

**BROKER-BASED FRAMEWORK FOR SERVICE  
ORCHESTRATION IN CLOUD AND FOG  
COMPUTING ENVIRONMENTS**

*Thesis submitted in fulfillment of the requirements for the Degree of*

**DOCTOR OF PHILOSOPHY**

By

**MANDEEP KAUR**



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING  
JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY,  
WAKNAGHAT, H.P.

MARCH 2024

@Copyright JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY

WAKNAGHAT

MARCH, 2024

ALL RIGHTS RESERVED

# Table of Contents

<b>Table of Contents</b>	<b>Page Number</b>
<i>TABLE OF CONTENTS</i>	<i>i</i>
<i>DECLARATION BY THE SCHOLAR</i>	<i>iv</i>
<i>SUPERVISOR'S CERTIFICATE</i>	<i>v</i>
<i>ACKNOWLEDGEMENT</i>	<i>vi</i>
<i>ABSTRACT</i>	<i>vii</i>
<i>LIST OF ACRONYMS AND ABBREVIATIONS</i>	<i>viii</i>
<i>LIST OF FIGURES</i>	<i>x</i>
<i>LIST OF TABLES</i>	<i>xii</i>
CHAPTER-1 INTRODUCTION	1-9
1.1 Orchestration	1
1.2 Distributed Systems	2
1.3 High Performance Distributed Systems	3
1.4 Load, Load Balancing, and Job Scheduling	4
1.5 Brokers	4
1.6 Open Issues	5-7
1.7 Motivation	7
1.8 Research Gap	7
1.9 Objectives	8
1.10 Organization of thesis	9
CHAPTER-2 LITERATURE REVIEW	11-38
2.1 Introduction	11
2.2 Broker-based Systems and Orchestrations	11-18
2.3 Load Balancing in Particular Public Cloud using S/W Agents	18-25
2.4 Dual Broker-based Framework (BBF) for IoT Job Scheduling	25-31
2.5 Multilevel Broker-Based Framework (BBF) For Better Geographic Coverage of Resources	31-37
2.6 Summary	37-38
CHAPTER-3 PARTITIONED PUBLIC CLOUD	39-48

3.1 Introduction	39
3.2 Background	39-41
3.2.1 Public Cloud	39
3.2.2 Partitioning a Public Cloud	39-40
3.2.3 Software Agents	41
3.3 Proposed Framework	41-45
3.3.1 Architecture	41-42
3.3.2 Application job distribution across cloud and fog Layer	42-43
3.3.3 Assessing the load status of FEs and FNs	43-45
3.3.4 Job Assignment	45
3.3.5 Complexity Analysis	45
3.4 Experimental Results and Discussion	46-48
3.5 Summary	48
<b>CHAPTER-4 DUAL BROKER-BASED JOB SCHEDULING</b>	<b>49-64</b>
4.1 Introduction	49
4.2 Background	49-50
4.2.1 Fog Computing	49-50
4.2.2 Naïve Bayes	50
4.2.3 Self Organizing Map (SOM)	50
4.3 Proposed Framework	51-57
4.3.1 Architecture	51-53
4.3.2 Naïve Bayes Algorithm for the Proposed Framework	53-54
4.3.3 Modified Self Organizing Maps (MSOM)	54-57
4.4 Experimental Results and Discussion	57-63
4.5 Summary	63-64
<b>CHAPTER-5 MULTILEVEL BROKER-BASED JOB SCHEDULING</b>	<b>65-80</b>
5.1 Introduction	65
5.2 Background	65-66
5.2.1 Coverage of Geographic Region	65
5.2.2 Hexagonal Fog Environment Organization	65-66
5.3 Proposed Framework	66-73
5.3.1 Architecture	66-67
5.3.2 The process for the selection of a FE	67-73

5.4 Experimental Results and Discussion	73-80
5.5 Summary	80
CHAPTER-6 CONCLUSION AND FUTURE	81-86
6.1 Thesis Summary	81
6.2 Concluding Remarks	81-82
6.3 Contribution	82-84
6.4 Future Scope	84
6.5 Summary	84-85
REFERENCES	86-93
LIST OF PUBLICATIONS	94
SYNOPSIS	95-116

## DECLARATION BY THE SCHOLAR

---

I hereby declare that the work reported in the Ph.D. thesis entitled, “**BROKER-BASED FRAMEWORK FOR SERVICE ORCHESTRATION IN CLOUD AND FOG COMPUTING ENVIRONMENTS,**” submitted at **Jaypee University of Information Technology, Wagnaghat, Solan (HP), India** is an authentic record of my work carried out under the supervision **Dr. Rajinder Sandhu**, Jaypee University of Information Technology, Solan, (HP) India and **Dr. Rajni Mohana**, Jaypee University of Information Technology, Solan, (HP) India. I have not submitted this work elsewhere for any other degree or diploma. I am fully responsible for the contents of my Ph.D. thesis.



Mandeep Kaur

Enrolment No.: 166201

Computer Science and Engineering

Jaypee University of Information Technology,

Wagnaghat, Solan (HP), India.

March 2024

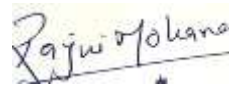
## SUPERVISOR'S CERTIFICATE

---

This is to certify that the work reported in the Ph.D. thesis entitled “**BROKER-BASED FRAMEWORK FOR SERVICE ORCHESTRATION IN CLOUD AND FOG COMPUTING ENVIRONMENTS**” submitted by **Mandeep Kaur** at **Jaypee University of Information Technology, Wagnaghat, Solan (HP), India**, is a bonafide record of her original work carried out under my supervision. This work has not been submitted elsewhere for any other degree or diploma.



Dr. Rajinder Sandhu  
Computer Science and Engineering  
Jaypee University of Information  
Technology, Wagnaghat,  
Solan, (HP), India  
March 2024



Dr. Rajni Mohana  
Computer Science and Engineering  
Jaypee University of Information  
Technology, Wagnaghat,  
Solan, (HP), India  
March 2024

## ACKNOWLEDGEMENT

---

First, I would like to thank my guide, Dr. Rajinder Sandhu, and co-guide, Dr. Rajni Mohana, in the Department of Computer Science and Engineering and Information Technology, for their support throughout this research work. I am thankful to them for sharing their knowledge and experience in the Distributed Systems (DS) field and allowing me to work under them. I could not finish this thesis without their advice and assistance. I owe them for their constant motivation and priceless advice in every aspect of this journey. I indeed consider myself fortunate to have had the chance to work with such wonderful people.

I want to thank our Honorable Vice Chancellor, Prof. (Dr.) Rajinder Kumar Sharma and Director & Academic Head Prof. (Dr.) Ashok Kumar Gupta to promote the research and facilitate resources in the institution. I would also like to thank Prof. Dr. Vivek Kumar Sehgal, Head Department of CSE&IT, for constant guidance, research facilities, and resources for my research work. I would also like to thank my doctoral committee members, Dr. Pradeep Kumar Gupta, Dr. Ruchi Verma, and Dr. Gopal Bisht, for their valuable feedback, critical reviews during presentations, and time-to-time help.

I am also grateful for the support from the JUIT, Wagnaghat; in particular, I thank all staff at the Department of Computer Science and Engineering and Information Technology, JUIT, Wagnaghat, who have been extremely helpful on numerous occasions. I thank my fellow Ph.D. friends for their consistent help and valuable discussion.

I am blessed with a lovely family and group of friends. I want to thank my parents, S. Major Singh and Smt. Surjit Kaur, Brothers Mr. Balwinder Singh and Mr. Javaid Ahmed, Sisters Ms. Kulwinder Kaur, Ms. Mandeep Kaur and Ms. Tanveera Hassan, my nephews Manpreet Singh, Gurmeet Singh, Paramjit Singh, and niece Silvi, for supporting me practically, emotionally and morally throughout this journey. I thank Dr. Pawan Kumar for constantly criticizing the negativity and motivating me to stay focused, Dr. Komal Singh Gill for his calm, friendly and in few cases, miracle like support, Dr, Ramandeep Kaur, Dr. Yasir Aafaq and all my amazing colleagues and friends at my workplace for their unconditional support and motivation during this period. Above all, I am thankful to the almighty for showering peace, patience, and positivity upon me, which enabled me to conclude this journey successfully.



# ABSTRACT

---

Distributed Systems have significantly contributed to the evolution of the field of computing by enabling job distribution and resource sharing. Distributed computing paradigms such as Cloud and Fog have eased users by making high-end resources available to the user without even possessing them physically. Users of these computing paradigms can use resources by paying nominal costs per their usage, scale their subscription, or shift from one set of resources to another as per their changing computational requirements. In this scenario, efficiently handling the requests and resources becomes crucial so that the available infrastructure can be utilized optimally. Job scheduling and load balancing are the formal ways to map the job requests on appropriate resources in DS appropriately. Both job scheduling and load balancing are prevalent research topics among researchers, and a considerable volume of literature is available to introduce tools and techniques for it. These techniques are expected to resolve the challenges faced by distributed computing, such as imbalanced load, the geographical coverage of resources, over/under utilization of resources, meeting Service Level Agreement, maintaining Quality of Service, and curbing the number of denied jobs. This thesis presents three frameworks: the first is a software agent-based broker framework for load balancing in a Partitioned Public Cloud, an ML classification-based dual broker framework that integrates QoS parameters for scheduling applications' jobs in FEs. This framework categorizes the resources available at the nodes as compute-intensive, memory-intensive, and GPU-intensive.

The suitable category of the newly submitted job is determined using Naïve Bayes Theorem, and these jobs are mapped onto the suitable category of the resources using Self-Organizing Maps. In addition to this, QoS parameters availability, physical distance, latency, and throughput are integrated into this system. The third framework proposed in this thesis is a multilevel hierarchical broker for job scheduling in hexagonal FEs for better global geographical coverage of distributed and ad-hoc Fog services to select the Fog node based upon a newly introduced parameter: QoS Score. Hexagonal Fog environments are introduced for geographic grouping of the Fog resources for better coverage of geographical regions of Fog resources. Experiments are performed using CloudSim, Aneka, and Azure platforms. Results have proved the improved performance of proposed frameworks in terms of average response time, average wait time, average completion time, resource availability, and average resource utilization compared to the conventional methodologies used for job scheduling in Cloud and Fog environments.

## List of Acronyms and Abbreviations

---

ACT	Average Completion Time
AET	Average Execution Time
ACET	Actual Execution Time
ART	Average Response Time
ARU	Average Resource Utilization
AWT	Average Waiting Time
BBF	Broker-based Framework
BW	Bandwidth
CB	Cloud Broker
CCB	Central Controller Broker
CC	Cloud Computing
DS	Distributed Systems
EC	Edge Computing
FC	Fog Computing
FE	Fog Environment
FN	Fog Node
GBR	Global Broker
GR	Geographic Region (GR)
IoT	Internet of Things
L1B	Level-1 Broker
L2B	Level-2 Broker
L3B	Level-3 Broker
LBR	Local Broker
MET	Maximum Execution Time
ML	Machine Learning
PB	Partition Broker
PC	Public Cloud
PP	Physical Proximity
PPC	Partitioned Public Cloud
QD	Quality of Devices

QoS	Quality of Service
RoI	Region of Interest
RAV	Resource Availability
SA	Software Agent
SLA	Service Level Agreement
SOM	Self-Organizing Map
TET	Total Execution Time
VM	Virtual Machine

## List of Figures

Figure 1.1	Roles played by the orchestration	1
Figure 1.2	Broker as a service orchestration agent	2
Figure 1.3	Categories of Distributed Systems	3
Figure 1.4	Cloud vs. Fog computing paradigm	3
Figure 1.5	Overview of a broker-based system	4
Figure 3.1	Size-wise comparison of Cloud types	40
Figure 3.2	Geographically partitioned public Cloud	40
Figure 3.3	Architecture: a framework for load balancing in PPC	42
Figure 3.4	Job segregation: Cloud vs Fog	43
Figure 3.5	MET Comparison	46
Figure 3.6	TET Comparison	47
Figure 3.7	AET Comparison	47
Figure 4.1	Positioning the Edge Devices and FNs in FE	49
Figure 4.2	High-level architecture: Dual broker-based framework	51
Figure 4.3	Triplet generation based upon QoS Parameters And Functional Requirements	54
Figure 4.4	Use of SOM to map the triplet on the resource cluster	55
Figure 4.5	Testbed for the performance evaluation of the proposed framework	58
Figure 4.6	ARU of available computing infrastructure	60
Figure 4.7	RAV comparison of the installed infrastructure	60
Figure 4.8	ART comparison of installed infrastructure	61
Figure 4.9	AWT comparison of installed infrastructure	61
Figure 4.10	ACT comparison of installed infrastructure	62
Figure 5.1	Geographic area partitioned in hexagon GRs for Fog resources	66
Figure 5.2	Interaction between the L2B and L1B	67
Figure 5.3	Interaction between L1B, L2B and L3B	67
Figure 5.4	Increase in the number of FNs with layer count	68
Figure 5.5	Interaction between L2B and L1Bs of a layer	69
Figure 5.6	Assigning the job to a suitable FE	71
Figure 5.7	Testbed for experimental setup and performance analysis	73

Figure 5.8	Setup of FEs for the testbed	75
Figure 5.9	Number of jobs generated by every application at a different time interval	76
Figure 5.10	ARU Comparison	76
Figure 5.11	RAV Comparison	77
Figure 5.12	ART Comparison	77
Figure 5.13	AWT Comparison	78
Figure 5.14	ACT Comparison	79
Figure 6.1	Thesis Summary	85

## List of Tables

Table 1.1	Open Issues in the field of DS	5
Table 2.1	Types and implementation of Brokers on various resources	15
Table 2.2	Summary of literature survey for PPC	22
Table 2.3	Summary of literature survey for dual BBF using ML techniques and QoS parameters	28
Table 2.4	Summary of literature survey for multilevel BBF for better geographic coverage of Fog resources	34
Table 3.1	Complexity Analysis	45
Table 3.2	Experimental Setup	46
Table 3.3	Comparison with other approaches	47
Table 4.1	Record kept by LBR	52
Table 4.2	Resource Availability based node types	53
Table 4.3	Details of QoS parameters	55
Table 4.4	Summary of obtained results	62
Table 4.5	Proposed framework vs other solutions	62
Table 5.1	Weightage of QoS parameters to calculate the QoS score	69
Table 5.2	Applications used in Testbed	74
Table 5.3	Number and Configuration of Nodes	74
Table 5.4	Parameters for Time used in the Proposed Framework's TestBed	74
Table 5.5	Summary of the obtained results	79
Table 5.6	Proposed framework vs other solutions	79

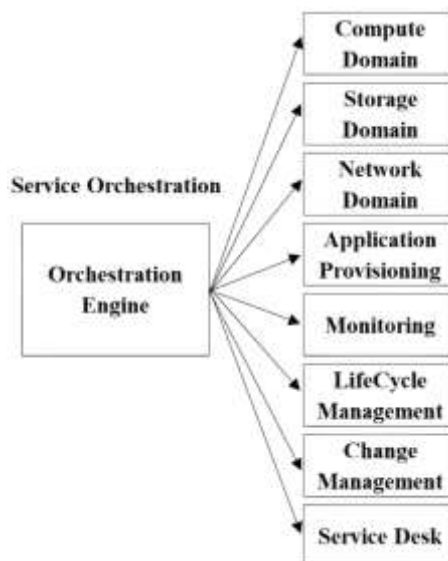
# CHAPTER 1

## INTRODUCTION

This thesis addresses the complexities of orchestrating services in Cloud and Fog computing environments by proposing Broker-Based Frameworks (BBFs). These frameworks leverage intelligent brokering strategies to optimize resource management, service orchestration, and deployment in heterogeneous computing environments. The thesis outlines the architecture and components of the BBF, emphasizing its potential to enhance responsiveness, resource efficiency, resource utilization, and resource availability in Distributed Computing Systems. Overall, the proposed BBFs offer a solution-oriented approach to the challenges associated with service orchestration in dispersed and dynamic computing ecosystems.

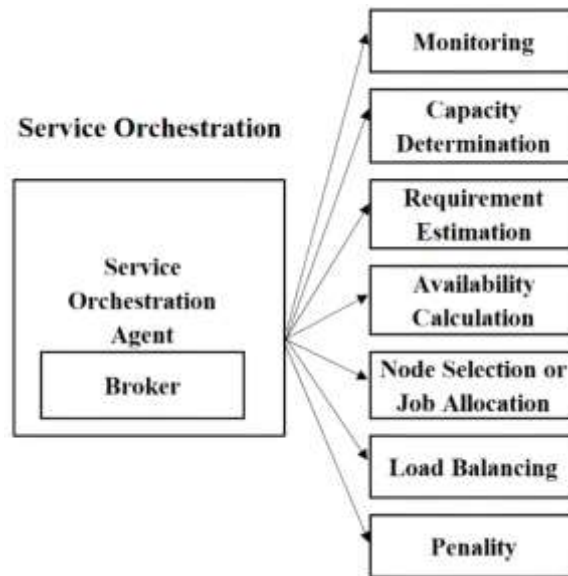
### 1.1 Orchestration

The term orchestration describes managing and synchronizing services, apps, and resources in Cloud and Fog Computing (FC) contexts. Cloud orchestration is the process of overseeing services and resources in a distant, centralized data center. It usually manages storage, networking, Virtual Machines (VMs), containers, and other Cloud infrastructure resources. Applications in Cloud settings can be automated, managed, and scaled using orchestration tools such as OpenStack, Docker Swarm, and Kubernetes. FC involves orchestrating computing, storage, and networking resources in edge devices, gateways, and local servers. Orchestration tools such as Cisco IOx, OpenFog, or edge computing (EC) platforms can manage and automate the services in FC environments. [1]–[3]



**Figure 1.1:** Roles played by the orchestration.

A service orchestration engine plans, directs, and automates the interactions between various services or systems in a network or application environment.



