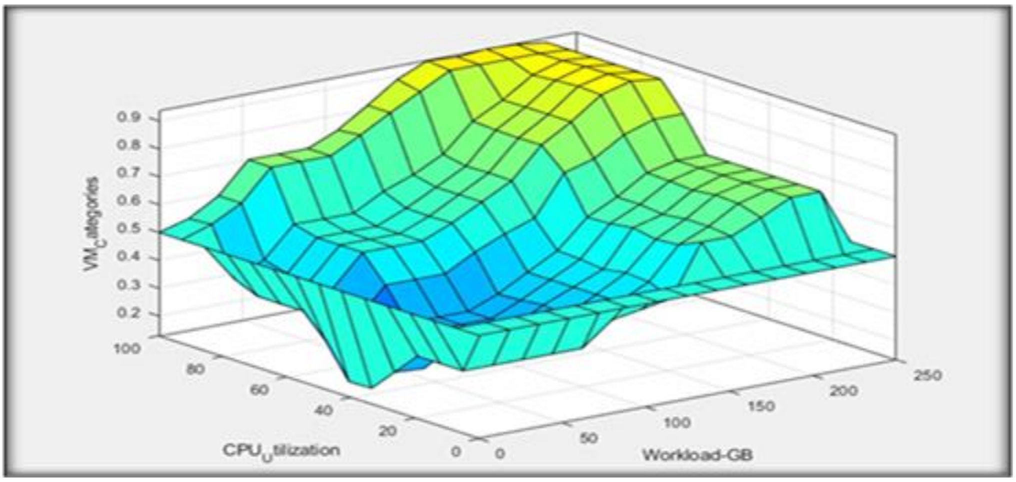
Appendix 8:0.2 Table 1. User base configuration.

User Bases	Geographic-Regions	Requests-per users per Hour	Peak Hours (GMT)		Avg peak users	Avg Off-peak users
			Start	End		
UB1 :1000	R0: North America	60	12	15	800	100
UB2 :1000	R1: South America	60	14	17	1000	100
UB3 :1000	R2: Europe	60	19	22	1000	100
UB4 :1000	R3: Asia	60	0	3	700	100
UB5 :1000	R4: Africa and middle east	60	20	23	900	100
UB6 :1000	R5: Africa	60	8	11	1000	100
UB7:1000	R0: North America	60	6	9	1000	100
UB8:1000	R1: South America	60	12	15	500	100
UB9 :1000	R2: Europe	60	18	21	750	100
UB10 :1000	R3: Asia	60	7	9	1000	100

Appendix 8:0.3 Table 2. Advanced VM configuration in a single data center.

Request Packet Size (Byte)	User factor in UBS	Request factor in DC	Executable request (byte)	Load balance Algorithm	Broker policy
500,000,000	1000	1000	1000,000	Throttling algorithm.	Optimize response time.
1,000,000,000	1000	1000	1000,000	Throttling algorithm.	Optimize response time.
10,000,000,000	1000	1000	1000,000	Throttling algorithm.	Optimize response time.
150,000,000,000	1000	1000	1000,000	Throttling algorithm.	Optimize response time.
200,000,000,000	1000	1000	1000,000	Throttling algorithm.	Optimize response time.

Appendix 8:0.4 Figure 2. Simulation process.



Appendix 8:0.5 Figure 3. Surface Viewer for Fuzzy Model Output.

References

[1] Aravinth, S. S., Krishnan, A. S. R., Ranganathan, R., Sasikala, M., Kumar, M. S., & Thiyagarajan, R. (2024). Cloud Computing—Everything as a Cloud Service in Industry 4.0. In Digital Transformation: Industry 4.0 to Society 5.0 (pp. 103-121). Singapore: Springer Nature Singapore.

[2] Qazi, F., Kwak, D., Khan, F. G., Ali, F., & Khan, S. U. (2024). Service Level Agreement in cloud computing: Taxonomy, prospects, and challenges. Internet of Things, 101126.

[3] Sissodia, R., Rauthan, M. S., & Barthwal, V. (2024). Service Level Agreements (SLAs) and Their Role in Establishing Trust. In Analyzing and Mitigating Security Risks in Cloud Computing (pp. 182-193). IGI Global.

[4] Fernandes, S. (2017). Performance evaluation for network services, systems and protocols (pp. 1-175). Heidelberg: Springer.

[5] Bose, R., Sengupta, S., & Roy, S. (2023). Interpreting SLA and related nomenclature in terms of Cloud Computing: a layered approach to understanding service level agreements in the context of cloud computing. Lambert Academic Publishing.

[6] Shan, L., Sun, L., & Rezaeipanah, A. (2024). Towards a novel service broker policy for choosing the appropriate data center in cloud environments. Computer Communications, 107939.

[7] Mendel, J., & Wu, D. (2010). Perceptual computing: aiding people in making subjective judgments. John Wiley & Sons.

[8] Tallón-Ballesteros, A. J., & Beltrán-Barba, R. (Eds.). (2023). Fuzzy Systems and Data Mining IX: Proceedings of FSDM 2023 (Vol. 378). IOS Press.

[9] Wickremasinghe, B., Calheiros, R. N., & Buyya, R. (2010, April). Cloudanalyst: A cloudsim-based visual modeller for analysing cloud computing environments and applications. In 2010 24th IEEE international conference on advanced information networking and applications (pp. 446-452). IEEE.

[10] Sonmez, C., Ozgovde, A., Ersoy, C.: Edgecloudsim: An environment for performance evaluation of edge computing systems. Transactions on Emerging Telecommunications Technologies 29(11), 3493 (2018).

[11] Rampérez, V., Soriano, J., Lizcano, D., Aljawarneh, S., & Lara, J. A. (2022). From SLA to vendor-neutral metrics: An intelligent knowledge-based approach for multi-cloud SLA-based broker. International Journal of Intelligent Systems, 37(12), 10533-10575.

[12] Malla, P. A., & Sheikh, S. (2023). Analysis of QoS aware energy-efficient resource provisioning techniques in cloud computing. International Journal of Communication Systems, 36(1), e5359.

[13] Palumbo, F., Aceto, G., Botta, A., Ciuonzo, D., Persico, V., & Pescapé, A. (2021). Characterization and analysis of cloud-to-user latency: The case of Azure and AWS. Computer Networks, 184, 107693.

[14] Choudhary, R., & Sharma, P. (2023). Data Transmission Reliability in Distributed Cloud Environments: Challenges and Solutions. Future Generation Computer Systems, 147, 300-315.

[15] Zhang, C., Wang, Y., Lv, Y., Wu, H., & Guo, H. (2019). An Energy and SLA-Aware Resource Management Strategy in Cloud Data Centers. Scientific Programming, 2019(1), 3204346.

[16] Navandar, R. K. (2024). Enhancing Cloud Computing Environments with AI-Driven Resource Allocation Models. Advances in Nonlinear Variational Inequalities, 27(3), 541-557.

[17] Rakib, A., & Uddin, I. (2019). An efficient rule-based distributed reasoning framework for resource-bounded systems. Mobile Networks and Applications, 24(1), 82-99.

[18] Faiz, M., & Daniel, A. K. (2024). A multi-criteria cloud selection model based on fuzzy logic technique for QoS. International Journal of System Assurance Engineering and Management, 15(2), 687-704.

[19] Reyes-García, C. A., & Torres-Garcia, A. A. (2022). Fuzzy logic and fuzzy systems. In Biosignal Processing and Classification Using Computational Learning and Intelligence (pp. 153-176). Academic Press.

[20] Kaur, H., & Gargrish, S. (2024). DRAP-CPU: a novel VM migration approach through a dynamic prioritized resource allocation strategy. Microsystem Technologies, 1-12.