**Figure 1.2:** Broker as a Service Orchestration Agent

Orchestration is crucial in Cloud Computing (CC) and FC to simplify and automate application and service deployment, scaling, and management. It ensures reliable, scalable, and efficient resource utilization in complex and distributed computing environments. [4]–[6]

## 1.2 Distributed Systems

Distributed Systems (DS) refers to two or more computers working together through a network to finish a single job, which is distributed over a group of computers, heterogeneous in operating systems, node characteristics, network design, and communication medium [7]. The generic quality of DSs in a group of autonomous computer components is that these seem like a single cohesive system to the users [8]. DS comprises autonomous computing nodes for better accessibility, transparency, openness, and scalability [9].

To optimize autonomous communication between nodes, a DS's performance can be measured through execution time, throughput, efficiency, system utilization, turnaround time, waiting time, response time, overheads, and reliability [9]–[12]. Figure 1.3 demonstrates the categories and relevant examples of DS.



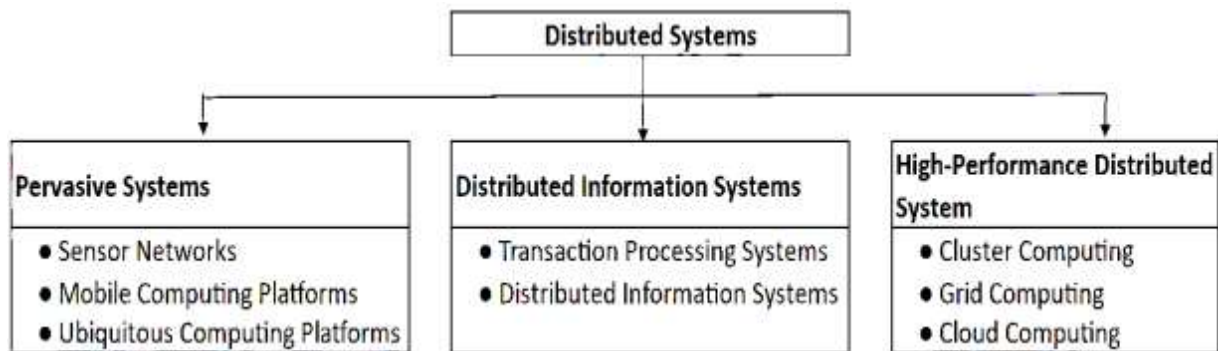


Figure 1.3: Categories of distributed systems.

### 1.3 High-Performance Distributed Systems

This category includes computing paradigms such as cluster computing, grid computing, and CC. FC is another paradigm that enables delivering Cloud application services closer to the Internet of Things (IoT) devices at the edge rather than relying on a distance Cloud. In contrast to transferring IoT data to the Cloud, which processes and stores it remotely on IoT computers, Fog provides better latency-sensitive resources. Thus, FC is the ideal choice for enabling the IoT to provide efficient and trustworthy services to many clients. Figure 1.4 highlights the fundamental differences between Cloud and FC paradigms.

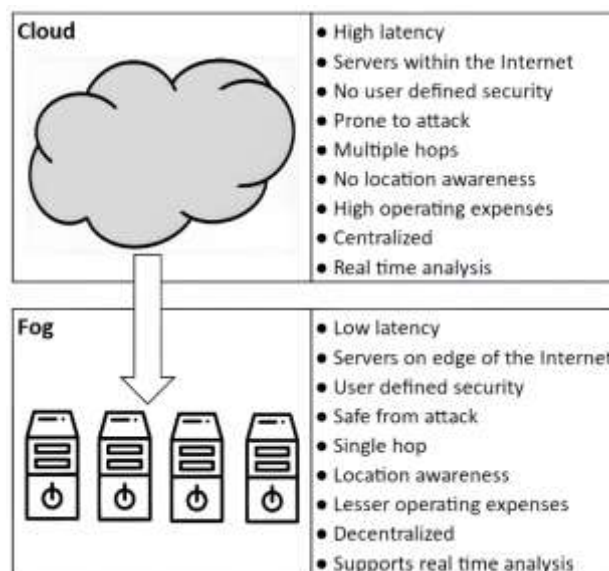


Figure 1.4: Cloud vs Fog computing paradigms.

## 1.4 Load, Load Balancing and Job Scheduling

In the context of CC, load refers to the volume of work or demand on a server, network, or system at a specific time. It mainly concerns how much resource is used or consumed in a Cloud-based infrastructure. Compute, network and storage loads are just a few examples of the components that comprise the term load in distributed systems [13]. Load balancing in DS evenly distributes network traffic or computational workload across multiple servers that optimize resources, scalability, fault tolerance, performance enhancement, and cost efficiency [14] [15]. On the other hand, job scheduling in cloud and fog computing efficiently allocates computing resources and manages task execution to manage resource utilization, prioritization, fairness, adaptability, flexibility, and latency [16]–[18].

## 1.5 Brokers

In Cloud and FC environments, Cloud Brokers (CBs) and Fog Brokers are significant for resource management, communication facilitation, and operational optimization. To help choose and handle Cloud services, Cloud brokerage involves mediators who link consumers or businesses with Cloud service providers. Cloud service brokers (CSBs), such as Right Scale and Gravitant, are instances of CBs. They offer platforms for managing Cloud services from various vendors. Figure 1.5 demonstrates the positioning and contribution of the brokers in DS.

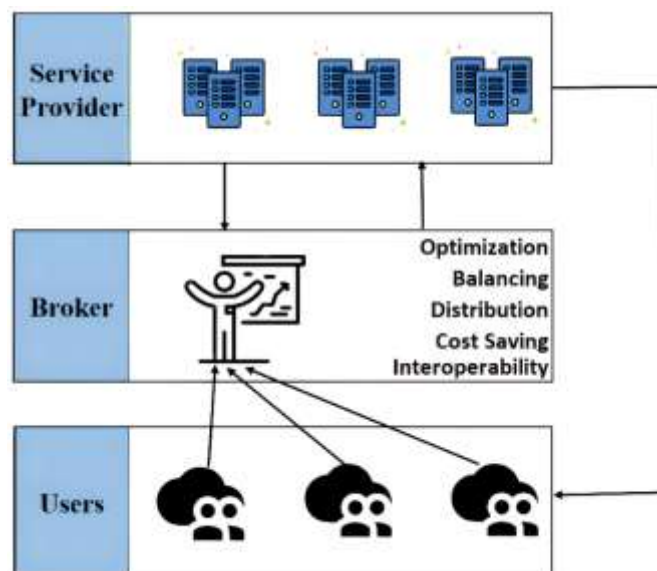


Figure 1.5: Overview of a broker-based system.

Companies that work with FC concepts include Cisco, IBM, and Microsoft. Some of these companies may also develop or offer Fog brokerage services. The brokerage model inevitably changes as technology develops and the demand for effective resource management in various computing environments increases.

### 1.6 Open Issues

Table 1.1 discusses some of the most commonly researched issues in the field of DS:

**Table 1.1:** Open Issues in the field of DS.

Open Issue	Description
The complex and dynamically changing Fog computing environment [19]–[22]	Heterogeneous Fog computing equipment leads to uneven resource distribution, causing latency issues and hindering overall system performance. Strategies like load balancing and resource optimization are needed to address these challenges.
The volume of requests and prolonged task queues [19], [22], [23]	Vital Fog Nodes (FNs) may be unable to process all of the service's input data because they have limited computational, storage, and other resource capabilities.
Mobility causes dynamic changes [19], [21], [23]	Mobile FN's dynamically influence Fog computing resources, necessitating adaptable resource allocation systems to manage the server shifting seamlessly.
Dynamically changing resource requirements [19]	Diverse application behaviors, time variations, and environmental conditions impact computing aspects like resource sharing, job scheduling, and task offloading, which shape the resource allocation in FC.

Load Balancing [19], [21], [23]	Load balancing techniques aid in optimizing task transfer between Fog and Cloud systems to minimize overload and processing time. However, exploratory strategies often favor more robust Fog resources, impacting workload balance.
Task Scheduling [19], [21], [23]	Task scheduling complexity arises from the substantial difference in capacity between FNs and Cloud servers, compounded by network heterogeneity and the uncertain wireless environment.
Real-Time Responsiveness [19], [21]–[24]	Managing real-time, delay-sensitive IoT applications alongside multidimensional complexities poses a key challenge for Fog computing networks.
Big Data Analysis [19], [23], [25]	Effectively distributing vast high-dimensional data from end-user devices and the IoT among resource-limited FNs poses a challenge for learning algorithms in big data analytics to yield reliable findings.
Load Prediction [19]–[21], [25]	Dynamic resource auto-scaling in Fog Environments (FEs) is crucial for efficiently distributing resources to varied workloads, preventing "Over-Provisioning" due to scarcity and "Under-Provisioning" from excess requests.

All the issues discussed in this section are generic. However, there are domain-specific variations of these issues, such as reinforcement learning [19], fuzzy logic [20], offloading

mechanism [21], bandit learning [22], and digital twin perspective [25]. All these domains have different perspectives to view these issues and research their solutions in different dimensions.

## **1.7 Motivation**

The rapid and effective placement of application jobs on available FNs for latency-sensitive Internet of Things (IoT) jobs improves the performance of a DS. If nodes are selected correctly and effectively, the probability of achieving low latency increases automatically [26], [27].

This thesis proposes job scheduling and load-balancing solutions for high-performance systems. These problems, being highly significant to the DSs, are widely researched by researchers, resulting in various techniques in this field. As per our knowledge, the existing solution could not address the following issues which our work is intended to resolve:

- Lack of autonomous behavior in scheduling and balancing components,
- Classification of available resources and mapping jobs to appropriate classes,
- Single point of load scheduling and balancing,
- Geographic coverage of Fog resources.

## **1.8 Research Gaps**

To the best of our knowledge, some of the existing gaps in the field of study are presented in this section:

1. PC has a large search space due to the accommodation of a massive number of Cloud or Fog resources which needs mechanism to introduce autonomous behaviour in and across the partitions.
2. Considering FN as a single bulk of resources and allocating jobs without segregation and QoS parameter consideration is not optimal for resource allocation in the Fog Environments.
3. Load increases on scheduling brokers with expansion in the geographic areas, and there are chances that the scheduling entities overlap some of the resources left unattended.

## 1.9 Objectives

As a result of our work, we suggest three progressive frameworks, which are:

- *To design a software agent (SA)-based broker framework for load balancing in a Partitioned Public Cloud (PPC)*

To reduce the search space in the Public Cloud (PC), to reduce the congestion at a single job scheduler, and to insert autonomous behavior in the load balancing technique, an SA-based broker framework is introduced for a single Cloud, which resulted in a reduced number of detained jobs, improved makespan, better average execution time (AET), maximum execution time (MET) and total execution time (TET). This method yielded good outcomes, but it can be enhanced by incorporating machine learning (ML) classification methods for automated decision-making and considering Quality of Service (QoS) parameters when evaluating system performance.

- *To develop an ML-based dual broker framework by integrating QoS parameters for scheduling applications' jobs in FEs*

An ML classification-based dual broker framework is proposed by integrating QoS parameters. Two brokers are used: Global Broker (GBR) deployed at the centralized level and Local Brokers (LBRs) deployed at each FE. This framework considers three QoS parameters: Physical Proximity (PP), Bandwidth (BW), and Device Quality (QD). A combination of Naïve Bayes Theorem and Self-Organizing Map (SOM) is used in this approach to classify and map jobs on a suitable set of resources (Generic, Compute Intensive, GPU Intensive, and Memory Intensive). This approach improves average resource utilization (ARU), resource availability (RAV), average response time (ART), average waiting time (AWT), and average completion time (ACT). However, this approach does not address geographical resource coverage, so some pervasively scattered nodes may be left unattended or overlapped. Congestion issues may still arise at dual-level hierarchical brokers when considering globally scattered Fog resources.

- *To develop a multilevel hierarchical broker for job scheduling in hexagonal FEs for better global geographical coverage of distributed and ad-hoc Fog services*

The third approach is a multilevel hierarchical broker-based framework for job scheduling in hexagonal FEs for better global geographical coverage of distributed and ad-hoc Fog services. The RoI (Region of Interest) in this chapter is divided into hexagonal regions due to the equal distance between their centers and ensuring complete coverage of FNs. This approach considers

global-level Fog resources across the Geographical Regions (GRs) and is flexible enough to add multiple hierarchical brokers to manage the increased congestion along with the expansion to multiple GRs. As an output of this approach, we could achieve higher RAV, ARU, ART, AWT, and ACT.

## **1.10 Organization of the Thesis**

The thesis has been organized into 6 chapters as described below:

**Chapter 1:** This chapter introduces the basic concepts of DS and narrows their scope while discussing the work's background concepts, open issues, motivation, and objectives.

**Chapter 2** presents the existing state-of-the-art approaches proposed by other researchers. The literature review is divided into four sections: first, basic concepts and the three sections contain a literature review corresponding to each objective.

**Chapter 3:** This chapter discusses partitioning the public Cloud (PC) and applying software agents, in the form of brokers, for load balancing.

**Chapter 4:** This chapter discusses an application's job allocation to suitable resources using ML classification techniques in a dual broker-based system. Two ML techniques are used in the proposed framework: the Naïve Bayes approach and the SOM.

**Chapter 5:** This chapter presents a multilevel broker-based approach considering the global geographical coverage of the Fog resources using hexagonal FEs. The proposed framework introduces a parameter QoS score to rank the available resources based on this score.

**Chapter 6:** This chapter summarizes and concludes the thesis and presents the potential future directions.

The thesis ends with a list of references and synopsis. This thesis aims to propose and implement the frameworks for efficiently scheduling applications' jobs on DS, such as Cloud and FC environments. Though CC and FC paradigms have resolved the issues of resource scarcity to a noticeable level, carefully allocating available resources to the jobs ensures optimal

resource utilization. This thesis resolves job scheduling at three different levels by introducing job scheduling frameworks for each.



# CHAPTER 2

## LITERATURE SURVEY

### 2.1 Introduction

This chapter presents the systematic literature review for all three objectives in this thesis. The following sections of the chapter present the literature studied for each proposed framework. Section 2.2 contains the set of papers relevant to broker-based systems and orchestration. The literature referred to in Section 2.3 is related to basic CC concepts, partitioning, load-balancing techniques, SAs, and PCs. The literature in Section 2.4 is related to the FC, ML techniques, IoT applications' job scheduling techniques, and QoS parameters. Section 2.5 contains the literature surveyed for IoT applications' job scheduling techniques, Geographic Region (GR) coverage, QoS parameters, and hexagonal structures.

### 2.2 Broker-based Systems and Orchestration

In DS, brokers can be used on various resources, including BW, Cloud, information, IoT, QoS, resource, and service brokers. Each one of these plays a particular role in the systems in which they are used. The jobs carried out by brokers in a DS can vary from application to application, even though their overall role remains constant [28]. In DS, brokers can serve a variety of roles in message queueing systems, content delivery networks, collaborative content distribution networks, and the security industry [28] [10] [29] [30].

**In 2016, Suomalainen et al.** [31] designed an improved architecture based on an information brokering system's privacy and security requirements. They suggested an adaptive pseudo-optimization architecture that made privacy attacks more difficult and offered real-time updates on how reliable the platforms' privacy protection mechanisms were. The authors developed this architecture to enhance infrastructure utilization in smart cities through real-time information sharing at the district level. They focused on measuring energy consumption and found their framework to be helpful.

**In 2016, López et al.** [32] developed a scheduler using the assistance calculations Deferrable Server and Earliest Deadline First (EDF) (DS). The self-reconfiguration of the manufacturing framework was quite enticing because setbacks could occur on any one of the specifics of the

assembling framework. The article proposed a method to tackle this issue. The approach involved analyzing the time taken by the system to produce an item and adjusting the rate of plant components of the specific operation accordingly. This helped identify the deficiencies of the system and address them effectively.

**In 2018, Ferdosian et al.** [33] presented a two-level downlink scheduling system that resolved significant difficulties by attempting to deliver all sorts of traffic while striving to meet the performance constraints of the LTE. A cogent resource allocation method was created for the higher-level algorithms using a game theory model to guarantee per-class fairness. The greedy-knapsack method was improved at a lower level for improved QoS and throughput to distribute resources to the best possible bearers. The suggested method's effectiveness was assessed under normal and overload network conditions. According to simulation results, the suggested scheduling algorithm performed better than reference scheduling strategies based on throughput, fairness, and QoS performance as measured by the rate of packet loss and latency for various service classes.

**In 2019, Hwang et al.** [34] introduced and outlined the U-mosquitto protocol, an improvement to the popular MQTT Mosquitto broker. They tried to expedite the delivery of crucial signals by sending them out more quickly than other system messages. Studies have demonstrated that as the number of publishers rises, so does the difference in delivery time between routine communications and urgent messages.

**In 2020, Mei et al.** [35] introduced a new function for CB, which acts as a mediator between Cloud providers and short-term Cloud users to reduce costs. Short-term customers often pay more than their usage due to hourly billing cycles, even when not using Cloud Resources (CRs) for long periods. Long-term users receive discounted CRs, while short-term users do not. To address this issue, the CB rents reserved VMs at a lower cost and then distributes these resources to users at a lower cost on demand. The authors of a recent study focused on CB configuration and calculating a fee that ensured maximum profit for the broker while lowering the cost for Cloud consumers. They modeled a profit maximization problem as an ideal multi-server setup price problem and suggested a heuristic strategy that used partial derivative and bisection search methods to optimize the problem. The authors also analyzed the factors

influencing the broker's profit, such as user demand, VM buy price, and VM sale price. A linear price-demand function is used to find the best options.

**In 2020, Oh et al.** [36] investigated the subject of user cost minimization in MCC networks. Different brokers assign CRs to mobile users in an MCC model, which was considered. A competitive approach and a compete-then-cooperate method were examined as performance bounds for Cloud reservation strategies. Cooperation can lead to significant gains over competition in markets with fewer brokers. However, in markets with many brokers, the benefits of cooperation are negligible, indicating no clear advantage to working together.

**In 2020, Zhang et al.** [37] focused on researching the price and selection of services for IoT applications in a multi-MEC (Multi Edge Computing) system with multiple Edge Cloud Service Providers (ESPs). The researchers specifically paid attention to the distribution of workload. They modeled the situation as a Stackelberg game, where the Cloud Service Broker (CSB) determines the Cloud service price and load balancing strategies to maximize revenue. Next, IoT users can select which ESP service they want. The backward induction method is used to find the best solutions. The proposed strategy is validated using simulation data.

**In 2020, Mishra et al.** [38] studied the progress made in M2M (Machine-to-Machine) protocol research, specifically focusing on MQTT, AMQP, and CoAP, over the past two decades. They found that MQTT research has outpaced the others. To assist academics and end-users in selecting a broker or client collection that best suits their needs, they have provided a classification to compare the features and characteristics of several MQTT solutions, including brokers and libraries, currently available in the public domain. Lastly, they highlighted some significant findings from my comparison and identified a few topics that require further investigation.

**In 2021, Akanksha et al.** [39] proposed a method to increase profit by optimizing the CB's setup. The maximization technique used by the CB is influenced by several factors, such as the customer's request, the selling price of the resource, the purchasing price, and the intensity of the request. The proposed approach seeks to optimize the profit of the CB by providing cloud infrastructure from an infrastructure vendor at a lower cost while meeting clients' demands.

Finally, the Hill-Climbing method is used to assign resources dynamically. The suggested solution used QoS and service price as determining variables to maximize the CB's net profit.

**In 2021, Razian et al.** [40] proposed a scalable anomaly-aware approach (SAIoT). The process has two main parts. The first part uses a machine learning-based anomaly detection technique to identify any pre-existing abnormal QoS records. The second part employs a reliable and efficient metaheuristic algorithm to discover an optimal composition close to ideal. The experimental findings derived from real-world data sets showed that their methodology achieved a composite plan's average QoS value enhancement by 30.64 percent for the same or less cost as earlier efforts, like information theory- and declared QoS-based methodologies.

**In 2021, Gruener et al.** [41] developed a resilience testing tool for MQTT brokers called MAYHEM 2021. This tool helps IoT practitioners, researchers, and broker suppliers make better architecture, design, and implementation improvement decisions. The researchers used MAYHEM for resiliency experiments on several MQTT brokers, including VerneMQ, Mosquitto, HiveMQ, and EMQ X. Their experiments yielded some interesting findings, such as: 1. MQTT QoS Level 0 is already robust against minor packet loss, 2. Most clustered MQTT brokers prioritize performance and availability over communication integrity; 3. Message loss may occur due to selected broker message persistency solutions.

**In 2021, Gruner et al.** [42] Self-Sovereign Identity (SSI) trust-enhancing attribute aggregation capability is proposed to be used by an Attribute Trust-enhancing Identity Broker (ATIB) to provide standard protocols and abstract SSI solutions. Although the brokered integration strategy, ATIB upholds a high level of security for users and does not violate any of the ten fundamental SSI requirements. The authors assessed ATIB's authentication process after connecting it to uPort, Jolocom, and HL Aries/Indy. These assessments included attributes used for authorization, performance measures, and SSI compliance review.

**In 2021, Abhishek et al.** [43] Offered the design and development of a Cloud service orchestrator that could assist the providers of application and network services in smoothly deploying their services on the required Cloud (Private or Public).

**Table 2.1:** Types and Implementation of Brokers on various resources

Ref. No.	Title	Year	Published In	Summary	Limitations
[31]	Enforcing secure and privacy-preserving information brokering in distributed information sharing	2016	IEEE transactions on information forensics and security	<b>Broker Type:</b> Information Broker <b>Issue Addressed:</b> An enhancing architecture based upon the security and privacy needs of an information brokering system	1. Single-level brokering, 2. QoS is not considered, 3. A single parameter is considered, which is cost, 4. Global coverage is not considered; 5. No clustering of resources and jobs.
[32]	ReconFigureuration Distributed Objects in an Intelligent Manufacturing Cell	2016	IEEE Latin America Transactions	<b>Broker Type:</b> Object Request Broker <b>Issue Addressed:</b> a scheduler utilizing the calculation EarliestDeadline First (EDF) and the help calculation Deferrable Server (DS)	1. Single-level brokering, 2. QoS is not considered, 3. A single parameter is considered, which is cost, 4. Global coverage is not considered; 5. No clustering of resources and jobs.
[33]	Fair-QoS Broker Algorithm for Overload-State Downlink Resource Scheduling in LTE Networks	2018	IEEE Systems Journal	<b>Broker Type:</b> QoS Broker <b>Issue Addressed:</b> A two-level scheduling system that supplies all types of traffic while attempting to fulfill LTE performance criteria solved the important issues.	1. Single-level brokering, 2. QoS is not considered, 3. A single parameter is considered, which is cost, 4. Global coverage is not considered; 5. No clustering of resources and jobs.

[34]	Modification of Mosquitto Broker for Delivery of Urgent MQTT Message	2019	IEEE Eurasia Conference on IoT, Communication, and Engineering (ECICE)	<b>Broker Type:</b> MQTT Broker <b>Issue Addressed:</b> Introduced the U-mosquitto protocol, which is an enhancement to the widely used MQTT Mosquitto broker	1. Single-level brokering, 2. QoS is not considered, 3. A single parameter is considered, which is cost, 4. Global coverage is not considered, 5. No clustering of resources and jobs.
[35]	Profit Maximization for Cloud Brokers in Cloud Computing	2020	IEEE Transactions on Parallel and Distributed Systems	<b>Broker Type:</b> Cloud Broker <b>Issue Addressed:</b> A new function of a Cloud broker, which operates as an intermediary between the Cloud provider and the Cloud user to lower the cost of Cloud usage for short-term customers, who often pay more than their usage	1. Single-level brokering, 2. QoS is not considered, 3. A single parameter is considered, which is cost, 4. Global coverage is not considered, 5. No clustering of resources and jobs.
[36]	Competitive Data Trading Model With Privacy Valuation for Multiple Stakeholders in IoT Data Markets	2020	IEEE Internet of Things Journal	<b>Broker Type:</b> IoT Broker <b>Issue Addressed:</b> An investigation of the subject of user cost minimization in MCC networks	1. Single-level brokering, 2. QoS is not considered, 3. A single parameter is considered, which is cost, 4. Global coverage is not considered, 5. No clustering of resources and jobs.
[37]	Service Pricing and Selection for IoT Applications Offloading in the Multi-Mobile Edge	2020	IEEE Access	<b>Broker Type:</b> IoT Broker <b>Issue Addressed:</b> An investigated service price and selection for IoT applications unloading a multi-MEC system	1. Single-level brokering, 2. QoS is not considered, 3. A single parameter is considered, which is cost, 4.

	Computing Systems				Global coverage is not considered, 5. No clustering of resources and jobs.
[38]	The Use of MQTT in M2M and IoT Systems: A Survey	2020	IEEE Access	<b>Broker Type:</b> IoT Broker <b>Issue Addressed:</b> Examination of the evolution of M2M protocol research (Message Queue Telemetry Transport (MQTT), CoAP, and AMQP) over the previous 20 years and discovered that MQTT research has surpassed the others	1. Single-level brokering, 2. QoS is not considered, 3. A single parameter is considered, which is cost, 4. Global coverage is not considered, 5. No clustering of resources and jobs.
[39]	Advanced Mechanism to Achieve QoS and Profit Maximization of Brokers in Cloud Computing	2021	EAI Endorsed Transactions on Cloud Systems	<b>Broker Type:</b> Cloud Broker <b>Issue Addressed:</b> A method centered on increasing profit by optimizing the Cloud broker's setup	1. Single-level brokering, 2. QoS is not considered, 3. A single parameter is considered, which is cost, 4. Global coverage is not considered, 5. No clustering of resources and jobs.
[40]	SAIoT: Scalable Anomaly-Aware Services Composition in Cloud IoT Environments	2021	IEEE Internet of Things Journal	<b>Broker Type:</b> QoS Broker <b>Issue Addressed:</b> a scalable anomaly-aware approach (SAIoT) with two primary parts: initially uses an ML anomaly detection method to eliminate any current anomalous QoS records, and the second employs a powerful and efficient metaheuristic algorithm to locate a close to ideal composition.	1. Single-level brokering, 2. QoS is not considered, 3. A single parameter is considered, that is, cost, 4. Global coverage is not considered, 5. No clustering of resources and jobs.

[41]	Towards Resilient IoT Messaging: An Experience Report Analyzing MQTT Brokers	2021	2021 IEEE 18th International Conference on Software Architecture (ICSA)	<b>Broker Type:</b> MQTT Broker <b>Issue Addressed:</b> developed MAYHEM, a tool to test the resilience of MQTT brokers, that aids IoT practitioners, researchers, and broker suppliers in architectural decisions and design and implementation improvement	1. Single-level brokering, 2. QoS is not considered, 3. A single parameter is considered, which is cost, 4. Global coverage is not considered, 5. No clustering of resources and jobs.
[42]	ATIB: Design and Evaluation of an Architecture for Brokered Self-Sovereign Identity Integration and Trust-Enhancing Attribute Aggregation for Service Provider	2021	IEEE Access	<b>Broker Type:</b> Identity Broker <b>Issue Addressed:</b> An Attribute Trust-enhancing Identity Broker (ATIB) uses Self-Sovereign Identity (SSI trust-enhancing ) attribute aggregation capacity to provide standard protocols and abstract dedicated SSI solutions	1. Single-level brokering, 2. QoS is not considered, 3. A single parameter is considered, which is cost, 4. Global coverage is not considered, 5. No clustering of resources and jobs.

### 2.3 Load Balancing in Partitioned Public Cloud using S/W Agents

Managing a large number of heterogeneous nodes in a public cloud is a challenging job. The geographic partitioning of the public cloud is a well-discussed solution to this challenge. The section below discusses a few load-balancing techniques for partitioned public clouds available in the literature:

**In 2013, Gaochao Xu et al.** [14] suggested a Cloud partitioning-based load-balancing method for PCs. They suggested changing the load balancing technology depending on the load at any given time. This technique manages the partitions with high load status by using game theory. However, this technique required more testing to establish an effective refresh rate and load degree calculation method. The compromise was that extensive testing was needed to ensure system availability and effectiveness.



**In 2014, Suguna et al.** [15] suggested a solution to handle dynamic Cloud partitioning and load balancing. The Partition Manager, which assigns a job to a partition, and the Job Distributor, which selects the node to which this job might be distributed, are crucial elements of this strategy. Partitions are built dynamically to provide adequate load distribution. The honey bee algorithm is used to do this. Beyond geographical boundaries, this paper left room for improvement in the transparency and Cloud division. To increase efficiency in the cloud environment, they employ game theory.

**In 2014, Sanjay et al.** [44] suggested a distributed algorithm that performs better for load balancing in a master-slave structure than the Closest Datacenter strategy regarding task distribution all through the system and optimized system performance... According to the study, a clustering method divides the network into clusters. A cluster is made up of all nodes. Inter-cluster communication (ICC) nodes are present in each cluster. Clustering occurs during network initialization.

**In 2014, Hui Zhang et al.** [45] the difficulties of service availability and dependability, a lack of Service Level Agreement (SLA), customer data security and privacy, and government compliance regulation requirements all dealt with in their work. With a quick and frequent data detection technique at its foundation, they have suggested an intelligent workload factoring service for proactive job management. This method aids in factoring requests based on both data volume and content. The main design aims of this workload refactoring are load redirection to prevent overloading scenarios, workload dynamics smoothening in the base zone application platform, and load decomposition to make the flash Cloud zone application platform adaptable. Improvements are needed, including managing data security for hybrid platforms, handling data replication and consistency in flash crowd zones, and implementing load-balancing techniques in two zones.

**In 2015, Priti Singh et al.** [46] suggested determining the load degree based on turnaround time. With this method, load PBs can enhance their load-balancing tactics when using a PC.

**In 2015, Stefano Sebastio et al.** [47] suggested conducting a preliminary analysis of a Cloud partitioning strategy that divides job execution requests among a volunteer Cloud. The Google workload data trail is used in a simulation-based statistical analysis for validation. Evaluations

of this model are conducted utilizing the same random data set and a comparison of the outcomes of a proposed method and an un-partitioned Cloud. This strategy offers room for improvement, such as adding additional sophistication by employing a bio-inspired solution. Performance can be enhanced by adding portable performance monitors. Workload management could be improved with a workload classification system.

**In 2015, Amir Nahir et al.** [48] attempted to remove the choice from Job's Critical Path to increase scheduling decision accuracy. They have employed job duplication in addition to the Schedule First and Manage Later methodology. They recommend replicating jobs and distributing these replicas across other servers in their suggested model. The server that chooses it first notifies any servers with a copy of this job. The main objective of this strategy is to simplify job scheduling by removing load-balancing jobs from the selection of VMs. However, this method lengthens the processing time by factoring in signal propagation delay.

**In 2017, Michael Pantazoglou et al.** [49] made an effort to call attention to several issues, such as difficulty with scalability, flexibility, energy efficiency, and high operational costs. For this, they have developed methods that combine preliminary VM deployment and partial and complete VM migration. The primary VM placement procedure may create or delete VMs at any time, depending on the load that specific VM is bearing. Overloaded compute nodes employ the partial VM migration mechanism to transfer a small number of jobs to one or more additional compute nodes. If a computing node is underutilized, the whole migration procedure is employed to move the entire load. This transfer aims to lower the data center's overall energy usage. Table 1 summarizes the static load balancing research that has been done up to this point.

**In 2021, Abdullah et al.** [50] advised a Deep Neural Networks Energy Cost-Efficient Partitioning and Task Scheduling (DNNECTS) algorithm framework, which includes a setup of the following elements: task sequencing, scheduling, and application partitioning. The suggested methods are demonstrated through experimental findings concerning application costs and energy consumption in a dynamic setting.

**In 2022, Chi Zhou et al.** [51] proposed RLCut, which uses Reinforcement Learning (RL) to manage the problem's complexity. In particular, RLCut uses multi-agent learning, which is more effective than single-agent RL, and adds a sampling-based optimization to adaptively

regulate the training process to fulfill the necessary trade-off between partitioning efficacy and efficiency for per under-graph dynamicity. The results of geo-distributed graph analytics are improved by RLCut by 10%–100% with equivalent overhead compared to state-of-the-art static partitioning approaches, according to experiments employing real Cloud datacenters and real-world graphs. We can further boost results by up to 43% if users are willing to put up with more partitioning overhead. Compared to cutting-edge dynamic partitioning, RLCut can increase performance by as much as 60% with different graph shifting frequencies.

**In 2022, Xiao et al.** [52] proposed a two-stage simulation task partitioning technique that calculates resource prediction data. The features of the collected data are analyzed in the initial stage to determine the best feature dimensions. The number of resources needed for the shortest runtime is determined by ranking the expected results after using a stacking ensemble learning approach to forecast the simulation runtime in a given scenario of assigning different computer resources. In the subsequent phase, a multi-weight graph structure is created to depict the dynamic interaction of simulation elements to minimize the load imbalance on each computer node and maximize the number of connections between nodes. Then, the simulated annealing optimization process is applied to divide various weight graphs and assign simulation jobs to resources.

**In 2022, Lan et al.** [53] presented a method for partitioning and orchestrating computer vision applications on heterogeneous edge computing systems, considering both CPUs and GPUs; there is a system framework called EDGE VISION. The program is divided into discrete tasks coordinated and distributed onto the many heterogeneous edge nodes using EDGE VISION, summarizing the heterogeneous hardware resources and task runtime environments. Aiming to reduce processing delay and overall system cost, we recommend two scheduling techniques in our framework: smallest latency task scheduling and minimum cost task scheduling. The framework is tested by putting the edge-based, 3-D SLAM application into practice on a real testbed of ten different edge devices. Evaluations demonstrate that EdgeVision can effectively reduce processing latency, system costs, and job processing latency by up to 30% and 15% better in terms of cost saving.