[21] Buyya, R., Ilager, S., & Arroba, P. (2024). Energy-efficiency and sustainability in new generation cloud computing: A vision and directions for integrated management of data centre resources and workloads. Software: Practice and Experience, 54(1), 24-38.

[22] Al-E'mari, S., Sanjalawe, Y., Al-Daraiseh, A., Taha, M. B., & Aladaileh, M. (2024). Cloud Datacenter Selection Using Service Broker Policies: A Survey. CMES-Computer Modeling in Engineering & Sciences, 139(1).

[23] Kavis, M. (2014). Architecting the cloud: design decisions for cloud computing service models (SaaS, PaaS, and IaaS). John Wiley & Sons, Inc., Hoboken, New Jersey.
[24] Ikram, M. A., & Hussain, F. K. (2018). Software as a service (saas) service selection based on measuring the shortest distance to the consumer's preferences. In Advances in Internet, Data & Web Technologies: The 6th International Conference on Emerging Internet, Data & Web Technologies (EIDWT-2018) (pp. 403-415). Springer International Publishing.
[25] Liu, F., Tong, J., Mao, J., Bohn, R., Messina, J., Badger, L., & Leaf, D. (2011). NIST cloud computing reference architecture. NIST special publication, 500(2011), 1-28.
[26] Wang, L., Ranjan, R., Chen, J., & Benatallah, B. (Eds.). (2011). Cloud computing: methodology, systems, and applications. CRC press.
[27] Rountree, D., & Castrillo, I. (2013). The basics of cloud computing: Understanding the fundamentals of cloud computing in theory and practice. Newnes.
[28] Cloud, H. (2011). The nist definition of cloud computing. National institute of science and technology, special publication, 800(2011), 145.
[29] Chandrasekaran, K. (2014). Essentials of cloud computing. CrC Press.
[30] Kingsley, M. S. (2023). Cloud Technologies and Services: Theoretical Concepts and Practical Applications. Springer Nature.
[31] L'Esteve, R. C. (2023). The cloud leader's handbook: strategically innovate, transform, and scale organizations.
[32] Wagdy, M., Babulak, E., & Al-Dabass, D. (2021). Network function virtualization over cloud-cloud computing as business continuity solution. Intechopen, Published: July 14th.
[33] Hiran, K. K., Doshi, R., Fagbola, T., & Mahrishi, M. (2019). Cloud computing: master the concepts, architecture and applications with real-world examples and case studies. Bpb Publications.
[34] Amankwah, R., Asianoa, R., & Birago, B. Virtualization and Cloud Computing. International Journal of Computer Applications, 975, 8887.
[35] Kocaleva, M., Zlatanovska, B., Karamazova Gelova, E., & Zlatev, Z. (2024). Cloud computing and virtualization: can cloud computing exist separately from virtualization?.
[36] Furht, B., & Escalante, A. (2010). Handbook of cloud computing (Vol. 3). New York: springer.
[37] Zerwas, J., Györgyi, C., Blenk, A., Schmid, S., & Avin, C. (2023). Duo: A high-throughput reconfigurable datacenter network using local routing and control. Proceedings of the ACM on Measurement and Analysis of Computing Systems, 7(1), 1-25.
[38] Haddadou, K., & Pujolle, G. (2024). Cloud et Edge Networking. ISTE Group.
[39] Dutt, D. G. (2019). Cloud native data center networking: architecture, protocols, and tools. O'Reilly Media.
[40] Dab, B., Fajjari, I., Belabed, D., & Aitsaadi, N. (2021). Architectures of Data Center Networks: Overview. Management of Data Center Networks, 1-27.
[41] Alkhatib, A., Shaheen, A., & Albustanji, R. N. (2024). A Comparative Analysis of Cloud Computing Services: AWS, Azure, and GCP. International Journal of Computing and Digital Systems, 16(1), 1-23.
[42] Borra, P. (2024). Comparison and Analysis of Leading Cloud Service Providers (AWS, Azure and GCP). International Journal of Advanced Research in Engineering and Technology (IJARET), 15(3).
[43] Buyya, R., Broberg, J., & Goscinski, A. M. (Eds.). (2010). Cloud computing: Principles and paradigms. John Wiley & Sons.
[44] Nicolazzo, S., Nocera, A., & Pedrycz, W. (2024). Service Level Agreements and Security SLA: A Comprehensive Survey. arXiv preprint arXiv:2405.00009.
[45] Ludwig, H. (2003, December). Web services QoS: external SLAs and internal policies or: how do we deliver what we promise?. In Fourth International Conference on Web Information Systems Engineering Workshops, 2003. Proceedings. (pp. 115-120). IEEE.
[46] D. Chaudhary and B. Kumar, "Cost optimized Hybrid Genetic-Gravitational Search Algorithm for load scheduling in Cloud Computing," Appl. Soft Compute. J., vol. 83, 2019.
[47] S. Mathew and J. Varia, "Overview of amazon web services," Amazon Whitepapers, vol. 105, no.1, p. 22, 2014.
[48] Ben-Yehuda, O. A., Ben-Yehuda, M., Schuster, A., & Tsafrir, D. The Rise of RaaS: The Resource-as-a-Service Cloud In the RaaS cloud, virtual machines trade in fine-grain resources on the fly.
[49] Mishra, P. (2023). Advanced AWS Services. In Cloud Computing with AWS: Everything You Need to Know to be an AWS Cloud Practitioner (pp. 247-277). Berkeley, CA: Apress.
[50] Kadaskar, H. R., & Kamthe, V. R. (2024). An overview of AWS. International Journal of Scientific Research in Modern Science and Technology, 3(7), 22-30.