**Table 2.2:** Summary of literature survey for PPC.

<b>Ref. No.</b>	<b>Title</b>	<b>Year</b>	<b>Published In</b>	<b>Summary</b>	<b>Limitations</b>
[14]	A Load Balancing Model Based on Cloud Partitioning for the Public Cloud	2013	Tsinghua Science And Technology	The PC's load balance model is based on Cloud partitioning and includes a switch mechanism to let users select various strategies depending on the load status—high, low, or normal. This model applies game theory to the load-balancing method to boost efficiency in a PC environment.	Lacks a thorough approach for Cloud partition, is ineffective in calculating the refresh period, 3. Creating a solid algorithm to determine the load degree; 4. Testing is necessary to compare various load-balancing solutions. 5. Various tests must be carried out to ensure system availability and effectiveness.
[15]	A novel approach for Dynamic Cloud Partitioning and Load Balancing in Cloud Computing Environment	2014	Journal of Theoretical and Applied Information Technology	The strategic model performs load balancing and the dynamic partition of the nodes of different Clouds. Game theory is used to develop load-balancing strategies to improve efficiency in the Cloud environment.	1. We need to increase the level of transparency. 2. Requires effective technique in updating the status report. 3. Time intervals are not very well managed; 4. Dynamic balancing technique could be made dynamic, 5. They are finding alternatives to geographical Cloud division methodology.
[44]	A Cluster-Based Load Balancing Algorithm in Cloud Computing	2014	IEEE Xplore	A distributed algorithm for load balancing in the master-slave architecture that outperforms the Closest Datacentre algorithm in terms of job distribution across the system	To evaluate the effectiveness of the proposed model in scenarios where a node belongs to more than one cluster, we believe that effective load balancing could also be achieved in this case.

				and optimal system performance	
[45]	Proactive Workload Management in Hybrid Cloud Computing+B17	2014	IEEE Transactions On Network And Service Management	One problem is addressed: service availability and reliability. Lack of SLA, privacy, and security of customer data, and 2. 4. Requirements for government compliance with regulations	Two zones have established load balancing systems, and the flash crowd zone has effective data duplication and consistency management. More effective security administration for a mixed platform
[46]	Load Degree Calculation for the Public Cloud based on the Cloud Partitioning Model using Turnaround Time	2015	International Journal of Computer Science and Information Technologies	The method for computing a node's load degree in a computer using turn-around time.	Lacks efficiency in the algorithm
[47]	A Workload-Based Approach to Partition of the Volunteer Cloud	2015	IEEE Conference on Collaboration and Internet Computing	A preliminary assessment of a job execution request distribution strategy using Cloud partitioning on the volunteer Cloud. Comparison of the suggested model's results between a Cloud with partition and a Cloud without partitions that employs the same random tasks	Improved workload classification mechanisms, More advanced algorithms, such as bio-inspired ones, Lightweight performance monitoring
[48]	Replication-based Load Balancing	2015	Transactions on Parallel and Distributed Systems	Taking the scheduling choice off the job's critical path will increase the precision of the scheduling choice. The implementation of job replication	Develop a one-attribute configuration ideal for all systems with signal propagation delay.

				follows the Schedule First and Manage Later strategy.	
[49]	Decentralized and Energy-Efficient Workload Management in Enterprise Clouds	2017	IEEE Transactions On Cloud Computing	The four issues addressed were elasticity, scalability, high operational costs, and efficient energy use. The following three algorithms are presented: Initial VM placement, partial VM migration, and complete VM migration are the three options.	1. Decentralized workload management is required inside an open-source Cloud operating system, like OpenStack; 2. can incorporate extra parameters into load-balancing formulas,
[54]	Resource Allocation Issues and Challenges in Cloud Computing	2014	International Conference on Recent Trends in Information Technology	Issues addressed include resource provisioning, job scheduling, overbooking, and resource overuse. Fourth scalability. Costing, Balancing the load, tier-two applications, 8. Accessibility, Network I/O Workload Overheads, and 10. QoS Restrictions	Is not elastic, the need to reduce expenses and maximize resource use, the requirement to provide high availability for lengthy jobs, Improved concurrent job scheduling,
[55]	Collaborative Agents for Distributed Load Management in Cloud Data Centers using Live Migration of Virtual Machines	2015	IEEE Transactions On Services Computing	The issues addressed are: 1. When to migrate the VMs, 2. Which VMs to be migrated, and 3. When to migrate, 4. When to turn on/off the hosts For these issues, a combination of CloudSim and	To help with the dynamic placement of VMs, it is necessary to construct resource utilization profiles of hosts and VMs using statistical forecasting, load balancing heuristics for initial allocation of VMs to hosts, designing VM-

				Software Modules is used.	centric management policies, and investigating the impacts of resource overselling of hosts.
--	--	--	--	---------------------------	--

## 2.4 Dual Broker-based Framework (BBF) for IoT Job Scheduling

The researchers have used ML algorithms in job scheduling in DS with proven improved efficiency and user experience results. The following section discusses a few examples of ML-based scheduling techniques in DS.

**In 2015, Sandhu and Sood** [16] Authors proposed a QoS-conscious scheduling algorithm for cloud computing. They used global and local schedulers to assign tasks to nodes and transformed data centers into virtual clusters to handle specific job categories. The proposed strategy improved QoS objective attainment, and the authors suggested applying the same tactic in fog conditions to reduce data extraction costs.

**In 2016, Deng et al.** [17] presented a problem to optimize workload distribution between fog and Cloud, balancing power usage and time constraints. They divided the issue into a primary issue and three related to different subsystems. In subsystems for FC, generalized benders deconstruction at the CC level, the Hungarian method for dispatching communication latency minimization, and convex optimization methods were used. They demonstrated how FC may improve CC performance by making minor resource sacrifices via simulations and numerical implementation. The authors' attention has only been on centralized optimization. It included communication and information exchange costs. The proposed system comprised local and global schedulers to accommodate centralized and decentralized scheduling strategies. Moreover, these scheduling mechanisms helped lower the amount of communication overhead.

**In 2017, Thapa et al.** [56] The team used Bayesian learning automata to assign jobs to the FE more accurately and efficiently. They applied a game-theoretic strategy to regulate clients' energy usage within power usage limitations set by energy providers. The proposed technique utilized learning automata to meet the power budget specified by the subnet while staying within the Nash Equilibrium (NE) point. The system categorized nodes based on available resources and job requirements to schedule resources more fairly.

**2017 Azimi et al. [57]** proposed a hierarchical computer architecture based on an IoT-based healthcare monitoring system that solves Cloud and Fog computing challenges. It uses closed-loop management to modify the platform based on an individual's health and ML-based data analytics. Despite a drawback with specific learning algorithms, the HICH model can use alternate algorithms.

**In 2018, Mahmud et al. [58]** presented a latency-sensitive policy to control the applications in FEs. Using this approach, they hoped to assure the best possible resource utilization in a foggy environment while increasing service delivery deadline QoS. For management and forwarding, they each defined two algorithms. They demonstrated the performance enhancement compared to other counterpart solutions using simulated findings. The authors' solution did not support non-deterministic latency-aware applications and run-time adjustments. Clusters built on a resource orientation can tackle this problem in their suggested structure.

**In 2018, Wan et al. [59]** built an energy-efficient load balancing and scheduling Fog-based model for an intelligent factory. The authors devised an energy consumption model to accomplish their purpose. For the best outcomes, they applied the enhanced particle swarm optimization algorithm. Finally, distributed scheduling of manufacturing clusters was accomplished using a multi-agent system. Simulation results in the targeted area demonstrated the best scheduling and load balancing.

**2018 Soualhia et al. [60]** offered a Hadoop scheduling framework that used data from a Cloud environment to organize tasks dynamically. The framework relies on policies generated by the Markovian decision-making process and the predictions provided by machine learning algorithms. Unlike the traditional static heartbeat-based fault diagnosis Hadoop uses, this framework uses an adaptive failure-aware Hadoop scheduler named ATLAS+. This scheduler can dynamically identify job tracker faults. The results of the experiments demonstrate that ATLAS+ significantly reduced the number of failed jobs, JET, CPU, and memory utilization. The supervised training process for the prediction models was the only area on which the authors of this study concentrated.

**In 2018, Zhu et al. [61]** proposed a Q-learning-based transmission scheduling system that can perform best when broadcasting packets across multiple channels. They used a Markov decision



process to create a model that depicted the evolution of the system's states. The relay was assisted in choosing the optimum course of action by applying the q-learning algorithm, a reinforcement learning technique. The authors used multilayer auto-encoder deep learning algorithms to map states and related activities. Studies showed that this proposed model could transport packets while consuming less power than competing methods. Although the scheduling approach in this case is straightforward, it serves only one relay.

**In 2018, Wei et al.** [62] proposed a job scheduling framework that considers QoS. A vital feature of this approach is its job scheduler, which uses deep reinforcement learning to select the job-VM mapping for online requests. The job scheduler used their expertise to make decisions; no prior information was required. Testing showed that the recommended framework handled different load scenarios assured QoS achievement, and decreased the average job response time. The approach proposed by the authors must be altered to handle the complex Cloud environments. Additional factors like VM failures, elastic resource supply, and inter-job interactions had to be considered using this technique. Through the division of scheduling into two layers, the proposed approach increases flexibility.

**In 2019, Chen et al.** [63] evaluated the effectiveness of road safety apps using perception-action time (PRT) in a study conducted by the authors of PRT. They used a virtualization strategy called Fog and networking technology based on information to reduce the PRT. A deep reinforcement learning approach was utilized to develop an online work scheduling system with the most efficient resource allocation. The experimental results were compared to traditional methods, and it was demonstrated that there was a significant reduction in PRT.

**In 2019, Zhu et al.** [64] implemented a design for Quality of Service (QoS) assurance that considered high data transfer rates, extensive connectivity, and exceptionally low latency. Standard programming techniques were found to be inadequate for making scheduling decisions due to increased complexity, dynamic network behavior, and a lack of quantifiable correlations between network events. They used the decision tree method to test the proposed architecture and found that it could accurately and independently predict upcoming QoS-related abnormalities.

**In 2019, Henri and Lu** [65] utilized random forests, support vector machines, and neural networks for scheduling and predicting logistic regression. This framework was developed for integrated PV and battery energy storage systems that model-based controllers controlled. The research suggested using supervised ML to plan and predict real-time processes. ML techniques enhanced the efficiency of model-based control mechanisms while lowering the computational cost, as shown by simulation results. Enhancing the deterioration model and considering the non-linear charging and discharging patterns is essential.

**In 2019, Zhao et al.** [66] conducted a survey that combined software-defined networking (SDN) architecture, machine learning (ML), and artificial intelligence (AI). They discussed potential advancements in ML algorithms and SDN architecture to create more intelligent, active, and personalized networking models.

**In 2019, Amiri and Gündüz** [67] discussed assigning duties to various workers through a master. Their research sought to illustrate how computing load affected overall completion time. Utilizing the jobs assigned to each worker and their execution schedule, they suggested two execution scheduling systems. Compared to existing systems, experimental findings obtained on an Amazon EC2 cluster have demonstrated a discernible reduction in overall completion time and improved efficiency.

**Table 2.3:** Summary of literature survey for dual BBF using ML techniques and QoS parameters.

Ref. No.	Title	Year	Published In	Summary	Limitations
[16]	Scheduling of big data applications on distributed Cloud based on QoS parameters	2015	Scheduling of big data applications on distributed Cloud based on QoS parameters	A two-layer scheduling structure and a QoS-conscious scheduling method. They employed two schedulers: a global scheduler and a local scheduler.	1. Missing brokering, 2. Missing hierarchical distribution of load balancing jobs, 3.No ML technique is used, 4.No clustering of jobs and resources

[17]	Optimal Workload Allocation in Fog-Cloud Computing Toward Balanced Delay and Power Consumption	2018	IEEE Internet of Things	Maintaining a trade-off between power and time consumption is challenging while optimally distributing the workload across Fog and Cloud.	1. Missing brokering, 2. Missing hierarchical distribution of load balancing jobs, 3. QoS parameters not considered, 4.No ML technique is used, 5.No clustering of jobs and resources
[56]	A Learning Automaton-Based Scheme for Scheduling Domestic Shiftable Loads in Smart Grids	2018	IEEE Access	A game-theoretic method increases the efficiency and precision of job requests sent to the FE for a Smart Grid subnet on a local area network with a single power source and numerous users.	1. Missing brokering, 2. Missing hierarchical distribution of load balancing jobs, 3. QoS parameters not considered, 4.No ML technique is used, 5.No clustering of jobs and resources
[57]	HiCH: Hierarchical Fog-Assisted Computing Architecture for Healthcare IoT	2018	ACM Transactions on Embedded Computing Systems	A computational architecture for dealing with many operational problems connected to Clouds and Fog, such as accessibility, promptness, dependability, accuracy, and adaptability	1. Missing brokering, 2. Missing hierarchical distribution of load balancing jobs, 3. No ML technique is used, 4.No clustering of jobs and resources
[58]	Latency-Aware Application Module Management for Fog Computing Environments	2018	ACM Transactions on Internet Technology	FEs must follow a latency-sensitive policy when managing the application modules.	1. Missing brokering, 2. Missing hierarchical distribution of load balancing jobs, 3. QoS parameters not considered, 4.No ML technique is used, 5.No clustering of jobs and resources
[59]	Fog Computing for Energy-Aware Load Balancing and	2018	IEEE Transactions On Industrial	An energy-efficient load balancing as well as scheduling Fog-based	1. Missing brokering, 2. Missing hierarchical distribution of load balancing jobs, 3. QoS

	Scheduling in Smart Factory		Informatics	approach for a smart factory	parameters not considered, 4.No ML technique is used, 5.No clustering of jobs and resources
[60]	A Dynamic and Failure-Aware Task Scheduling Framework for Hadoop	2018	IEEE Transactions on Cloud Computing	A system that is associated with both the Hadoop scheduler and uses the information gathered to organize the jobs for a Cloud environment dynamically	1. Missing brokering, 2. Missing hierarchical distribution of load balancing tasks, 3.QoS parameters not considered, 4.No clustering of jobs and resources
[61]	A New Deep-Q-Learning-Based Transmission Scheduling Mechanism for the Cognitive Internet of Things	2018	IEEE Internet of Things Journal	A technique for scheduling transmissions based on Q learning ensured the highest throughput when sending packets via several channels.	1. Missing brokering, 2. Missing hierarchical distribution of load balancing tasks, 3. QoS parameters not considered, 4. No clustering of jobs and resources
[62]	DRL-Scheduling: An Intelligent QoS-Aware Job Scheduling Framework for Applications in Clouds	2018	IEEE Access	A framework for managing jobs using QoS. The framework's main component was an algorithm for scheduling jobs based on deep reinforcement learning.	1. Missing brokering, 2. Missing hierarchical distribution of load balancing tasks, 3. No clustering of jobs and resources
[63]	A machine-learning-based time-constrained resource allocation scheme for vehicular Fog computing	2018	China Communications	A metric to assess the effectiveness of apps for improving road safety that combines information-based networking technologies with a Fog virtualization strategy to cut down on PRT authors.	1. Missing brokering, 2. Missing hierarchical distribution of load balancing tasks, 3. QoS parameters not considered, 4.No ML technique is used, 5.No clustering of jobs and resources

[64]	A Supervised Learning Based QoS Assurance Architecture for 5G Networks	2018	IEEE Access	A QoS assurance structure that takes into account widespread connectivity, extremely low latency, and high-speed data transfer rates as QoS parameters	1. Missing brokering, 2. Missing hierarchical distribution of load balancing tasks, 3. No ML technique is used, 5.No clustering of jobs and resources
[65]	A Supervised Machine Learning Approach to Control Energy Storage Devices	2019	IEEE Transactions on Smart Grid	A methodology utilizing logistic regression, support vector machines, neural networks, and random forest algorithms for forecasting and scheduling	1. Missing brokering, 2. Missing hierarchical distribution of load balancing tasks, 3. QoS parameters not considered, 4. No clustering of jobs and resources
[66]	A Survey of Networking Applications Applying the Software-Defined Networking Concept Based on Machine Learning	2019	IEEE Access	A study combining software-defined networking (SDN) architecture, machine learning, and artificial intelligence	1. Missing brokering, 2. Missing hierarchical distribution of load balancing tasks, 3. QoS parameters not considered, 4. No clustering of jobs and resources
[67]	Computation Scheduling for Distributed Machine Learning With Straggling Workers	2019	IEEE Transactions on Signal Processing	Showed the cumulative completion time as a function of computational load through the master-controlled allocation of tasks among numerous workers.	1. Missing brokering, 2. Missing hierarchical distribution of load balancing tasks, 3. QoS parameters not considered, 4.No ML technique is used, 5.No clustering of jobs and resources

## 2.5 Multilevel Broker-based Framework for Improved Geographic Coverage of Fog Resources

This section discusses the existing solutions proposed by the researchers for job scheduling in FC environments using ML techniques and geographic coverage.

**In 2016, Zeng et al.** [68] created a non-linear, mixed-integer programming problem to increase work scheduling and resource management effectiveness, focusing on reducing job completion time. They concentrated on balancing the workload, I/O interruptions, and putting job images on the storage servers as their three main concerns in this context. The authors provided a computation-efficient solution to this problem's high level of complexity and carried out comprehensive simulation-based research to confirm it.

**In 2017, Sthapit et al.** [69] discovered ways to offload the computationally demanding algorithms without Clouds or Fog. They suggested a different approach using a network of queues and linear programming to make scheduling decisions. Their suggested algorithm might increase the system's efficiency but at an additional expense. They suggested a revolutionary algorithm that, while consuming a little more energy, would increase the system's overall effectiveness. Whenever the job request flow is high, the proactive centralized strategy provides the optimum performance and energy compromise, whereas the reactive scattered approach generates successful outcomes whenever the job request flow is low.

**In 2017, Rahbari et al.** [70] developed a hyper-heuristic, security-aware technique based on data mining to schedule jobs on FNs. The authors have considered three factors, including authentication, integrity, and confidentiality, to preserve the security of Fog devices. They created an objective function considering BW, CPU, and security overheads. The results of the experiments revealed a considerable reduction in cost and average energy consumption compared to counterpart heuristic algorithms like Particle Swarm Optimization, Ant Colony Optimization, and Simulated Annealing.

**In 2017, Chen et al.** [71] devised an architecture for task classification and resource scheduling for information-centric IoT applications in 2017. The architecture enabled optimal dispatching of distributed resources at the least expensive rate, facilitating the achievement of the application's QoS in a Cloud-FC context. The proposed solution also effectively reduced the load on CC by distributing it between FEs and the Cloud.

**In 2018, Yang et al.** [72] developed the Delay and Energy-Balanced Task Scheduling (DEBTS) algorithm to balance performance indicators efficiently. They recommended an effective

control parameter to explain the tradeoff connection during the runtime procedure of task scheduling. Simulations have shown that DEBTS may produce task scheduling outcomes with significantly better delay-energy performance.

**In 2018, Yin et al.** [73] created a reallocation technique and a task scheduling model that considered containers' roles, ensuring FNs would have fewer delays and better concurrency. The authors created an algorithm to accomplish the goals using container properties. Through simulated trials, they showed improved results regarding the reduced delay and better concurrency.

**In 2019, Casquero et al.** [74] proposed a multi-agent system (MAS) and an application orchestrator to create a solution for Fog-in-the-loop applications. The authors devised a bespoke scheduler built on the MAS to divide the scheduling duties among the available FNs. The authors found during the experiment that the performance of the proposed method compared to that of the K8s default scheduler.

**In 2019, Zhang et al.** [75] devised a broad analytics-based task scheduling model that categorizes FNs into two groups: task nodes (TN) and voluntary nodes (VN). They suggested an offloading approach that would enable VNs to assist in completing duties at their neighboring TNs with the most minor delay. The simulation results demonstrated that the suggested technique can effectively provide the ideal set of helper nodes to neighboring nodes, minimizing the overall job processing delay. VN produced better results in voluntary mode than offloading in command mode.

**In 2019, Zhu et al.** [76] proposed a novel approach for task distribution in vehicle fuel cells using binary particle swarm optimization and optimization based on linear programming. Their method suggested a dynamic, event-triggered paradigm to minimize quality loss and reduce average latency. As per the results, the proposed plan reduced quality loss by up to 56% and average latency by up to 27%, making it a latency-sensitive and quality-optimized scheduling solution for vehicle fuel cells.

**In 2019, Abedin et al.** [77] solved a load balancing issue for narrowband IoT (NB-IoT) in Fog computing (FC) networks using a network of queues and a bankruptcy game model. They

scheduled uplinks with a Shapley value-based policy and introduced a less complex method, GITS. They balanced the load using the Hitchcock-Koopman transportation problem.

**In 2019, Sthapit et al.** [78] offered a method for mobile app offloading in the absence of Cloud or Fog. Their approach utilizes a network of queues and linear programming to facilitate scheduling decisions. After conducting simulated experiments on various centralized and distributed algorithms, it was discovered that the system's overall effectiveness may be improved by increasing energy consumption.

**In 2019, Stavrinides and Karatza** [79] suggested a method for scheduling that focuses on the efficient use of Cloud resources for real-time applications in FEs. The authors studied many aspects influencing these parameters to develop the suggested approach based on the compromise between performance and cost. They compared their job scheduling heuristic to a baseline strategy utilizing Fog resources under various amounts of workflow input data.

**In 2019, Benblidia et al.** [80] introduced a linguistic and fuzzy quantified approach for ranking FNs based on their level of satisfaction. The research team collected user preferences and FN features and utilized two parameters, namely the least satisfactory proportion (LSP) and the most significant satisfactory proportion (GSP), to differentiate the similarities in their proposed method. Through the results of their experiments, they demonstrated higher user preference satisfaction. Additionally, their method has produced a compromise between the execution delay of the parameter, average user satisfaction, and energy usage.

**Table 2.4:** Summary of literature survey for multilevel BBF for better geographic coverage of Fog resources.

Ref No.	Title	Year	Published In	Summary	Limitations
[68]	Joint Optimization of Task Scheduling and Image Placement in Fog Computing Supported Software-Defined	2016	IEEE Transactions on Computers	An issue with mixed-integer non-linear programming to increase job scheduling and resource management effectiveness while	1. Missing brokering, 2. Missing hierarchical distribution of load balancing tasks, 3. QoS parameters not considered, 4.No ML technique is used, 5.No clustering of jobs and resources



	Embedded System			reducing task completion time	
[69]	Distributed computational load balancing for real-time applications	2017	2017 25th European Signal Processing Conference (EUSIPCO)	A solution in which scheduling decisions are made by linear programming and a network of queues	1. Missing brokering, 2. Missing hierarchical distribution of load balancing tasks, 3. QoS parameters not considered, 4. No clustering of jobs and resources
[70]	Security-aware scheduling in Fog computing by the hyper-heuristic algorithm	2017	2017 3rd Iranian Conference on Intelligent Systems and Signal Processing (ICSPIS)	A data mining-based security-aware hyper-heuristic algorithm to schedule the jobs on Fog devices	1. Missing brokering, 2. Missing hierarchical distribution of load balancing tasks, 3. QoS parameters not considered, 4.No ML technique is used, 5.No clustering of jobs and resources
[71]	Cloud-Fog computing for information-centric Internet-of-Things applications	2017	2017 International Conference on Applied System Innovation (ICASI)	Using task categorization and resource scheduling features, a scheduling structure for information-centric IoT applications	1. Missing brokering, 2. Missing hierarchical distribution of load balancing tasks, 3. QoS parameters not considered, 4.No ML technique is used
[72]	DEBTS: Delay Energy Balanced Task Scheduling in Homogeneous Fog Networks	2018	IEEE Internet of Things Journal	An energy and delay-neutral task scheduling (DEBTS) method balances the performance indicators efficiently.	1. Missing brokering, 2. Missing hierarchical distribution of load balancing tasks, 3. QoS parameters not considered, 4.No ML technique is used, 5.No clustering of jobs and resources
[73]	Tasks Scheduling and Resource Allocation in Fog Computing Based on	2018	IEEE Transactions on Industrial Informatics	A task scheduling model and a reallocation mechanism that considered the role of containers	1. Missing brokering, 2. Missing hierarchical distribution of load balancing tasks, 3. No ML technique is used,

	Containers for Smart Manufacturing			ensured reduced delays as well as improved concurrency for FNs	4.No clustering of jobs and resources
[74]	Distributed scheduling in Kubernetes based on MAS for Fog-in-the-loop applications	2019	IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)	A system for Fog-in-the-loop applications by integrating a model multi-agent system (MAS) with a containerized application orchestrator	1. Missing brokering, 2. Missing hierarchical distribution of load balancing tasks, 3. QoS parameters not considered, 4.No ML technique is used, 5.No clustering of jobs and resources
[75]	DOTS: Delay-Optimal Task Scheduling Among Voluntary Nodes in Fog Networks	2019	IEEE Internet of Things Journal	A general analytical model of task scheduling that divides FNs into two categories, which are voluntary nodes (VN) and task nodes (TN)	1. Missing brokering, 2. Missing hierarchical distribution of load balancing tasks, 3. QoS parameters not considered, 4.No ML technique is used
[76]	Folo: Latency and Quality Optimized Task Allocation in Vehicular Fog Computing	2019	IEEE Internet of Things Journal	A dynamic, event-triggered framework for task allocation using linear programming-based optimization and binary particle swarm optimization	1. Missing brokering, 2. Missing hierarchical distribution of load balancing tasks, 3. QoS parameters not considered, 4.No ML technique is used, 5.No clustering of jobs and resources
[77]	Fog Load Balancing for Massive Machine Type Communications: A Game and Transport Theoretic Approach	2019	IEEE Access	A Fog load balancing problem for narrowband IoT (NB-IoT) technology, aiming to minimize the cost of load balancing in Fog computing networks	1. Missing brokering, 2. Missing hierarchical distribution of load balancing tasks, 3. No ML technique is used, 4.No clustering of jobs and resources

[78]	Computational Load Balancing on the Edge in Absence of Cloud and Fog	2019	IEEE Transactions on Mobile Computing	A solution to offload the applications from mobile devices when Cloud or Fog was not available	1. Missing brokering, 2. Missing hierarchical distribution of load balancing tasks, 3. QoS parameters not considered, 4.No ML technique is used, 5.No clustering of jobs and resources
[79]	Cost-Effective Utilization of Complementary Cloud Resources for the Scheduling of Real-Time Workflow Applications in a Fog Environment	2019	2019 7th International Conference on Future Internet of Things and Cloud (FiCloud)	A scheduling approach for real-time workflow applications in FEs, which focused on the utilization of complementary Cloud resources at reduced costs	1. Missing brokering, 2. Missing hierarchical distribution of load balancing tasks, 3. No ML technique is used, 4.No clustering of jobs and resources
[80]	Ranking Fog Nodes for Tasks Scheduling in Fog-Cloud Environments: A Fuzzy Logic Approach	2019	2019 15th International Wireless Communications & Mobile Computing Conference (IWCMC)	A linguistic and fuzzy quantified proposition for ranking the FNs from most satisfactory to least satisfactory ranking	1. Missing brokering, 2. Missing hierarchical distribution of load balancing tasks, 3. QoS parameters not considered, 4. No clustering of jobs and resources

## 2.6 Summary

As seen in this chapter, job scheduling is being researched extensively. It is crucial to thoughtfully, effectively, and promptly assign jobs to the appropriate FNs to obtain the optimal benefits of FC. Our work mainly focuses on improving the performance of CC and FC environments by efficient load balancing and job scheduling.

This thesis recommends combining centralized and decentralized load control mechanisms with the autonomous behavior of load-balancing entities using SAs. The aim is to obtain the advantages of PC resources while reducing the challenges of managing large volumes of

computing nodes. After going through the existing literature, to our knowledge, none of the researchers have combined the concept of SAs and hierarchical brokers.

Further segregating the Fog resources as available and mapping the jobs to the most suitable set of resources further improves their performance and user satisfaction. To our knowledge, none of the researchers have proposed the concept of resource segregation and the implementation of brokers in a Fog environment. A combination of methodologies such as Naïve Bayes and SOM for selecting an FE and FN through a hierarchical arrangement of brokers is another novelty of our work.

This thesis takes care of the following points while implementing the proposed solutions:

1. Reduced search space in PC
2. Utilizing the concept of brokers-based service orchestration
3. Autonomous behavior of the load balancing and job scheduling components
4. ML-based classification of resources and mapping of IoT application jobs
5. Reduce the congestion on Job scheduling components
6. Global geographical coverage of Fog resources

To our knowledge, none of the existing job scheduling solutions address all these problems together. The novelty of our solutions is the selection of performance parameters, the combination of multiple technologies, and their sequencing in the implemented experimental setups to achieve the desired results.

# **CHAPTER 3**

## **PARTITIONED PUBLIC CLOUD**

### **3.1 Introduction**

CC empowers resource flexibility with minimal upfront costs, facilitating scalability and cost-effectiveness [48] [81]. According to CISCO, Cloud traffic will reach 14.1 ZB annually soon [82]. The PC is a shared space for users to outsource their jobs [83], where Load-balancing approaches are required to handle under loaded and overloaded resources at the available servers [84]–[87]. Cloud Partitioning is one of the commonly used methods for load balancing in the PC [46], [49], [53]. This chapter discusses a software agent-based load-balancing framework for partitioned public Cloud.

### **3.2 Background**

The following sections introduce the concepts of PC, PPC, and SA, which are used later in this chapter to implement the software agent-based load-balancing framework for partitioned public Cloud.

#### **3.2.1 Public Cloud (PC)**

PC describes a category of online computer services provided by independent vendors, offering the feature of pay-as-you-go to access computing resources, including VNs, storage, processing power, cutting-edge technologies like artificial intelligence and ML, and hosting websites and applications for individuals and businesses. Well-known PC providers are Google Cloud Platform (GCP), Microsoft Azure, and Amazon Web Services (AWS). Key features of a PC are its accessibility, scalability, multi-tenancy, cost benefits, and well-managed services [14], [15].

#### **3.2.2 Partitioned of Public Cloud (PPC)**

A PC has nodes spread across various geographic locations [14], causing challenges such as complex management, resource allocation, optimization, cost management, data management, and latency management. Figure 3.1 compares various Cloud types and justifies the existence of these issues in the PC environment.

Private Cloud	Hybrid Cloud	Public Cloud
<ul style="list-style-type: none"> <li>• Small Scale</li> <li>• Limited Resources and Services</li> <li>• Dedicated to a single organization</li> <li>• On-Premises or private data center</li> </ul>	<ul style="list-style-type: none"> <li>• Medium Scale</li> <li>• Combination of public and private Cloud environments</li> <li>• Flexible, offers resource allocation and scalability</li> </ul>	<ul style="list-style-type: none"> <li>• Large Scale</li> <li>• Vast Resources and Services</li> <li>• Shared by multiple users globally</li> <li>• Scalable, pay-as-you-go model</li> </ul>

Figure 3.1: Size-wise comparison of Cloud types.

To simplify it, a PC is geographically partitioned to manage its massive size, where each partition is handled separately in the load-balancing process. Additionally, the partitions are dynamic; that is, partitions can be added or removed depending on the status report maintained by the Partition Manager and Controller Broker [15].

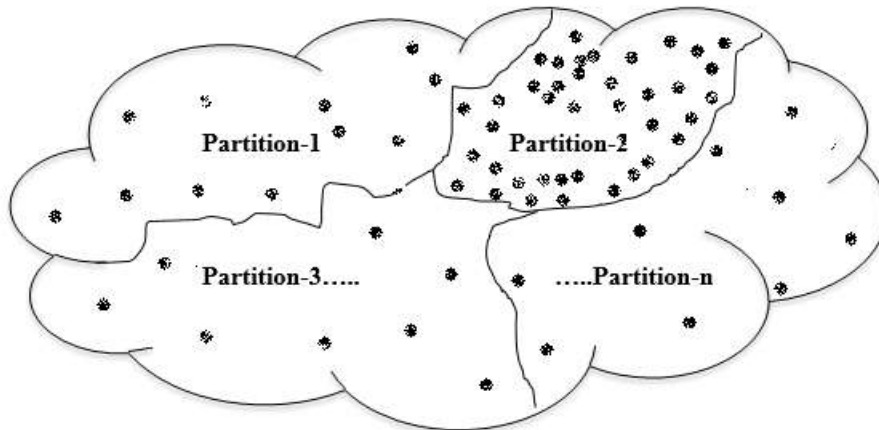


Figure 3.2: Geographically partitioned public Cloud.

A Cloud can be divided into  $n$  number of partitions, each with a set of  $m$  number of nodes, containing the physical resources covered under that partition. Once the framework has partitioned the PC geographically, each partition manages its resources, clients, and requests separately and independently, reducing search space and resolving PC management issues.

### 3.2.3 Software Agents (SA)

Intelligent agents, or SAs, are self-governing, goal-oriented entities that can act on behalf of entities like users or other systems. These agents work in a given environment, collecting data, coming to conclusions, and acting to accomplish their goals [88]. An SA's behavior is contingent upon its mental state and the conditions of its surroundings [89]. SAs can communicate with other agents by sending and receiving messages [88]. Primarily, the self-adaptive behavior of SAs makes them usable in load balancing. An SA-based system can manage changes dynamically while maintaining reliability [90]–[92].

The use of SAs in load balancing framework adds the following capabilities to it:

- the capacity to *detect*, *diagnose*, and *assess* shifts in the operating environment
- the capacity to *evaluate* its behavior based on environmental shifts;
- the capacity to *modify* its behavior in response to new shifts and
- the capacity to *change* its internal and external behavior intentionally and automatically

## 3.3 Proposed Load Balancing Framework in Partitioned Public Cloud

This section proposes a software-agent-based framework to perform the load balancing within and across the geographically partitioned public Cloud, intended to improve the MET, TET, and AET. Compared to traditional, primary load-balancing methodologies such as FCFS and SJF [93], [94]. The proposed framework determines if applying SAs as brokers can improve the performance parameters in a PPC. The framework accepts the client requests as input and allocates them to the suitable node, which can optimally improve the performance parameters.