[51] Patibandla, K. R. (2024). Design and Create VPC in AWS. Journal of Artificial Intelligence General science (JAIGS) ISSN: 3006-4023, 1(1), 273-282.
[52] Talluri, S., & Makani, S. T. (2023). Managing Identity and Access Management (IAM) in Amazon Web Services (AWS). Journal of Artificial Intelligence & Cloud Computing. SRC/JAICC-159. DOI: doi. org/10.47363/JAICC/2023 (2), 147, 2-5.
[53] Amazon Web Services. (2024, June 27). AWS Well-Architected Framework: Cost Optimization Pillar. https://docs.aws.amazon.com/wellarchitected/latest/cost-optimization-pillar/welcome.html.
[54] Amazon Web Services. (2024). AWS account management: Reference guide. https://docs.aws.amazon.com/accounts/latest/reference/manage-acct-regions.html.
[55] Hunter, T., & Porter, S. (2018). Google Cloud Platform for developers: build highly scalable cloud solutions with the power of Google Cloud Platform. Packt Publishing Ltd.
[56] Borra, P. (2024). A Survey of Google Cloud Platform (GCP): Features, Services, and Applications. International Journal of Advanced Research in Science, Communication and Technology (IJARSCT), 4(3), 191-199.
[57] Haq, M. N. U. (2023). CLOUD SERVICE PROVIDERS AND THE ECOSYSTEM (Doctoral dissertation, School of Science and Technology, Glocal University).
[58] Deshpande, A., Kumar, M., & Chaudhari, V. (2020). Hands-On Artificial Intelligence on Google Cloud Platform: Build Intelligent Applications Powered by TensorFlow, Cloud AutoML, BigQuery, and Dialogflow. Packt Publishing Ltd.
[59] Google Cloud. (2024). Compute Engine: Documentation, guides - Regions and zones. https://cloud.google.com/compute/docs/regions-zones.
[60] BlueXP by NetApp. (2021). Google Cloud pricing: The complete guide. https://bluexp.netapp.com/blog/gcp-cvo-blg-google-cloud-pricing-the-complete-guide.
[61] Andersson, J. C. (2023). Learning Microsoft Azure. " O'Reilly Media, Inc.".
[62] Boneder, S. (2023). Evaluation and comparison of the security offerings of the big three cloud service providers Amazon Web Services, Microsoft Azure and Google Cloud Platform (Doctoral dissertation, Technische Hochschule Ingolstadt).
[63] Falck, O., & Wass, L. (2024). Azure App Service Plan Optimization: Cloud Resource optimization.
[64] Ascensão, P., Neto, L. F., Velasquez, K., & Abreu, D. P. (2024, June). Assessing Kubernetes Distributions: A Comparative Study. In 2024 IEEE 22nd Mediterranean Electrotechnical Conference (MELECON) (pp. 832-837). IEEE.
[65] Soueidi, C. (2015). Microsoft Azure Storage Essentials. Packt Publishing Ltd.
[66] Ramesh, R. S. (2024). Scalable Systems and Software Architectures for High-Performance Computing on cloud platforms. arXiv preprint arXiv:2408.10281.
[67] Mäenpää, J. (2009, April). Cloud computing with the Azure platform. In TKK T-110.5190 Seminar on Internet Working.
[68] Borra, P. (2024). Microsoft Azure Networking: Empowering Cloud Connectivity and Security. International Journal of Advanced Research in Science, Communication and Technology (IJARSCT) Volume, 4.
[69] Borra, P. (2024). Advancing Data Science and AI with Azure Machine Learning: A Comprehensive Review. International Journal of Research Publication and Reviews, 5(6), 1825-1831.
[70] Borra, P. (2024). Impact and Innovations of Azure IoT: Current Applications, Services, and Future Directions. International Journal of Recent Technology and Engineering (IJRTE) ISSN, 2277-3878.
[71] Sabharwal, N., Barua, S., Anand, N., & Aggarwal, P. (2019). Developing Cognitive Bots Using the IBM Watson Engine: Practical, Hands-on Guide to Developing Complex Cognitive Bots Using the IBM Watson Platform. Apress.
[72] Vehniä, V. J. (2020). Implementing Azure Active Directory Integration with an Existing Cloud Service.
[73] Microsoft. (2024, March 20). Azure geographies: Availability zones overview. https://learn.microsoft.com/en-us/azure/reliability/availability-zones-overview?tabs=azure-cli.
[74] Microsoft. (2024). Microsoft Azure official site: Pricing. https://azure.microsoft.com/en-us/pricing.
[75] Shukla, S., Hassan, M. F., Tran, D. C., Akbar, R., Paputungan, I. V., & Khan, M. K. (2023). Improving latency in Internet-of-Things and cloud computing for real-time data transmission: a systematic literature review (SLR). Cluster Computing, 1-24.
[76] Marinescu, D. C. (2022). Cloud computing: theory and practice. Morgan Kaufmann.

[77] Dang, T. K., Mohan, N., Corneo, L., Zavodovski, A., Ott, J., & Kangasharju, J. (2021, November). Cloudy with a chance of short RTTs: analyzing cloud connectivity in the internet. In Proceedings of the 21st ACM Internet Measurement Conference (pp. 62-79).
[78] Selimi, M., Freitag, F., Cerdà-Alabern, L., & Veiga, L. (2016). Performance evaluation of a distributed storage service in community network clouds. Concurrency and Computation: Practice and Experience, 28(11), 3131-3148.
[79] Yadav, R. K., Chattopadhyay, S., Jaidka, P., & Upadhyay, P. (2024, March). Performance Analysis of Cloud-Assisted Resource Allocation Algorithms in 6G Networks. In 2024 2nd International Conference on Disruptive Technologies (ICDT) (pp. 1038-1043). IEEE.
[80] Arslan, S., Li, Y., Kumar, G., & Dukkipati, N. (2023). Bolt:{Sub-RTT} Congestion Control for {Ultra-Low} Latency. In 20th USENIX Symposium on Networked Systems Design and Implementation (NSDI 23) (pp. 219-236).
[81] Buyya, R., Yeo, C. S., & Venugopal, S. (2008, September). Market-oriented cloud computing: Vision, hype, and reality for delivering it services as computing utilities. In 2008 10th IEEE international conference on high performance computing and communications (pp. 5-13). Ieee.
[82] Padmanabhan, V. N., & Subramanian, L. (2001, August). An investigation of geographic mapping techniques for Internet hosts. In Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications (pp. 173-185).
[83] Rak, J. Resilient Routing in Communication Networks: A Systems Perspective. Springer Nature.
[84] Haitjema, M. A. (2013). Delivering Consistent Network Performance in Multi-tenant Data Centers. Washington University in St. Louis.
[85] McCabe, J. D. (2010). Network analysis, architecture, and design. Elsevier.
[86] Bathini, R., & Vurukonda, N. (2024). A survey to build framework for optimize and secure migration and transmission of cloud data. Bulletin of Electrical Engineering and Informatics, 13(2), 812-820.
[87] Mark, J., & Bommu, R. (2024). Tackling Environmental Concerns: Mitigating the Carbon Footprint of Data Transmission in Cloud Computing. Unique Endeavor in Business & Social Sciences, 3(1), 99-112.
[88] Jin, H., Ibrahim, S., Bell, T., Gao, W., Huang, D., & Wu, S. (2010). Cloud types and services. Handbook of cloud computing, 335-355.
[89] Hopgood, A. A. (2021). Intelligent systems for engineers and scientists: a practical guide to artificial intelligence. CRC press.
[90] Le Thi, H. A., Le, H. M., Dinh, T. P., & Nguyen, N. T. (2015). Advanced computational methods for knowledge engineering. Cham: Springer International Publishing.
[91] Chandrasekaran, E., Anandan, R., Suseendran, G., Balamurugan, S., & Hachimi, H. (2021). Fuzzy Intelligent Systems: Methodologies, Techniques, and Applications. Scrivener Publishing. https://www.scrivenerpublishing.com. https://doi.org/10.1002/9781119763437
[92] Shehu, A., & Maraj, A. (2012). Fuzzy logic approach for QoS routing analysis. Fuzzy Logic-Algorithms, Techniques and Implementations, 149-172.
[93] Nithya, S., Maithili, K., Kumar, T. S., Nethani, S., Sharath, M. N., Gupta, K. G., & Bhuvaneswari, G. (2024). A fuzzy logic and cross-layered optimization for effective congestion control in wireless sensor networks to improve efficiency and performance. In MATEC Web of Conferences (Vol. 392, p. 01145). EDP Sciences.
[94] Huang, H., Wang, Y., Cai, Y., & Wang, H. (2024). A novel approach for energy consumption management in cloud centers based on adaptive fuzzy neural systems. Cluster Computing, 1-24.
[95] Goel, u., wittie, m. P., claffy, k. C., & le, a. (2015). Survey of end-to-end mobile network measurement testbeds, tools, and services. Ieee communications surveys & tutorials, 18(1), 105-123.
[96] Mirkovic, d., armitage, g., & branch, p. (2018). A survey of round-trip time prediction systems. Ieee communications surveys & tutorials, 20(3), 1758-1776.
[97] GeeksforGeeks. (2023, April 13). What is RTT (Round-Trip Time)? GeeksforGeeks. https://*www*.geeksforgeeks.org/what-is-rttround-trip-time.
[98] Klir, g. J., & yuan, b. (1996). Fuzzy sets and fuzzy logic: theory and applications. Possibility theory versus probab. Theory, 32(2), 207-208.https://doi.10.5860/choice.33-2786.
[99] Pedrycz, w. (1994). Why triangular membership functions? Fuzzy sets and systems, 64(1), 21-30. Https://doi.org/10.1016/0165-0114(94)90003-5
[100] Baliyan, N., & Kumar, S. (2013, October). Quality assessment of software as a service on cloud using fuzzy logic. In 2013 IEEE International Conference on Cloud Computing in Emerging Markets (CCEM) (pp. 1-6). IEEE. https://doi: 10.1109/CCEM.2013.6684439