### 3.3.1 Architecture

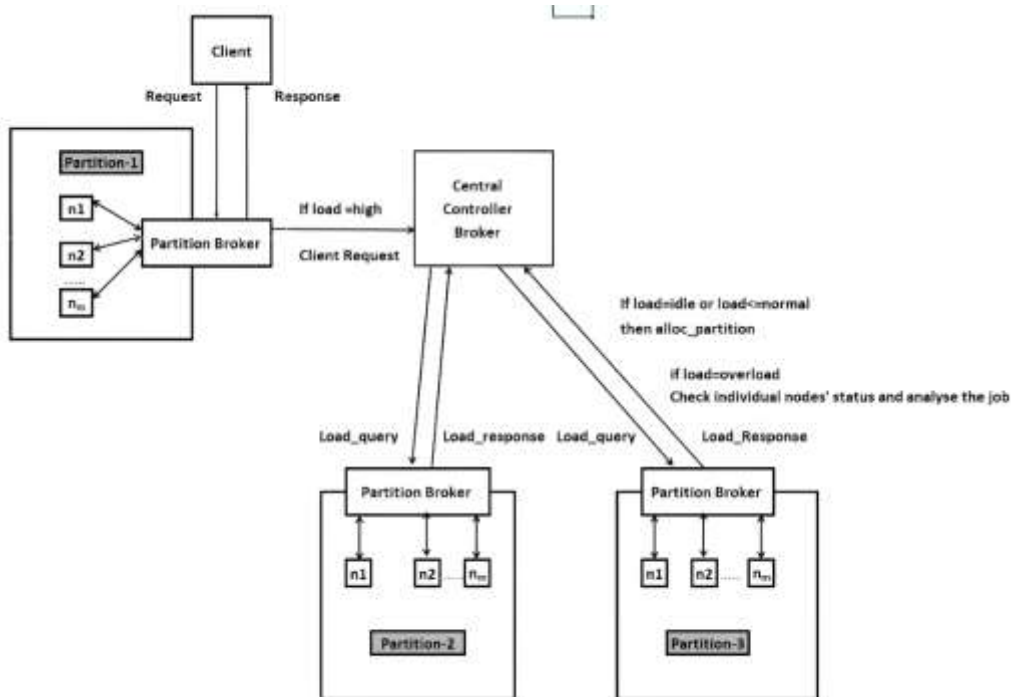
The brokers in this architecture are practical representations of the SAs:

- Client Agent                           => Client
- Partition Agent                       => Partition Broker (PB)
- Controller Agent                      => Controller Broker (CCB)

The major components in the architecture are:

- **Client:** The primary duty of the client is to submit requests, along with the required set of resources, to a Partition Broker (*PB*).

- **PB:** Each partition has a local PB to receive the client requests. PB records the load status of all the nodes inside its partition and shares these details with the Central Controller Broker (CCB) whenever required.
- **CCB:** CCB interacts with all the PBs to collect the nodes' load status. CCB is invoked for load balancing across the partitions.
- **Node:** Nodes contain the physical resources, such as storage, computing capacity, graphical resources, memory, and other such resources, to be utilized for processing user requests. Each partition is assumed to have m nodes ( $n_1, n_2, n_3... n_m$ ).



**Figure 3.3:** Architecture: Framework for load balancing in PPC.

### 3.3.2 Application job distribution across Cloud and Fog Layer

Fog offers various data services in an IoT setup, including filtering, segregating, aggregating, data encryption, and caching [95]–[99]. An FE keeps all job requests that are latency-sensitive. However, a Cloud system can manage infrequent, time-consuming, and resource-intensive jobs. The proposed framework performs load balancing on a Cloud.



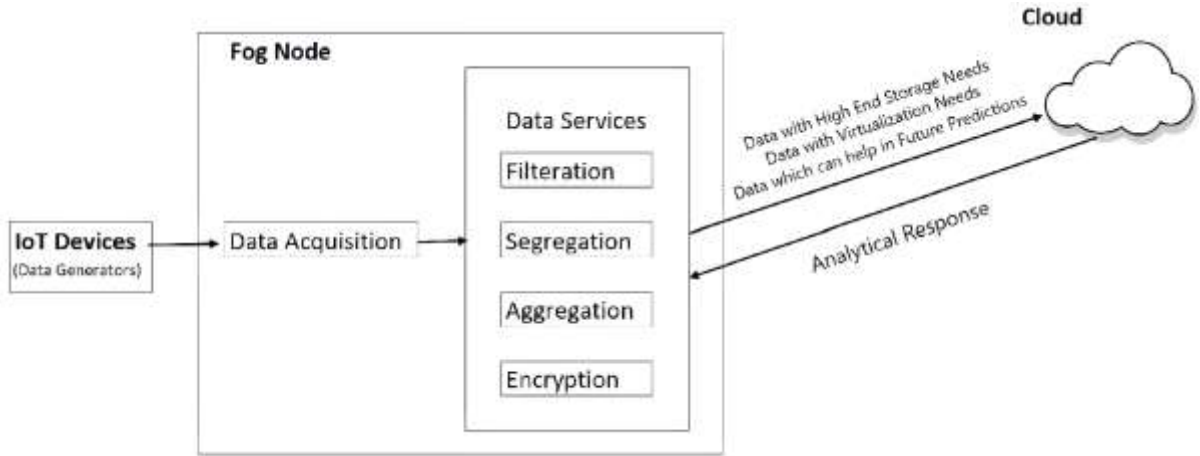


Figure 3.4: Job Segregation: Cloud vs Fog.

### 3.3.3 Assessing the Load Status of Partitions and Nodes

The number of jobs allocated to a node at a given time is known as its workload. The load status of the Cloud includes the load status of partitions and their nodes and is used to determine the availability of resources for job allocation. The system needs to assess the availability at two levels listed below:

- Choosing the  $P_i$  partition in which a node can be searched.
- Finding a node  $N_j$  to which a job can be allocated.

The proposed framework defines four load states, which are:

- idle,
- normal,
- overloaded and
- full

These states are determined based on the static threshold values pre-defined for the partitions by the CCB and the nodes by the PBs. At any given instance of time, the load state of the partition depends upon the load state of its nodes. Assuming that there are  $i$  number of partitions,  $j$  number of nodes, given below is the formulation of these load states:

#### 1. Load State='idle'

- Node State:

A  $j^{th}$  node  $N_j$  is considered to be in an idle state if resources available with  $j^{th}$  node  $R_{n,j}$  are beyond a predefined threshold  $t_{n,idle}$ . i.e.

$$N_j = \text{'idle'} \text{ IF } R_{n,j} > t_{n,idle} \quad \text{Eq. 3.1}$$

b) Partition State:

An  $i^{th}$  partition  $P_i$  is considered to be in idle state if the number of idle nodes in that partition are beyond a predefined threshold  $t_{p,idle}$ , i.e.,

$$P_i = \text{'idle'} \text{ IF } C_i > t_{p,idle} \quad \text{Eq. 3.2}$$

where  $C_i$  is no. of idle nodes in a partition.

**2. Load State='normal'**

a) Node State:

A  $j^{th}$  node  $N_j$  is considered to be in a normal state, if resources available with  $j^{th}$  node  $R_{n,j}$  are beyond a predefined threshold  $t_{n,normal}$ . i.e.

$$N_j = \text{'normal'} \text{ IF } R_{n,j} > t_{n,normal} \quad \text{Eq. 3.3}$$

b) Partition State:

An  $i^{th}$  partition  $P_i$  is considered to be in a normal state if the number of normal nodes in that partition are beyond a predefined threshold  $t_{p,normal}$ , i.e.

$$P_i = \text{'normal'} \text{ IF } C_i > t_{p,normal} \quad \text{Eq. 3.4}$$

where  $C_i$  is no. of normal nodes in a partition.

**3. Load State='overload'**

a) Node State:

A  $j^{th}$  node  $N_j$  is considered to be in an overloaded state, if resources available with  $j^{th}$  node  $R_{n,j}$  are beyond a predefined threshold  $t_{n,ovld}$ . i.e.

$$N_j = \text{'overloaded'} \text{ IF } R_{n,j} > t_{n,ovld} \quad \text{Eq. 3.5}$$

b) Partition State:

An  $i^{th}$  partition  $P_i$  is considered to be in an overloaded state if the number of overloaded nodes in that partition are beyond a predefined threshold  $t_{p,ovld}$  i.e.

$$P_i = \text{'overloaded'} \text{ IF } C_i > t_{p,ovld} \quad \text{Eq. 3.6}$$

where  $C_i$  is the number of ovld nodes in a partition.

**4. Load State='full'**

An  $i^{th}$  partition  $P_i$  is considered to be in full load state if the number of overloaded nodes  $C_i$  in that partition is equal to  $j$ , i.e.,

$$P_i = \text{'full'} \text{ IF } C_i = j \quad \text{Eq. 3.7}$$

where  $j$  is the total number of nodes in that partition.

### 3.3.4 Job Assignment

Following this load status review, three outcomes are possible:

- **Case 1:** If a single node  $N_j$  is available with an idle or normal state, then allocate the job to  $N_j$  and update the load status of the partition ' $S_{p,i}$ ' and the resources at node  $R_{n,j}$
- **Case 2:** If more than one node is available with an idle or normal state, then make a list of all these nodes as  $N_{capable[]}$
- **Case 3:** If No node is available with sufficient resources, PB transfers the request to the CCB to search for a node in other partitions.

The CCB calls all other partition's PBs and adds the partition's ID to a list of  $P_{capable[]}$ , capable partitions whenever it discovers a  $P_{normal}$  or  $P_{idle}$  load state. A similar process is followed to search a node  $N_j$  in all the partitions in  $P_{capable[]}$ .

### 3.3.5 Complexity Analysis

Time complexity varies according to the stages of the process. When the job was initially submitted, just one node's availability status was assessed. Consequently, the complexity of this phase  $O(1)$ . The complexity class  $O(N)$  is applicable in all other scenarios, and the value of  $N$  varies according to the number of nodes in a partition and their RAV. Table 3.1 compiles the complexity for different possible scenarios.  $P$  denotes a Partition, and  $N$  is a Node in Table 3.1. The number of partitions and nodes within each partition impacts the overall complexity.

**Table 3.1:** Complexity Analysis

Sr. No.	Step	Time Complexity
1.	Checking the current node for the free resources	$O(1)$
2.	Checking other nodes in the current partition	$O(N_p)$ : $N_p$ is the number of nodes in the partition $P$
3.	Checking other partitions	
	a) Every partition has the same number of nodes	$O(P*N_p)$
	b) Partitions have different number of nodes	$O(N_{p1} + N_{p2} + \dots)$
4.	Checking across multiple partitions	$O(\text{Count}(P_{capable}))$
5.	Checking across multiple nodes	$O(\text{Count}(P_{capable}(N)))$

### 3.4 Experimental Results and Discussion

This section presents experimental results and their discussion. This technique is implemented in CloudSim.

The AET, MET, and TET are compared with FCFS and SJF algorithms within the same setup to assess the performance of the proposed framework. Each algorithm is run five times as part of the experiment. Table 3.3 details the experimental setup with the heterogeneous sizes of VMs and jobs.

**Table 3.2:** Experimental setup

Sr. No.	Parameter Name	Values
1.	Partitions	2
2.	Brokers	2
3.	Datacenters	4 (2 Datacenters/partition)
4.	Host	2 hosts for each Datacenter
5.	VMs	6 VMs in each Datacenter
6.	Tasks	40 tasks with each broker

All these parameters are considered to be the basic evaluation parameters because the users of DS, such as Cloud or Fog, are very particular towards the time-based parameters. The execution and response time of the application jobs are significant and helpful in assessing a site's performance. Other parameters, such as accuracy, reliability, and QoSs, are considered to be taken care of by default. However, time-oriented parameters require experimentation to devise and follow load-balancing schemes that offer better results in terms of such parameters. According to the results, the proposed framework performs better than the FCFS algorithm, while the SJF algorithm performs well overall. Figure 3.5 compares MET [100] and [101].



**Figure 3.5:** MET Comparison

Figures 3.6 and 3.7 compare AET and TET.

	Total Finish Time (millisec)	Total Waiting Time (millisec)	Total Actual Run Time (millisec)
Proposed Algorithm	93834.15	56615.47	34072.09
FCFS	160563.31	107551.58	46196.64
SJF	39182.74	11040.08	28255.36



Figure 3.6: TET Comparison

	Average Finish Time (millisec)	Average Waiting Time (millisec)	Average Actual Run Time (millisec)
Proposed Algorithm	1839.89	1110.11	668.08
FCFS	2352.28	1605.14	695.51
SJF	979.57	276.00	706.88



Figure 3.7: AET Comparison

Table 3.3: Comparison with other approaches.

Sr. No.	Title	Goal	Parameters
1	[93]	Optimal Scheduling Approach	Cost, degree of imbalance, makespan, and throughput
2	[94]	Comparison between space-shared scheduling policy and time-shared scheduling policy to find the difference between the actual execution time of tasks	The ratio of actual and estimated execution time
3	Proposed Framework	Load balancing in PPC using SAs	AET, MET, and TET

### **3.5 Summary**

The proposed framework implements a Software Agent-based load-balancing framework for partitioned public Cloud. Experimental results show a significant decline in the time-based performance parameters such as AET, MET, and TET of the workload at a given time. The proposed framework suggests the preliminary segregation of the jobs between the CC and FC by serving all the latency-sensitive and frequent jobs at the Fog and all the resource-demanding, infrequent, and large-sized job requests at the Cloud, reducing the load at the Cloud end. All the jobs chosen for execution in the Cloud are allocated to the nodes based on those nodes' existing load status. An overloaded state significantly reduces the number of unserved jobs by searching the available Nodes, even in overloaded partitions.

The next Chapter presents the implementation of a combination of Naïve Bayes and SOM ML techniques in a dual-broker-based job scheduling framework that integrates the QoS parameters to improve the performance of an FC environment.

# CHAPTER 4

## DUAL BROKER-BASED JOB SCHEDULING

### 4.1 Introduction

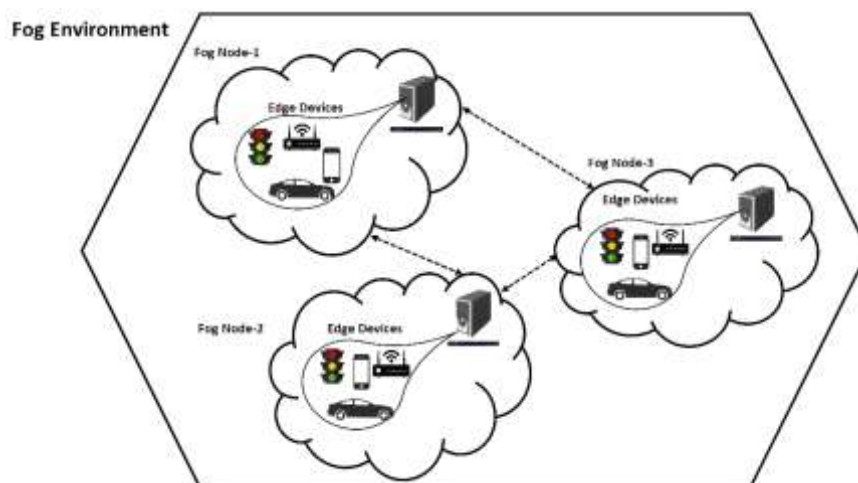
As seen in the previous chapter, efficient resource allocation to the requested jobs significantly improves the performance of DSs. Modern domains are being researched and implemented in job scheduling, including classification techniques in ML. ML classification is a versatile and powerful tool that enables automation, pattern recognition, and predictive modeling across various applications and industries. It helps extract valuable insights from data, improve decision-making processes, and enhance efficiency. The current chapter discusses an ML-based dual broker framework integrating QoS parameters for scheduling applications' jobs.

### 4.2 Background

This section introduces the key terms Naïve Byes and SOM combined to implement the proposed framework. Section 4.1.1 differentiates the CC and FC, which is significant for understanding both paradigms' behavior and expectations.

#### 4.2.1 Fog Computing (FC)

The FC paradigm is frequently called "Cloud, close to the application" [98], [102]. To reduce the latency, FC infrastructure enables applications to run as close to the source of a request as possible. Figure 4.1 demonstrates the relationship between the FE, FN, and edge devices.



**Figure 4.1:** Positioning the Edge Devices and FNs in an FE

FC represents a global network of interconnected items that can each be addressed differently based on established communication protocols [103]. These items act as the IoT devices' physical representations, the origin of data collection, and they can be measured, comprehended, and analyzed [102]. FC has become popular among latency-sensitive IoT application jobs due to all these capabilities. These devices linked on FC are multiplying daily, and it is estimated that by 2025, there will be around 75.44 billion connected devices worldwide [104]. The chances of achieving low latency automatically increase when the right FNs are selected correctly [80], [105].

#### **4.2.2 Naive Bayes**

The Naive Bayes algorithm relies on the assumption of predictor independence and is based on the Bayes theorem. The Naive Bayes algorithm for classification assumes that a feature's presence in a class is independent of the existence of any other feature. Naive Bayes classification aims to forecast the class label  $C$  for a specific instance  $X$ , depending upon its features or attributes  $x_1, x_2, x_3, \dots, x_n$ . Using the Bayes theorem, Naive Bayes determines the likelihood that an instance belongs to each class and then places the class with the greatest likelihood to the input.

#### **4.2.3 Self-Organizing Maps (SOMs)**

In ML, SOMs are a kind of artificial neural network utilized for unsupervised learning. SOMs are used for dimensionality reduction, clustering, and data visualization. Some significant features of SOM are:

- Labeled data is not necessary for SOMs.
- SOMs preserve the input data's topological characteristics.
- They simplify complex data by reducing its dimensions to a lower-dimensional map.
- They work exceptionally well for exploratory data visualization and analysis.

SOM helps identify connections or trends in the dataset and for exploratory data analysis. They reveal patterns or groupings within the data, offering perceptions of the underlying frameworks.



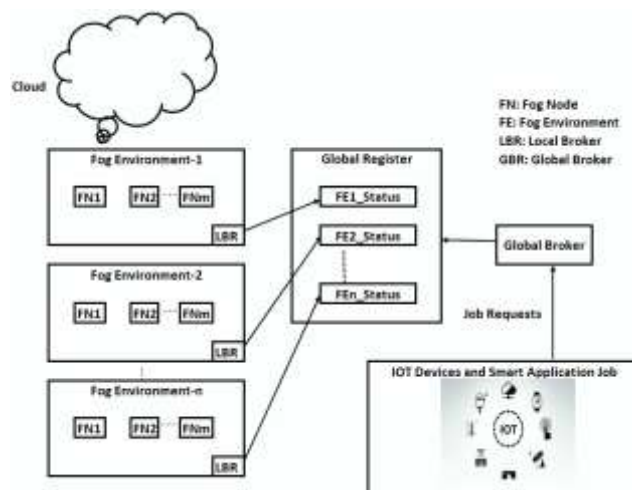
### 4.3 Proposed Framework

In this framework, FC is approached using one of the methods already used in another paradigm. The proposed framework is inspired by the scheduling mechanism proposed by Sandhu and Sood [16]. They presented a scheduling architecture considering quality of service (QoS) for large data applications. Their architecture utilizes K Nearest Neighbors (KNN), the Naive Bayes algorithm, and the Self-Organizing Map (SOM). Modifying well-established scheduling algorithms and switching to the other DS paradigms reduces the computational effort significantly.

The entire set of characteristics under consideration is altered when the focus is switched from the Cloud to the Fog. In Fog, latency is the most critical factor when choosing a job allocation node. Both methodologies use completely distinct implementation setups, datasets, and applications for experimentation. By extending the dual scheduler approach to FEs with specific parameters and real-time experimental setups, the proposed framework aims to broaden its scope. The proposed framework aims to reduce the job's execution time and increase efficiency by moving it from Cloud to Fog.

#### 4.3.1 Architecture

Figure 4.2 depicts components of the proposed framework for scheduling IoT application jobs in FEs.



**Figure 4.2:** High-Level Architecture: Dual Broker-Based Framework

The proposed framework consists of three primary components discussed below:

- **LBR:** Every FE has an independent LBR responsible for record-keeping the resources on the FNs in an FE.
- **GBR:** Global Broker is a centralized component that receives the load status from the Global Register and makes the necessary decisions for job scheduling based on these status details.
- **Global Register:** A global register is a component that stores the status data of each available FE in the network.

Conceptually, there are  $n$  number of FEs ( $FE_1, FE_2, FE_3, \dots, FE_n$ ) and  $m$  number of FNs inside an FE. The client part of the framework includes the IoT devices and the Smart Application Job. LBRs have direct access to the FNs' resource status, as depicted in Figure 4.2, and the GBR is directly linked to the applications that need specific resources for their jobs.

**Table 4.1:** Record kept by an LBR

Fog Nodes				
LBR				
	CPU	Memory	GPU	Storage
FN1	FN1_CPU	FN1_Memory	FN1_GPU	FN1_Storage
FN2	FN2_CPU	FN2_Memory	FN2_GPU	FN2_Storage
FN3	FN3_CPU	FN3_Memory	FN3_GPU	FN3_Storage
....	...	...	...	...
FNm	FNm_CPU	FNm_Memory	FNm_GPU	FNm_Storage

The entire job scheduling process in the proposed framework can be summarized in the following eight steps. Inputs from IoT application's job requests or SLAs are used as the process's QoS and operational needs. Two scheduling sub-algorithms are used in the process, and the process ends by returning an ID of the FN or group of nodes to which the present job may be assigned. If none of the available nodes can execute the job, it is dispatched to the Cloud and returns a null value.

---

**Set of the steps to be performed for scheduling IoT job request**

---

**Step 1:** Start

**Step 2:** The user sends the request to the IoT environment

**Step 3:** LBR receives the request

**Step 4:** LBR chooses the FE for the job assignment (Algorithm 4.1)

**Step 5:** Request is received LBR of the chosen FE.

**Step 6:** LBR fixes the final node, based on QoS parameters. (Algorithm 4.2)

**Step 7:** The submitted request is assigned to the chosen FN.

**Step 8:** END

---

### 4.3.2 Naïve Bayes Algorithm for the Proposed Framework

The Naive Bayes classification method is used here to group the applications. For this, the Bayes algorithm is employed.

$$P(\text{category}_i/\text{tuple})=P(\text{tuple}/\text{category}_i)P(\text{category}_i)/p(\text{tuple})$$

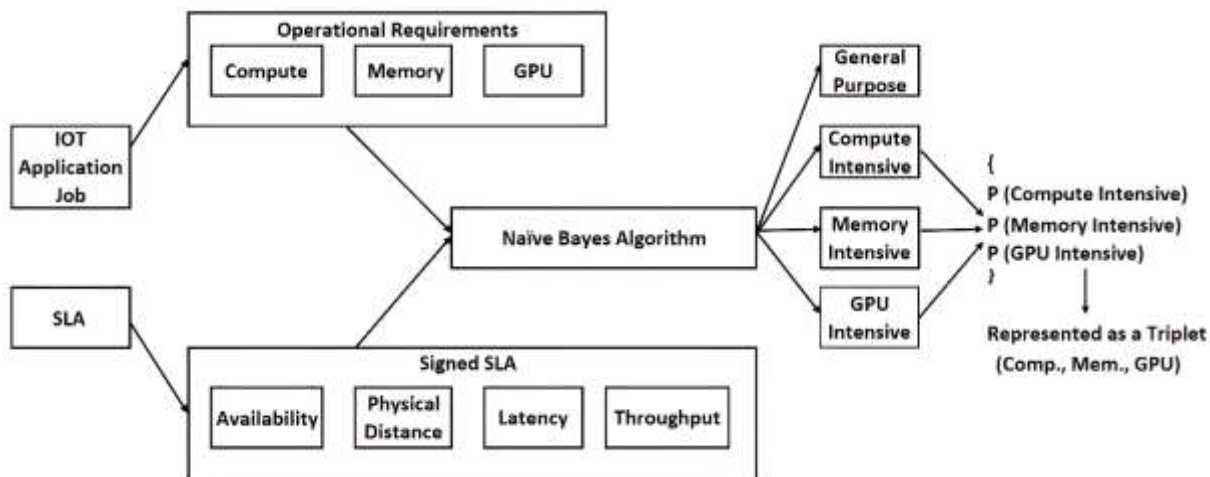
**Eq. 4.1**

Assuming that every job request has its operational and QoS requirements, the Naïve Bayes Algorithm is provided with both sets of requirements, and as an output, it generates a tuple (Comp., Mem, GPU) for each job request. This tuple determines the probability of a job falling into a specific category: Generic, Compute Intensive, GPU Intensive, and Memory Intensive. Table 4.2 displays the node types and their resource.

**Table 4.2:** Resource availability based node types

<b>Node Type</b>	<b>GPU</b>	<b>Memory</b>	<b>Compute</b>
<b>General Purpose</b>	N	N	N
<b>GPU</b>	High	N	N
<b>Memory</b>	N	High	N
<b>Compute</b>	N	N	High

The default choice for a request is the general-purpose (generic) category. One application job may fall under multiple categories. Figure 4.3 demonstrates the process of tuple generation.



**Figure 4.3:** Triplet Generation based upon QoS Parameters and Functional Requirement

This exercise ensures that every request receives enough resources to handle it efficiently. The operation of LBRs located inside each FE is represented by Algorithm 4.1.

#### **Algorithm 4.1: LBR Algorithm**

**Inputs:** All the operational parameters

**Outputs:** An identifier.

**Step 1:** Calculate the prior probability for given class labels.

**Step 2:** Find the probability of likelihood with each attribute for each class

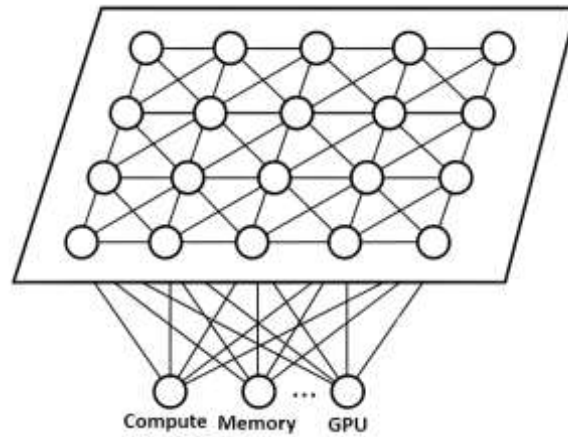
**Step 3:** Utilize the resultant values in the Bayes Formula and calculate posterior probability.

**Step 4:** Select the class with the highest probability, given that the input belongs to the highest probability class.

#### **4.3.3 Modified Self Organizing Maps (MSOM)**

SOM creates topological ordering through a competitive learning mechanism. The degree of their relationship is shown by how closely the FNs are spaced in topological ordering.

The proposed framework using a Self-Organizing Map (SOM) builds four types of virtual nodes. These nodes are computing-intensive, memory-intensive, GPU-intensive, and generic. Figure 4.4 shows the mapping of the generated triplet onto the resource cluster.



**Figure 4.4:** Use of SOM to map the triplet on the resource cluster

Each virtual node's specifications and Quality of Service (QoS) values are stored in an LBR. Table 4.3 displays the information that was recorded for each QoS parameter.

**Table 4.3:** Details of QoS parameters

QoS Parameter	Measured	Value Type	Provided By
Availability			
Memory	GB	Absolute	User Defined
CPU	GHz	Absolute	User Defined
GPU	GB	Absolute	User Defined
Physical Distance	Hops	Absolute	Network Architecture
Latency	Round Trip Time (RTT) or the Time to First Byte (TTFB)	Absolute	Dynamically System Defined
Throughput	Gbps	Absolute	Dynamically System Defined

Job requests are routed to specific virtual FN based on the probability tuple generated by the Naive Bayes algorithm. If a particular FN has surpassed its capacity, it can use the topological ordering of Algorithm 4.2 to access resources from nearby free virtual nodes. This technique is highly effective; it minimizes the wait time for IoT application job requests and ensures compliance with QoS standards.

---

**Algorithm 4.2: Creating Virtual Cluster using MSOM Technique**

---

**Input:**  $category\_list []$ ,  $server\_list [] []$

**Output:** virtual FN

**Step 1:** Set  $p=0$

**Step 2:** For all elements in  $category\_list []$

**Step 2.1:** Set  $q=0$

**Step 2.2:** for all machines in  $server\_list [] []$

**Step 2.2.1:** Determine the closed reference vector for  $server\_list [q] []$   
        based on  $category\_list [p]$ .

**Step 2.2.2:** Update the close reference vectors using

$$m_j(t+1) = m_j(t) + y_j(t) (z(t) - m_j(t))$$

**Step 2.2.3:** set  $q=q+1$

**Step 2.3:** set  $p=p+1$

**Step 3:** Assign each server machine to its closest reference vector

**Step 4:** Return reference vectors and create a virtual node

---

In Algorithm-4.2,

- $category\_list []$  represents the list of IoT application's job categories generated based on functionality and QoS requirements.
- $server\_list [[]]$  contains the machines installed in FEs and their specifications.
- $z(t)$  represents the server machine currently considered at time  $t$ .
- $m_j(t)$  represents the value of the nearest reference vector to  $z(t)$  at time  $t$ .
- $y_j(t)$  represents the neighborhood effect between  $m_j$  and  $y$  for creating topological ordering at time  $t$ .

It is now time to assign jobs to these job nodes after creating the virtual nodes. Algorithm-4.3 depicts how a virtual node is assigned a job:

---

**Algorithm 4.3: Job Assignment to Virtual Node**

---

**Inputs:** Resource Utilization table, current node, new job request, p[].

**Step 1:** Set  $i=0$ .

**Step 2:** Select node from p [i].

**Step 3:** if the node is over-utilized

**Step 3.1:** Find a virtual node close in topological ordering with free resources.

**Step 3.2:** if (node\_free! = null)

**Step 3.2.1:** Shift desired resources from node\_free to the selected node.

**Step 3.2.2:** Update the resource utilization table

**Step 3.3:** Else

**Step 3.3.1:** Set  $i=i+1$

**Step 4:** END

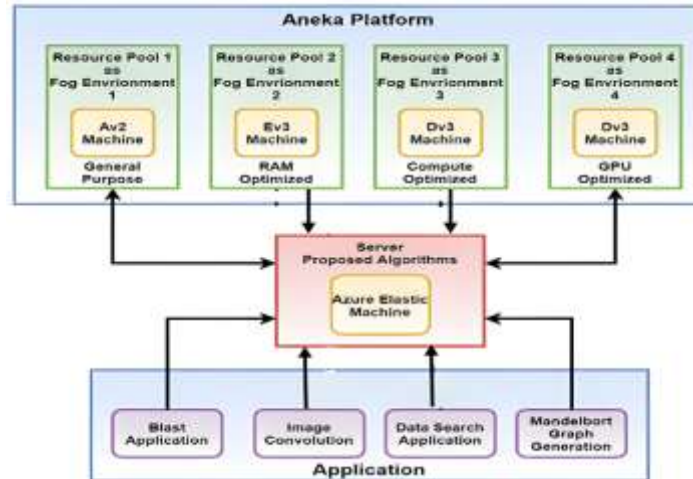
---

Algorithm 4.3 includes a resource utilization table that tracks the current and past resource utilization of an FN in the FE. The LBR keeps this table updated. Node\_free is an FN with enough resources to complete a new job and is closer to the present FN in topological ordering. The top three IoT application job categories are listed in p[] based on conditional probabilities generated by the Naive Bayes method.

#### 4.4 Experimental Results and Discussion

The framework has been tested on Microsoft Azure and Aneka resources for organizing platforms and applications. Aneka, a .NET-based service-oriented grid computing platform, is used for workload discovery, scheduling, and balancing [106]. Four ready-made Aneka-based applications are employed in testing. Microcomputing resources used as FC nodes and grouped to form FC environments have been created using Azure. The proposed testbed consists of four settings, each running on an Av2, Ev3, or Dv3 computer tailored for a particular job. The proposed framework has been put to the test using four applications:

- Blast: a parameter sweep application;
- Convolution: an image processing application;
- Mandelbort: a standard thread application;
- Data Search: an algorithmic search application.



**Figure 4.5:** Testbed for the Performance Evaluation of Proposed Framework

The experiment lasted for 100 minutes and involved generating time-varying loads. These loads were sent to server computers operating within the same Azure network as other machines. The server used the proposed framework to choose the most suitable computer to send requests.

A saturation was seen after this period; thus, the experiment was continued for an additional 100 minutes. In the experiment, we tracked five metrics for three installed pieces of infrastructure:

- resource utilization,
- availability,
- response time,
- waiting time,
- completion time.

Two computing infrastructures are employed, and they are briefly detailed below:

- a) **CC infrastructure:** Microsoft Azure's Australia Region has provisioned a computer with an elastic configuration. An application request was sent from India to consider network latency. When the demand for the application increases, an auto scaler on the machine purchased from Azure automatically changes the configuration.
- b) **FC infrastructure:** As seen in Figure 4.5, a set of machines connected to the same network where the apps were causing the load were set up to reduce latency.

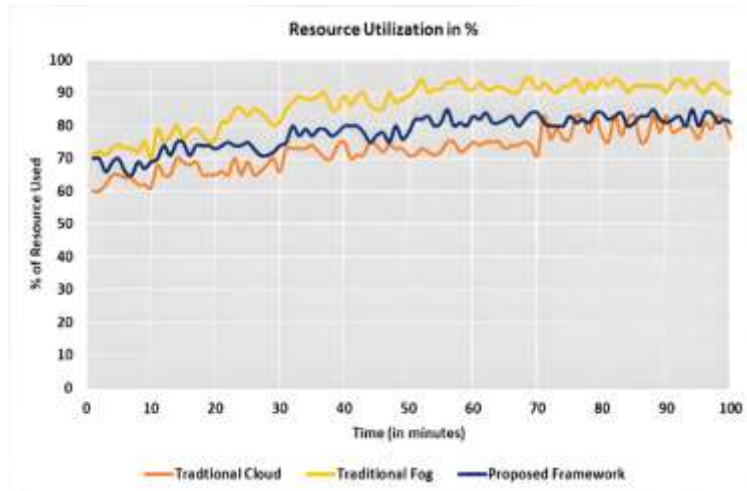


The performance evaluation of three computing models is compared as described below.

- a) **Conventional CC:** this computing paradigm directs all jobs straight to CC resources. In this model, FC is not used. [16]
- b) **Traditional FC:** This computing model treats all jobs equally without segregation and assigns a job to any available FC resource [107].
- c) **Proposed FC:** This framework uses an ML-based computing model to allocate jobs to best-suited FNs. Requests for Image Convolution, an image processing application, were routed to computers with GPU power via FNs created for them.