[101] Alhamad, M., Dillon, T., & Chang, E. (2011). A trust-evaluation metric for cloud applications. International Journal of Machine Learning and Computing, 1(4), 416. https://doi: 10.7763/IJMLC. 2011.V1.62
[102] Xiaoyong, Y., Ying, L., Tong, J., Tiancheng, L., & Zhonghai, W. (2015, July). An analysis on availability commitment and penalty in cloud sla. In 2015 IEEE 39th Annual Computer Software and Applications Conference (Vol. 2, pp. 914-919). IEEE. https://doi:10.1109/COMPSAC.2015.39
[103] Kihuya, W. B., Otieno, C., & Rimiru, R. Analysis of Computer Network Quality of Experience Using Fuzzy Logic Model: A Survey. https://doi:10.9790/1813-0804028596
[104] Al Moteri, M. A. (2017). Decision Support for Shared Responsibility of Cloud Security Metrics.
[105] Abery, B., Bonner, M., Fossum, P., Koch, T., Montie, J., Nordness, K., ... & Vandercook, T. (1998). The Shared Responsibility Framework of Social Interaction for Collective Investment: Introducing a Model To Enhance School Improvement.
[106] Qiqing, F., Xiaoming, P., Qinghua, L., & Yahui, H. (2009, May). A global qos optimizing web services selection algorithm based on moaco for dynamic web service composition. In 2009 International forum on information technology and applications (Vol. 1, pp. 37-42). IEEE. https://doi: 10.1109/IFITA.2009.91
[107] Tran, V. X., & Tsuji, H. (2008, October). QoS based ranking for web services: Fuzzy approaches. In 2008 4th international conference on next generation web services practices (pp. 77-82). IEEE. https://doi: 10.1109/NWeSP.2008.41
[108] Patel, P., Ranabahu, A. H., & Sheth, A. P. (2009). Service level agreement in cloud computing.
[109] Alhamad, M., Dillon, T., & Chang, E. (2010, April). Conceptual SLA framework for cloud computing. In 4th IEEE international conference on digital ecosystems and technologies (pp. 606-610). IEEE. https://doi:10.1109/DEST.2010.5610586
[110] Qiu, M. M., Zhou, Y., & Wang, C. (2013, June). Systematic analysis of public cloud service level agreements and related business values. In 2013 IEEE International Conference on Services Computing (pp. 729-736). IEEE. https://doi: 10.1109/SCC.2013.24
[111] Brunnström, K., Beker, S. A., De Moor, K., Dooms, A., Egger, S., Garcia, M. N., ... & Zgank, A. (2013). Qualinet white paper on definitions of quality of experience.
[112] Baset, S. A. (2012). Cloud SLAs: present and future. ACM SIGOPS Operating Systems Review, 46(2), 57-66. https://doi.org/10.1145/2331576.2331586
[113] Godhrawala, H., & Sridaran, R. (2023). Apriori Algorithm Based Approach for Improving QoS and SLA Guarantee in IaaS Clouds Using Pattern-Based Service-Oriented Architecture. SN Computer Science, 4(5), 700.
[114] Akbari-Moghanjoughi, A., Amazonas, J. R. D. A., Santos-Boada, G., & Solé-Pareta, J. (2023). Service level agreements for communication networks: A survey. arXiv preprint arXiv:2309.07272.
[115] Saqib, M., Elbiaze, H., & Glitho, R. (2024). Adaptive In-Network Traffic Classifier: Bridging the Gap for Improved QoS by Minimizing Misclassification. IEEE Open Journal of the Communications Society.
[116] Bauer, E., & Adams, R. (2012). Reliability and availability of cloud computing. John Wiley & Sons.
[117] Maciel, P. R. M. (2023). Performance, reliability, and availability evaluation of computational systems, volume I: performance and background. Chapman and Hall/CRC.
[118] Nabi, M., Toeroe, M., & Khendek, F. (2016). Availability in the cloud: State of the art. Journal of Network and Computer Applications, 60, 54-67. https://doi.org/10.1016/j.jnca.2015.11.014
[119] Toeroe, M., & Tam, F. (Eds.). (2012). Service availability: principles and practice. John Wiley & Sons.
[120] Hauer, T., Hoffmann, P., Lunney, J., Ardelean, D., & Diwan, A. (2020). Meaningful availability. In 17th USENIX Symposium on Networked Systems Design and Implementation (NSDI 20) (pp. 545-557).
[121] Hanczewski, S., Stasiak, M., & Weissenberg, M. (2024). High-Accuracy Analytical Model for Heterogeneous Cloud Systems with Limited Availability of Physical Machine Resources Based on Markov Chain. Electronics, 13(11), 2161.
[122] Aceto, G., Botta, A., Marchetta, P., Persico, V., & Pescapé, A. (2018). A comprehensive survey on internet outages. Journal of Network and Computer Applications, 113, 36-63.
[123] Miracle, N. O. (2024). The role of network monitoring and analysis in ensuring optimal network performance. International Research Journal of Modernization in Engineering Technology and Science. https://doi. org/10.56726/irjmets59269.
[124] Strauss, J., & Kaashoek, M. F. Estimating Bulk Transfer Capacity.

[125] Ramos, J., del Río, P. S., Aracil, J., & de Vergara, J. L. (2011). On the effect of concurrent applications in bandwidth measurement speedometers. Computer Networks, 55(6), 1435-1453.
[126] Barney, D. (2024, June 10). 12 network metrics and KPIs you should probably care about. Network Computing. Retrieved from https://www.networkcomputing.com.
[127] Abts, D., & Kim, J. (2022). High performance datacenter networks: Architectures, algorithms, and opportunities. Springer Nature.
[128] Schmidt, F. (2015). Heuristic Header Error Recovery for Corrupted Network Packets. Shaker Verlag.
[129] Kim, D., & Cho, I. H. (1998). An Optimal COG Defuzzification Method for A Fuzzy Logic Controller. In Soft Computing in Engineering Design and Manufacturing (pp. 401-409). Springer London.https://doi. DOI:10.1007/978-1-4471-0427-8_44
[130] Regaya, C. B., Farhani, F., Hamdi, H., Zaafouri, A., Chaari, A. "Robust ANFIS vector control of induction motor drive for high-performance speed control supplied by a photovoltaic generator," WSEAS Transactions on Systems and Control, 15(37), pp. 356–365, 2020. https://doi.org/10.37394/23203.2020.15.37
[131] Tahmasebi, M., Gohari, M., Emami, A. "An autonomous pesticide sprayer robot with a color-based vision system," International Journal of Robotics and Control Systems, 2(1), pp. 115–123, 2022. https://doi.org/10.31763/ijrcs.v2i1.480
[132] Chakchouk, W., Ben Regaya, C., Zaafouri, A., Sellami, A. "Fuzzy supervisor approach design-based switching controller for pumping station: Experimental validation," Mathematical Problems in Engineering, 2017(1), Article ID 3597346, 2017. https://doi.org/10.1155/2017/3597346
[133] Regaya, C. B., Farhani, F., Zaafouri, A., Chaari, A. "High-performance control of IM using MRAS-fuzzy logic observer," International Journal of Tomography and Simulation, 30(2), pp. 40–52, 2017. ISSN 0973-7294.
[134] Ben Regaya, C., Zaafouri, A., Chaari, A. "Electric drive control with rotor resistance and rotor speed observers based on fuzzy logic," Mathematical Problems in Engineering, 2014(1), Article ID 207826, 2014. https://doi.org/10.1155/2014/207826
[135] Sharma, R., Gaur, P., Mittal, A. P. "Design of two-layered fractional order fuzzy logic controllers applied to robotic manipulator with variable payload," Applied Soft Computing, 47, pp. 565–576, 2016. https://doi.org/10.1016/j.asoc.2016.05.043
[136] Berkachy, R. "Fuzzy Rule-Based Systems," In: The Signed Distance Measure in Fuzzy Statistical Analysis, Fuzzy Management Methods, Springer, Cham, 2021, pp. 35–45. ISBN 978-3-030-76915-4 https://doi.org/10.1007/978-3-030-76916-1_3
[137] Zuliana, E., Abadi, A. M. "Sugeno fuzzy inference method and MATLAB application program for simulation of student performance evaluation in the elementary mathematics learning process," International Journal of Advanced Trends in Computer Science and Engineering, 9, pp. 4223–4228, 2020. ISSN 2278-3091. https://doi.org/10.30534/ijatcse/2020/08942020
[138] Petrović, V. M. "Artificial intelligence and virtual worlds–toward human-level AI agents," IEEE Access, 6, pp. 39976–39988, 2018. https://doi.org/10.1109/ACCESS.2018.2855970
[139] Voskoglou, M. "Fuzzy logic: History, methodology and applications to education," Sumerianz Journal of Education, Linguistics and Literature, 1(1), pp. 10–18, 2018. ISSN (p): 2617-1732.
[140] Mounika, G., Rajyalakshmi, K., Rajkumar, G. V. S., Sravani, D. "Prediction and optimization of process parameters using design of experiments and fuzzy logic," International Journal on Interactive Design and Manufacturing (IJIDeM), 18(4), pp. 2333–2343, 2024. https://doi.org/10.1007/s12008-023-01446-x
[141] Valdés, L. V. "Methods and elements of graph theory and fuzzy logic for communication network management," PhD, Universidad de Málaga, 2022.
[142] Lagunes, M. L., Castillo, O., Soria, J. "Optimization of membership function parameters for fuzzy controllers of an autonomous mobile robot using the firefly algorithm," In: Castillo, O., Melin, P., Kacprzyk, J. (eds.) Fuzzy Logic Augmentation of Neural and Optimization Algorithms: Theoretical Aspects and Real Applications, Springer, Cham, 2018, pp. 199–206. ISBN 978-3-319-71008-2 https://doi.org/10.1007/978-3-319-71008-2_16
[143] Nadeem, A., Rizvi, A. A., Noor, M. Y. "Applying a higher number of output membership functions to enhance the precision of a fuzzy system," IEEE Transactions on Emerging Topics in Computational Intelligence, 1, pp. 1–12, 2024. https://doi.org/10.1109/TETCI.2024.3425309
[144] Ying, H. "Fuzzy Control and Modeling: Analytical Foundations and Applications," Wiley-IEEE Press, 2000. ISBN 0780334973. https://doi.org/10.1109/9780470544730

[145] Zadeh, L. A. "Fuzzy sets," Information and Control, 8(3), pp. 338–353, 1965. https://doi.org/10.1016/S0019-9958(65)90241-X
[146] Dubois, D., Prade, H. Fuzzy Sets and Systems: Theory and Applications, Academic Press, 1980. ISBN 0-12-222750-6.
[147] Gupta, K., Kumar, P., Upadhyaya, S., Poriye, M., Aggarwal, S. "Fuzzy logic and machine learning integration: Enhancing healthcare decision-making," International Journal of Computer Information Systems and Industrial Management Applications, 16(3), pp. 20–20, 2024.
[148] Zheng, Y., Xu, Z., Wu, T., et al. "A systematic survey of fuzzy deep learning for uncertain medical data," Artificial Intelligence Review, 57, pp. 230, 2024. https://doi.org/10.1007/s10462-024-10871-7
[149] Herrera, F., Martínez, L. "A 2-tuple fuzzy linguistic representation model for computing with words," IEEE Transactions on Fuzzy Systems, 8(6), pp. 746–752, 2000. https://doi.org/10.1109/91.890332
[150] Marín Díaz, G., Galdón Salvador, J. L. "Group decision-making model based on 2-tuple fuzzy linguistic model and AHP applied to measuring digital maturity level of organizations," Systems, 11(7), p. 341, 2023. https://doi.org/10.3390/systems11070341
[151] Wang, L. X., Mendel, J. M. "Fuzzy basis functions, universal approximation, and orthogonal least-squares learning," IEEE Transactions on Neural Networks, 3(5), pp. 807–814, 1992. https://doi.org/10.1109/72.159070
[152] Al-qaysi, Z. T., Albahri, A. S., Ahmed, M. A., Salih, M. M. "Dynamic decision-making framework for benchmarking brain–computer interface applications: a fuzzy-weighted zero-inconsistency method for consistent weights and VIKOR for stable rank," Neural Computing and Applications, 36(17), pp. 10355–10378, 2024. https://doi.org/10.1007/s00521-024-09605-1
[153] Perera, L. P., Carvalho, J. P., Soares, C. G. "Solutions to the failures and limitations of Mamdani fuzzy inference in ship navigation," IEEE Transactions on Vehicular Technology, 63(4), pp. 1539–1554, 2013. https://doi.org/10.1109/TVT.2013.2288306
[154] Raju, M. R., Mothku, S. K. "Delay and energy aware task scheduling mechanism for fog-enabled IoT applications: A reinforcement learning approach," Computer Networks, 224, Article ID 109603, 2023. https://doi.org/10.1016/j.comnet.2023.109603
[155] Pedrycz, W. "Evolvable fuzzy systems: Some insights and challenges," Evolving Systems, 1, pp. 73–82, 2010. https://doi.org/10.1007/s12530-010-9002-1
[156] Kovacic, Z., Bogdan, S. "Fuzzy controller design: theory and applications" [e-book], CRC Press, 2018. ISBN 9781315221144. https://doi.org/10.1201/9781420026504
[157] Dong, T., Li, H., Zhang, Z. "The using effect of fuzzy analytic hierarchy process in project engineering risk management," Neural Computing and Applications, pp. 1–11, 2023. https://doi.org/10.1007/s00521-023-09046-2
[158] Seddik, H. M., Rachid, C. "Fuzzy approach and possibility to solve uncertainty weaknesses in conventional quantitative risk assessment," Soft Computing, 27(10), pp. 6109–6133, 2023. https://doi.org/10.1007/s00500-023-07960-0
[159] Sahoo, S. K., Goswami, S. S. "A comprehensive review of multiple criteria decision-making (MCDM) methods: advancements, applications, and future directions," Decision Making Advances, 1(1), pp. 25–48, 2023. https://doi.org/10.31181/dma1120237
[160] He, S. F., Pan, X. H., Wang, Y. M., Zamora, D. G., Martínez, L. "A novel multi-criteria decision-making framework based on evidential reasoning dealing with missing information from online reviews," Information Fusion, 106, Article ID 102264, 2024. https://doi.org/10.1016/j.inffus.2024.102264
[161] Gen, M., Lin, L. "Genetic algorithms and their applications," In: Pham, H. (ed.) Springer Handbook of Engineering Statistics, Springer Handbooks, Springer, London, 2023, pp. 635–674. ISBN 978-1-4471-7502-5. https://doi.org/10.1007/978-1-4471-7503-2_33
[162] Guerrero Granados, B., Quintero M, C. G., Núñez, C. V. "Improved genetic algorithm approach for coordinating decision-making in technological disaster management," Neural Computing and Applications, 36(9), pp. 4503–4521, 2024. https://doi.org/10.1007/s00521-023-09218-0
[163] Zadeh, L. A. "Fuzzy logic = computing with words," IEEE Transactions on Fuzzy Systems, 4(2), pp. 103–111, 1996. https://doi.org/10.1109/91.493904
[164] Mitiku, T., Manshahia, M. S. "Neuro fuzzy inference approach: A survey," International Journal of Scientific Research in Science, Engineering and Technology, 4(7), pp. 505–519, 2018. Print ISSN: 2395-1990, Online ISSN: 2394-4099.

[165] Zadeh, L. A., Klir, G. J., Yuan, B. "Fuzzy sets, fuzzy logic, and fuzzy systems: Selected papers," World Scientific, 1996. ISBN: 978-981-02-2421-9. https://doi.org/10.1142/2895
[166] Hasan, M. H., Jaafar, J., Hassan, M. F. "Fuzzy C-Means and two clusters' centers method for generating interval type-2 membership function," In: 2016 3rd International Conference on Computer and Information Sciences (ICCOINS), Kuala Lumpur, Malaysia, 2016, pp. 627–632. https://doi.org/10.1109/ICCOINS.2016.7783288
[167] Takagi, T., Sugeno, M. "Fuzzy identification of systems and its applications to modeling and control," IEEE Transactions on Systems, Man, and Cybernetics, 15(1), pp. 116–132, 1985. https://doi.org/10.1109/TSMC.1985.6313399
[168] Abramowitz, M., Stegun, I. A. (eds.) Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables, US Government Printing Office, 1988. https://doi.org/10.1119/1.15378
[169] Oluborode, K. O. "Adaptive neuro-fuzzy controller for double lane traffic intersections," PhD, Federal University of Technology Akure, 2021. https://doi.org/10.30534/ijatcse/2020/330942020
[98] Klir, G., Yuan, B. "Fuzzy Sets and Fuzzy Logic," Prentice Hall, New Jersey, 1995. ISBN 0-13-101171-5.
[170] Simon, M. K. "Probability distributions involving Gaussian random variables: A handbook for engineers and scientists," Kluwer Academic Publishers, Boston-Dordrecht-London, 2002. ISBN 978-0-387-34657-1.
[171] Miller, S. "Probability and random processes: With applications to signal processing and communications," Academic Press, 2012. ISBN 978-0-12-386981-4.
[172] Elgendi, M. "PPG signal analysis: An introduction using MATLAB®" [e-book], CRC Press, 2020. ISBN 9780429449581. https://doi.org/10.1201/9780429449581
[173] Ruparelia, N. B. (2023). Cloud computing. Mit Press.
[174] Rao, M. N. (2015). Cloud computing. PHI Learning Pvt. Ltd.
[175] Gong, C., Liu, J., Zhang, Q., Chen, H., & Gong, Z. (2010, September). The characteristics of cloud computing. In 2010 39th International Conference on Parallel Processing Workshops (pp. 275-279). IEEE.
[176] Bharti, P., Ranjan, R., & Prasad, B. (2021). Broker-based optimization of SLA negotiations in cloud computing. Multiagent and Grid Systems, 17(2), 179-195.
[177] Nagarajan, R., Vinothiyalakshmi, P., & Thirunavukarasu, R. (2023). An Intelligent Cloud Broker with Service Ranking Algorithm for Validation and Verification of Cloud Services in Multi-cloud Environment.
[178] Dilli, R., Argou, A., Pilla, M., Pernas, A. M., Reiser, R., & Yamin, A. (2018, April). Fuzzy logic and MCDA in IoT resources classification. In Proceedings of the 33rd Annual ACM Symposium on Applied Computing (pp. 761-766).
[179] Ghasemi, A., Toroghi Haghighat, A., & Keshavarzi, A. (2023). Enhanced multi-objective virtual machine replacement in cloud data centers: combinations of fuzzy logic with reinforcement learning and biogeography-based optimization algorithms. Cluster Computing, 26(6), 3855-3868.
[180] Mongia, V., & Sharma, A. (2021). Performance and resource-aware virtual machine selection using fuzzy in cloud environment. In Progress in Advanced Computing and Intelligent Engineering: Proceedings of ICACIE 2020 (pp. 413-426). Springer Singapore.
[181] Singh, H., Tyagi, S., & Kumar, P. (2021). Comparative analysis of various simulation tools used in a cloud environment for task-resource mapping. In Proceedings of the International Conference on Paradigms of Computing, Communication and Data Sciences: PCCDS 2020 (pp. 419-430). Springer Singapore.
[182] Mohanty, S., Patra, S., Sarkar, S., Dube, P., & Pattnaik, P. K. (2024, February). Load Balancing in Cloud Environment to Minimize Average Response Time. In 2024 International Conference on Emerging Systems and Intelligent Computing (ESIC) (pp. 187-192). IEEE.
[183] Garg, R., Sharma, R. K., Dalip, D., Singh, T., Malik, A., & Kumpsuprom, S. (2023). Optimization of Cloud Services Performance using Static and Dynamic Load Balancing Algorithms.
[184] Zhao, W., Peng, Y., Xie, F., & Dai, Z., Modeling and simulation of cloud computing: A review. In 2012 IEEE Asia Pacific cloud computing congress (APCloudCC) (pp. 20-24). IEEE. (2012)
[185] Chauhan, S. S., Pilli, E. S., Joshi, R. C., Singh, G., & Govil, M. C., Brokering in interconnected cloud computing environments: A survey. Journal of Parallel and Distributed Computing, 133, 193-209. (2019)
[186] Ahmad, S. G., Iqbal, T., Munir, E. U., & Ramzan, N., Cost optimization in cloud environment based on task deadline. Journal of Cloud Computing, 12(1), 9. (2023)

[187] Yao, J., Yang, M., Deng, T., & Guan, H., The Cloud Service Broker in Multicloud Demand Response. IEEE Cloud Computing, 5(6), 80-91. (2018)
[188] Cinar, B., The Role of Cloud Service Brokers: Enhancing Security and Compliance in Multi-cloud Environments. Journal of Engineering Research and Reports, 25(10), 1-11. (2023)
[189] Petcu, D., Portability and interoperability between clouds: challenges and case study. In Towards a Service-Based Internet: 4th European Conference, ServiceWave 2011, Poznan, Poland, October 26-28, 2011. Proceedings 4 (pp. 62-74). Springer Berlin Heidelberg. (2011)
[190] Chafai, Z., Nacer, H., Lekadir, O., Gharbi, N., & Ouchaou, L., A performance evaluation model for users' satisfaction in federated clouds. Cluster Computing, 1-22. (2024)
[191] Calheiros, R. N., Toosi, A. N., Vecchiola, C., & Buyya, R., A coordinator for scaling elastic applications across multiple clouds. Future Generation Computer Systems, 28(8), 1350-1362. (2012)
[192] El Karadawy, A. I., Mawgoud, A. A., & Rady, H. M., An empirical analysis on load balancing and service broker techniques using cloud analyst simulator. In 2020 international conference on innovative trends in communication and computer engineering (ITCE) (pp. 27-32). IEEE. (2020)
[193] Achhra, S. N. M., Shah, R., Tamrakar, A., & Joshi, P. K., Prof Sowmiya Raksha,"Analysis OF Service Broker And Load Balancing In Cloud Computing,". International Journal Of Current Engineering And Scientific Research (IJCESR), 2(4), 92-98. (2015)
[194] Wittig, A., & Wittig, M. (2023). Amazon Web Services in Action: An in-depth guide to AWS. Simon and Schuster.
[195] Manvi, S., & Shyam, G. (2021). Cloud computing: Concepts and technologies. CRC Press.
[196] Ahmed, A., & Sabyasachi, A. S., Cloud computing simulators: A detailed survey and future direction. In 2014 IEEE international advance computing conference (IACC) (pp. 866-872). IEEE. (2014).
[197] Srujana, R., Roopa, Y. M., & Mohan, M. D. S. K., Sorted round robin algorithm. In 2019 3rd International Conference on Trends in Electronics and Informatics (ICOEI) (pp. 968-971). IEEE. (2019)
[198] Youm, D. H., & Yadav, R., Load balancing strategy using round robin algorithm. Asia-pacific Journal of Convergent Research Interchange, 2(3), 1-10. (2016)
[199] Patel, H., & Patel, R., Cloud analyst: an insight of service broker policy. International Journal of Advanced Research in Computer and Communication Engineering, 4(1), 122-127. (2015)
[200] Gaur, A.; Garg, K. Survey paper on cloud computing with load balancing policy. Int. J. Eng. Res. 2015, 2, 7.
[201] Arseniev, D. G., Overmeyer, L., Kälviäinen, H., & Katalinić, B. (Eds.), Cyber-Physical Systems and Control (Vol. 95). Springer Nature. (2019)
[202] Puri, T., Challa, R. K., & Sehgal, N. K., Energy-efficient delay-aware preemptive variable-length time slot allocation scheme for WBASN (edpvt). In Proceedings of 2nd International Conference on Communication, Computing and Networking: ICCCN 2018, NITTTR Chandigarh, India (pp. 183-194). Springer Singapore. (2019)
[203] Benlalia, Z., Beni-hssane, A., Abouelmehdi, K., & Ezati, A. A new service broker algorithm optimizing the cost and response time for cloud computing. Procedia Computer Science, 151, 992-997. (2019)
[204] Radi, M., Efficient service broker policy for large-scale cloud environments. arXiv preprint arXiv:1503.03460. (2015)
[205] Mesbahi, M. R., Hashemi, M., & Rahmani, A. M., Performance evaluation and analysis of load balancing algorithms in cloud computing environments. In 2016 Second International Conference on Web Research (ICWR) (pp. 145-151). IEEE. (2016)
[206] Khalil, K. M., Abdel-Aziz, M., Nazmy, T. T., & Salem, A. B. M., Cloud simulators–an evaluation study. International Journal Information Models and Analyses, 6(1). (2017)
[207] Nayak, S., & Patel, P., Analytical Study for Throttled and Proposed Throttled Algorithm of Load Balancing in Cloud Computing using Cloud Analyst. International Journal of Science Technology & Engineering, 1(12), 90-100. (2015)
[208] Bahwaireth, K., Tawalbeh, L. A., Benkhelifa, E., Jararweh, Y., & Tawalbeh, M. A. (2016). Experimental comparison of simulation tools for efficient cloud and mobile cloud computing applications. EURASIP Journal on Information Security, 1-14. (2016)
[209] Mondal, S., Faruk, F. B., Rajbongshi, D., Efaz, M. M. K., & Islam, M. M. (2023). GEECO: Green Data Centers for Energy Optimization and Carbon Footprint Reduction. Sustainability, 15(21), 15249.

[210] Liu, J., Yan, L., Yan, C., Qiu, Y., Jiang, C., Li, Y., ... & Cérin, C. (2023). Escope: An energy efficiency simulator for internet data centers. Energies, 16(7), 3187.
[211] Rimal, B. P., Choi, E., & Lumb, I. (2009). A taxonomy and survey of cloud computing systems. Network and Communication Technologies, 4(4), 1-10.
[212] Buyya, R., Yeo, C. S., Venugopal, S., Broberg, J., & Brandic, I. (2009). Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility. Future Generation Computer Systems, 25(6), 599-616.
[213] Zhang, Q., Cheng, L., & Boutaba, R. (2010). Cloud computing: state-of-the-art and research challenges. Journal of Internet Services and Applications, 1(1), 7-18.
[214] Biswas, A., Majumdar, S., Nandy, B., & El-Haraki, A. (2017). A hybrid auto-scaling technique for clouds processing applications with service level agreements. Journal of Cloud Computing, 6, 1-22.
[215] Hwang, K., Fox, G. C., & Dongarra, J. (2012). Distributed and cloud computing: From parallel processing to the internet of things. Morgan Kaufmann.
[216] Calheiros, R. N., Ranjan, R., De Rose, C. A. F., & Buyya, R. (2009). CloudSim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. Software: Practice and Experience, 41(1), 23-50.
[217] Beloglazov, A., Buyya, R., Lee, Y. C., & Zomaya, A. Y. (2011). A taxonomy and survey of energy-efficient data centers and cloud computing systems. Advances in Computers, 82, 47-111.
[218] Xiao, Z., Song, W., & Chen, Q. (2013). Dynamic resource allocation using virtual machines for cloud computing environment. IEEE Transactions on Parallel and Distributed Systems, 24(6), 1107-1117.
[219] Kansal, N. J., & Chana, I. (2012). Cloud load balancing techniques: A step towards green computing. International Journal of Computer Science Issues, 9(1), 238-246.
[220] Wided, A., Çelebi, N., & Fatima, B. (2023). Effective Cloudlet Scheduling Algorithm for Load Balancing in Cloud Computing Using Fuzzy Logic. In Privacy Preservation and Secured Data Storage in Cloud Computing (pp. 226-243). IGI Global.
[221] Sangaiah, A. K., Javadpour, A., Pinto, P., Rezaei, S., & Zhang, W. (2023). Enhanced resource allocation in distributed cloud using fuzzy meta-heuristics optimization. Computer Communications, 209, 14-25.
[222] Aljuhani, A., & Alhubaishy, A. (2023). Dynamic Cloud Resource Allocation: A Broker-Based Multi-Criteria Approach for Optimal Task Assignment. Applied Sciences, 14(1), 302.
[223] Adami, D., Gabbrielli, A., Giordano, S., Pagano, M., & Portaluri, G. (2015, December). A fuzzy logic approach for resources allocation in cloud data center. In 2015 IEEE Globecom Workshops (GC Wkshps) (pp. 1-6). IEEE.
[224] Zaidi, R. T. (2018). Virtual Machine Allocation Policy in Cloud Computing Environment using CloudSim. International Journal of Electrical & Computer Engineering (2088-8708), 8(1).
[225] Li, X., Pan, L., & Liu, S. (2023). A DRL-based online VM scheduler for cost optimization in cloud brokers. World Wide Web, 26(5), 2399-2425.
[226] Gong, Y., Huang, J., Liu, B., Xu, J., Wu, B., & Zhang, Y. (2024). Dynamic resource allocation for virtual machine migration optimization using machine learning. arXiv preprint arXiv:2403.13619.
[227] Belgacem, A., Mahmoudi, S., & Kihl, M. (2022). Intelligent multi-agent reinforcement learning model for resources allocation in cloud computing. Journal of King Saud University-Computer and Information Sciences, 34(6), 2391-2404.
[228] Afrin, M., Jin, J., Rahman, A., Rahman, A., Wan, J., & Hossain, E. (2021). Resource allocation and service provisioning in multi-agent cloud robotics: A comprehensive survey. IEEE Communications Surveys & Tutorials, 23(2), 842-870.
[229] Huang, J., Chen, X., & Wang, H. (2020). Edge computing-based VM allocation for latency-sensitive applications in cloud environments. IEEE Internet of Things Journal, 7(8), 7345-7357.
[230] Kang, J., Yu, S., & Yang, K. (2020). Energy-efficient resource allocation for cloud data centers using a hybrid heuristic algorithm. Journal of Supercomputing, 76(3), 1631-1649.
[231] Zhang, Q., Cheng, L., & Boutaba, R. (2020). Cloud computing: State-of-the-art and research challenges. Journal of Internet Services and Applications, 11(1), 1-23.
[232] Taheri, H., Abrishami, S., & Naghibzadeh, M. (2023). A cloud broker for executing deadline-constrained periodic scientific workflows. IEEE Transactions on Services Computing.

[233] Balachandar, S., & Chinnaiyan, R. (2023). Intelligent Broker Design for IoT Using a Multi-Cloud Environment. In Convergence of Deep Learning and Internet of Things: Computing and Technology (pp. 23-41). IGI Global.
[234] Ramakrishnan, S. (Ed.). (2017). Modern Fuzzy Control Systems and Its Applications. BoD–Books on Demand.
[235] Mateen, M., Hayat, S., Tehreem, T., & Akbar, M. A. (2020). A self-adaptive resource provisioning approach using fuzzy logic for cloud-based applications. International Journal of Computing and Digital Systems, 9(03).
[236] Shahid, M. A., Alam, M. M., & Su'ud, M. M. (2023). A systematic parameter analysis of cloud simulation tools in cloud computing environments. Applied Sciences, 13(15), 8785.
[237] Buyya, R., Vecchiola, C., & Selvi, S. T. (2013). Mastering cloud computing: foundations and applications programming. Newnes.
[238] Singh, A., & Kumar, R. (2020, January). Performance evaluation of load balancing algorithms using cloud analyst. In 2020 10th International Conference on Cloud Computing, Data Science & Engineering (Confluence) (pp. 156-162). IEEE.
[239] Velte, A.T.V.T.J., Elsenpeter, P.D.R.: Cloud Computing, (2010)
[240] Giust, F., Costa-Perez, X., Reznik, A.: Multi-access edge computing: An overview
of etsi mec isg. IEEE 5G Tech Focus 1(4), 4 (2017)
[241] Bonomi, F., Milito, R., Zhu, J., Addepalli, S.: Fog computing and its role in the internet of things. In: Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing, pp. 13–16 (2012)
[242] Zhu, Z., Li, X., Chu, Z.: Three major operating scenarios of 5g: embb, mmtc, urllc. Intell. Sens. Commun. Internet Everything 1, 15–76 (2022)
[243] Bellavista, P., Carella, G., Foschini, L., Magedanz, T., Schreiner, F., Campowsky,K.: Qos-aware elastic cloud brokering for ims infrastructures. In: 2012 IEEE Symposium on Computers and Communications (ISCC), pp. 000157–000160 (2012). IEEE
[244] D'Agostino, D., Galizia, A., Clematis, A., Mangini, M., Porro, I., Quarati, A.: A qos-aware broker for hybrid clouds. Computing 95, 89–109 (2013)
[245] Devgan, M., Dhindsa, K.S.: Qos and cost aware service brokering using pattern-based service selection in cloud computing. International Journal of Soft Computing and Engineering 3, 441–446 (2014)
[246] Anastasi, G.F., Carlini, E., Coppola, M., Dazzi, P.: Qos-aware genetic cloud brokering. Future Generation Computer Systems 75, 1–13 (2017)
[247] Li, X., Pan, L., Liu, S.: An online service provisioning strategy for container-based cloud brokers. Journal of Network and Computer Applications 214, 103618, (2023)
[248] Rogers, O., Cliff, D.: A financial brokerage model for cloud computing. Journal of Cloud Computing: Advances, Systems and Applications 1(1), 1–12 (2012)
[249] Wang, X., Wu, S., Wang, K., Di, S., Jin, H., Yang, K., Ou, S.: Maximizing the profit of cloud broker with priority aware pricing. In: 2017 IEEE 23rd International Conference on Parallel and Distributed Systems (ICPADS), pp. 511–518, (2017). IEEE
[250] Mei, J., Li, K., Tong, Z., Li, Q., Li, K.: Profit maximization for cloud brokers in cloud computing. IEEE Transactions on Parallel and Distributed Systems 30(1),190–203, (2018)
[251] Sathish, A., Dsouza, D., Ballal, K., Archana, M., Singh, T., Monteiro, G.:Advanced mechanism to achieve qos and profit maximization of brokers in cloud computing. EAI Endorsed Transactions on Cloud Systems 7(20) (2021)
[252] Iturriaga, S., Nesmachnow, S., Dorronsoro, B.: Optimizing the profit and qos of virtual brokers in the cloud. Cloud Computing: Principles, Systems and Applications, 277–300 (2017)
[253] Li, X., Pan, L., Liu, S.: A survey of resource provisioning problem in cloud brokers. Journal of Network and Computer Applications 203, 103384 (2022)
[254] Jyoti, A., Shrimali, M.: Dynamic provisioning of resources based on load balancing and service broker policy in cloud computing. Cluster Computing 23(1), 377–395 (2020)
[255] Valarmathi, R., Sheela, T.: Differed service broker scheduling for data centres in cloud environment. Computer Communications 146, 186–191 (2019)
[256] Jyoti, A., Shrimali, M., Tiwari, S., Singh, H.P.: Cloud computing using load balancing and service broker policy for it service: a taxonomy and survey. Journal of Ambient Intelligence and Humanized Computing 11, 4785–4814 (2020)

[257] Alwada'n, T., Al-Tamimi, A.-K., Mohammad, A.H., Salem, M., Muhammad, Y.:Dynamic congestion management system for cloud service broker. International Journal of Electrical and Computer Engineering (IJECE) (2023)
[258] Ray, B.K., Khatua, S., Roy, S.: Negotiation based service brokering using game theory. In: 2014 Applications and Innovations in Mobile Computing (AIMoC), pp. 1–8 (2014). IEEE
[259] Shi, T., Ma, H., Chen, G., Hartmann, S.: Location-aware and budget-constrained service brokering in multi-cloud via deep reinforcement learning. In: Service-Oriented Computing: 19th International Conference, ICSOC 2021, Virtual Event,November 22–25, 2021, Proceedings 19, pp. 756–764 (2021). Springer
[260] Shannaq, F., Alshorman, A., Al-Sayyed, R., Shehab, M., Alomari, W.: Weighted service broker algorithm in cloud environment. Informatica 48(7) (2024)
[261] Chauhan, S.S., Pilli, E.S., Joshi, R.C.: Bss: a brokering model for service selection using integrated weighting approach in cloud environment. Journal of Cloud Computing 10, 1–14 (2021)
[262] Singh, N.K., Jain, A., Arya, S., Bhambu, P., Shruti, T., Chaudhary, V.K.: Cloud service broker using ontology-based system. Engineering Proceedings 59(1), 11 (2023)
[263] Achar, R., Thilagam, P.S.: A broker based approach for cloud provider selection.In: 2014 International Conference on Advances in Computing, Communications and Informatics (ICACCI), pp. 1252–1257 (2014). IEEE
[264] Vimercati, S.D.C., Foresti, S., Livraga, G., Piuri, V., Samarati, P.: A fuzzy-based brokering service for cloud plan selection. IEEE Systems Journal 13(4), 4101–4109 (2019)
[265] Shivakumar, U., Ravi, V., Gangadharan, G.: Ranking cloud services using fuzzy multi-attribute decision making. In: 2013 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE), pp. 1–8 (2013). IEEE
[266] Qu, L., Wang, Y., Orgun, M.A.: Cloud service selection based on the aggregation of user feedback and quantitative performance assessment. In: 2013 IEEE International Conference on Services Computing, pp. 152–159 (2013). IEEE
[267] Vakili, M., Jahangiri, N., Sharifi, M.: Cloud service selection using cloud service brokers: approaches and challenges. Frontiers of Computer Science 13, 599–617 (2019)
[268] Ionescu, S.: Best cloud broker of 2024 (2023). https://www.techradar.com/best/best-cloud-brokers
[269] Sonmez, C., Ozgovde, A., Ersoy, C.: Edgecloudsim: An environment for performance evaluation of edge computing systems. Transactions on Emerging Telecommunications Technologies 29(11), 3493 (2018)