The proposed framework is evaluated on five parameters: ARU, RAV, ART, AWT, and ACT. Many other parameters can be evaluated, but the selected set of evaluation parameters is considered specifically significant because DS is in demand for providing high-end, low-cost resources to its users. A system is successful if it utilizes the resources without waste and keeps them available when requested. Except these, other parameters such as ART, AWT, and ACT are to be taken care of so that requests can be completed within agreed-upon timelines.

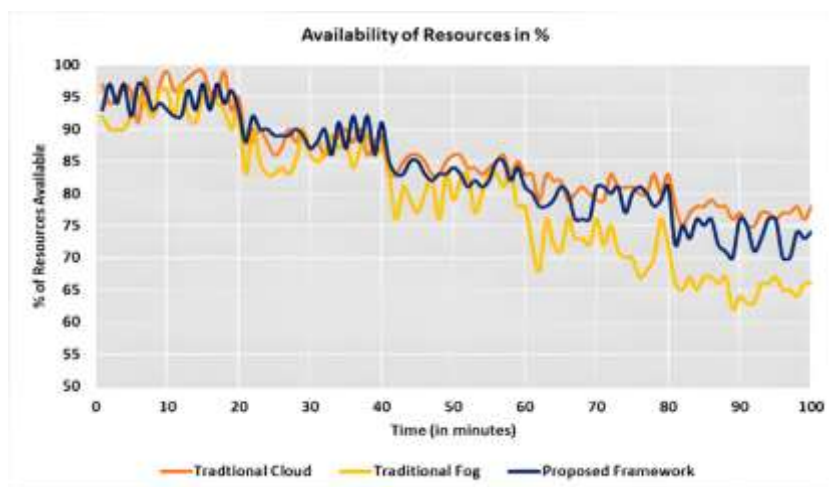
The resource consumption for all three schemes is shown in Figure 4.6, where the ARU for standard FC infrastructure was close to 92 percent. However, with maximum usage of 80%, CC and the proposed framework were operating at the same level. High usage has a negative impact on the application and infrastructure's overall performance. The average resource consumption for plotting the graph was calculated every five minutes. Compared to CC and the proposed framework, the conventional Fog technique was the least accessible to the submitted jobs because it was heavily used. Even the proposed framework performed superior to the CC setup.



**Figure 4.6:** ARU of Available Computing Infrastructure

Figure 4.7 depicts the RAV for each of the three computing models used in the experiment. In the beginning, all models had an RAV of almost 90%. However, the proposed FC framework offered better resource scheduling than the conventional FE, which resulted in greater availability. The proposed framework improved availability and resource use.

As visible in the results, the response time is increasing. The system has enough resources to execute the requests and behaves as expected. The response time is increasing because new requests are being added without waiting for the existing requests to end [106]. The framework is not intended to evaluate the system's performance; instead, it is an attempt to verify if a distributed broker can be applied to this approach and if it produces better results than its counterparts.



**Figure 4.7:** RAV Comparison at the installed infrastructure

The typical ART of various infrastructures is shown in Figure 4.8. Response time is measured when an application sends a job to a Cloud-based machine and receives a response in return.

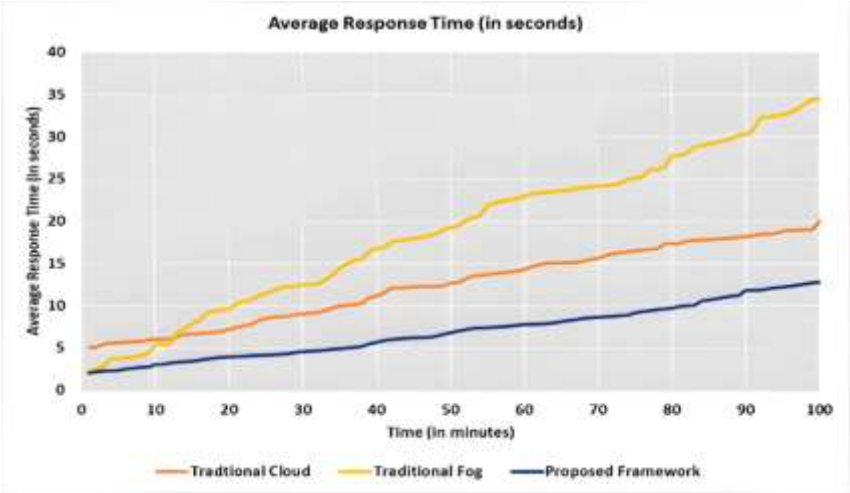


Figure 4.8: ART comparison of the Installed Infrastructure

The proposed framework had the quickest reaction time. The AWT for a job posted by the application is shown in Figure 4.9. Traditional FC initially saw shorter waiting times than CC due to lower network latency. However, fewer resources were available after five minutes, which caused the waiting time to lengthen. However, the proposed framework outperformed all others due to its efficient resource utilization. Job sent to the infrastructure would take longer to complete depending on the response and waiting times. The same pattern was evident, with the proposed framework outperforming the other two.

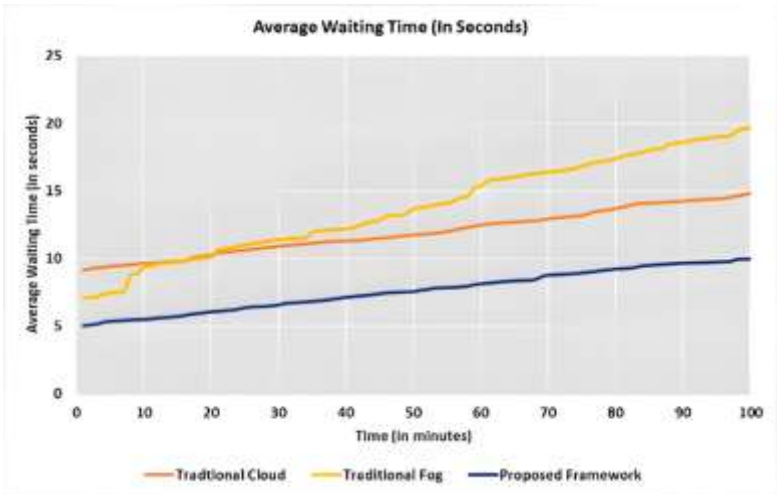


Figure 4.9: AWT comparison of Installed Infrastructure

The average job completion time for each of the three computing models is shown in Figure 4.10. The proposed framework has the best average job completion time because of its effective ML job scheduling and usage of an FC infrastructure to reduce latency.

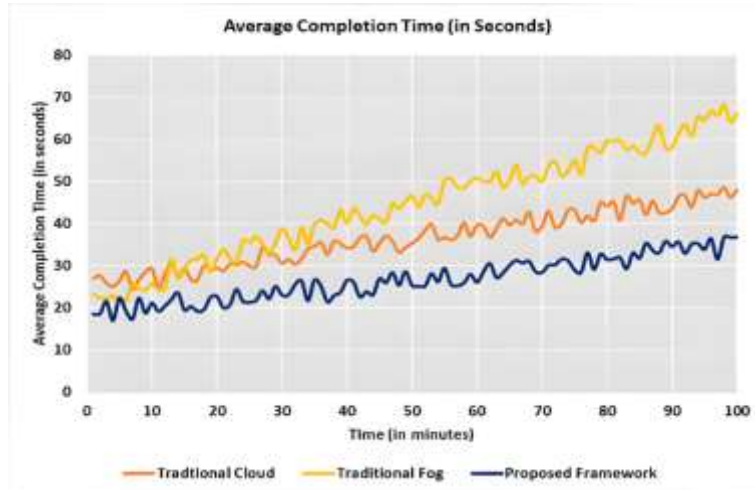


Figure 4.10: ACT comparison of the Installed Infrastructure

Table 4.4 summarizes the discussion on the experimental results, comparing all three approaches discussed in this section.

Table 4.4: Summary of obtained results.

		ARU	RAV	ART	AWT	ACT
<b>Traditional Cloud</b> [16]		75%	78%	20 sec	15 sec	48 sec
<b>Traditional Fog</b> [80]		90%	65%	35 sec	20 sec	66 sec
<b>Proposed Framework</b>		80%	75%	13 sec	10 sec	37 sec
<b>Improvement</b>	<b>Cloud</b>	<b>5%</b>	<b>3%</b>	<b>7 sec</b>	<b>5 sec</b>	<b>11 sec</b>
	<b>Fog</b>	<b>-10%</b>	<b>-10%</b>	<b>22 sec</b>	<b>10 sec</b>	<b>29 sec</b>

Table 4.5: Proposed Framework vs Other Solutions

Ref. No.	Title	Goal	Limitations
[58]	Latency-Aware Application Module Management for Fog	FEs must follow a latency-sensitive policy when managing	1. Missing brokering, 2. Missing hierarchical distribution of load balancing jobs, 3. QoS parameters not considered, 4.No ML

	Computing Environments	the application modules.	technique is used, 5.No clustering of jobs and resources
[62]	DRL-Scheduling: An Intelligent QoS-Aware Job Scheduling Framework for Applications in Clouds	A framework for managing jobs using QoS. The framework's main component was an algorithm for scheduling jobs based on deep reinforcement learning.	1. Missing brokering, 2. Missing hierarchical distribution of load balancing tasks, 3. No clustering of jobs and resources
[63]	A machine-learning-based time-constrained resource allocation scheme for vehicular Fog computing	A metric to assess the effectiveness of apps for improving road safety that combines information-based networking technologies with a Fog virtualization strategy to cut down on PRT authors.	1. Missing brokering, 2. Missing hierarchical distribution of load balancing tasks, 3. QoS parameters not considered, 4.No ML technique is used, 5.No clustering of jobs and resources
[64]	A Supervised Learning Based QoS Assurance Architecture for 5G Networks	A QoS assurance structure that takes into account widespread connectivity, extremely low latency, and high-speed data transfer rates as QoS parameters	1. Missing brokering, 2. Missing hierarchical distribution of load balancing tasks, 3. No ML technique is used, 5.No clustering of jobs and resources
[67]	Computation Scheduling for Distributed Machine Learning With Stragglers	Showed the cumulative completion time as a function of computational load through the master-controlled allocation of tasks among numerous workers.	1. Missing brokering, 2. Missing hierarchical distribution of load balancing tasks, 3. QoS parameters not considered, 4.No ML technique is used, 5.No clustering of jobs and resources

## 4.5 Summary

A proposed framework aims to allocate IoT application jobs to FNs in diverse FEs, using ML classification and integrating the Quality of Service (QoS) parameters. The framework's performance is evaluated using QoS parameters: throughput, availability, latency, and physical

proximity. It offers 6% better resource utilization, 4% greater RAV than traditional Cloud [16], 1.8 times faster response times, and 2.7 times better resource consumption than traditional Cloud [16]. Last but not least, the ACT was 1.6 times faster than traditional Fog [80] and 1.3 times faster than traditional Cloud [16]. By scheduling the jobs to FNs, where it is possible to produce better outcomes in terms of timely execution and high-quality output, the proposed framework improved resource utilization, reduced completion time, and decreased latency.

The next chapter discusses an approach to develop a multilevel hierarchical broker for job scheduling in hexagonal FEs for better global geographical coverage of distributed and ad-hoc Fog services.

# **CHAPTER 5**

## **MULTILEVEL BROKER-BASED JOB SCHEDULING**

### **5.1 Introduction**

The previous Chapter elaborates on a dual-broker implementation in FC. That solution refers to a single FE. This chapter presents a hierarchical broker-based technique for job scheduling in hexagonal FEs for better global geographical coverage of distributed and ad-hoc Fog services.

### **5.2 Background**

Traditional CC infrastructures cannot meet the rigid latency requirements of IoT applications [105]. To cater to the demands of latency-sensitive applications, various computing paradigms, such as mobile EC [108], FC [109], and sensor Clouds [110], are being employed. Among these, Fog is a highly effective and popular computing platform. Various operational and QoS aspects influence job scheduling decisions, including energy consumption, waiting times, and operating expenses. The framework discussed in this chapter aims to enhance five QoS parameters: RAV, ARU, ART, AWT, and ACT.

#### **5.2.1 Coverage of Geographic Regions (GR)**

IoT applications involve managing an extensive network of sensor nodes randomly placed in a RoI to serve a job's immediate processing needs on behalf of the Cloud [111]. These nodes are responsible for sensing the environment and carrying out specific tasks. However, it is possible that some of these nodes may be left unattended or may make their own decisions while performing their tasks [19]. The RoI in this chapter is divided into hexagonal regions due to the equal distance between their centers, ensuring complete coverage of FNs [112]. However, the radius of a hexagon may change from case to case.

#### **5.2.2 Hexagonal FE Organization**

The hexagonal FE grid structure is chosen for its efficiency in geographical classification, optimal placement of IoT devices, energy-saving capabilities, and spatial consistency [112]—any IoT application job generated within a particular FE (FE1). In case FE does not have sufficient resources, another suitable FN is searched, prioritizing the nearby nodes (FE2 to FE7)

at an assumed distance  $d1$ . The next set of FEs (FE8 to FE19) is situated at a distance of  $d2$ . Preferably, an alternative FE is searched at distance  $d1$ , as long as the job's latency deadline does not exceed it. The job is sent to a Cloud data center if all available FEs exceed the latency deadline. Figure 5.1 demonstrates the hexagonal arrangement of the FEs.



**Figure 5.1:** Geographic area partitioned in hexagonal GRs for Fog resources

### 5.3 Proposed Framework

This section elaborates on a hierarchical broker-based technique for job scheduling in hexagonal FEs for better global geographical coverage of distributed and ad-hoc Fog services. The proposed framework verifies if the congestion at the job scheduling entities can be reduced by adding multiple hierarchical scheduling brokers and providing better geographical resource coverage.

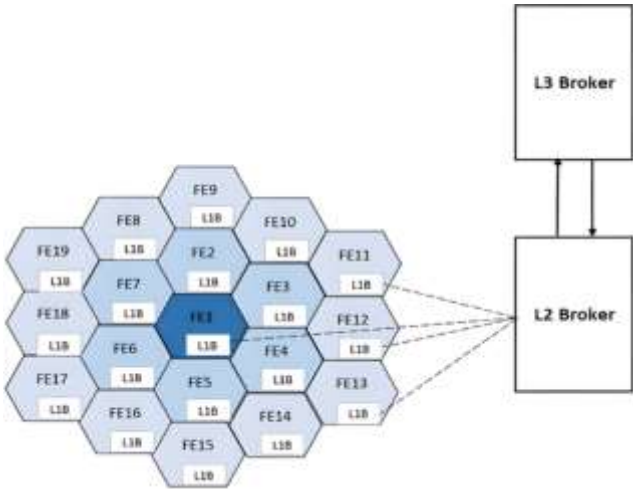
#### 5.3.1 Architecture

Each hexagonal region in the proposed framework corresponds to an FE, accommodating a specific number of FNs. Three brokers, known as Level-1 Broker (L1B), Level-2 Broker (L2B), and Level-3 Broker (L3B), are used to schedule the jobs:

- **L1B:** Each FE has an L1B deployed inside it. In the proposed framework, it is the responsibility of the L1B to keep track of the resources available within a specific FE.
- **L2B:** An L2B broker serves as a mediator between the L1Bs and L3B.
- **L3B:** The L3B is a centralized broker that interacts with all L2Bs of every GR.

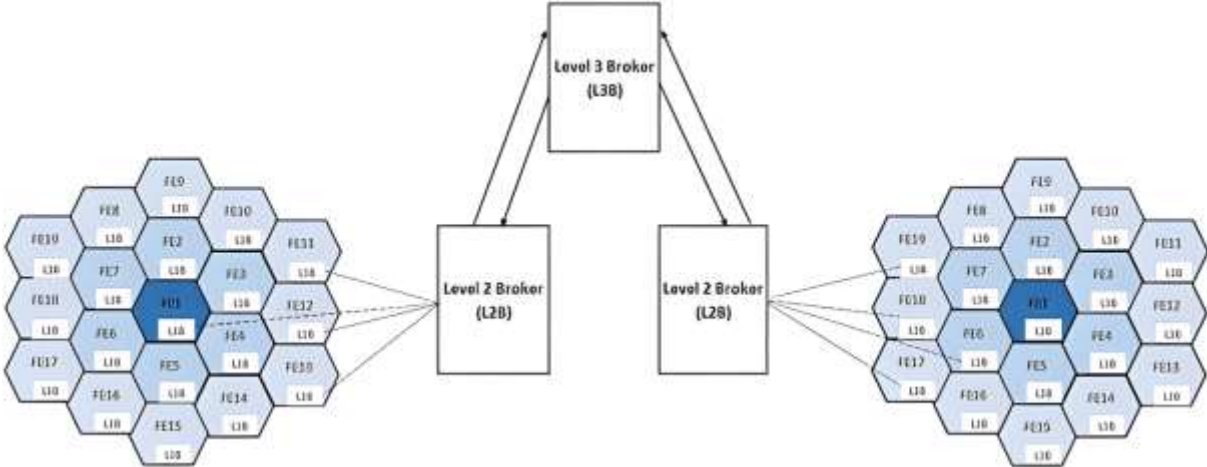


Figure 5.2 shows the placement and communication of the FE, L1B, L2B, and L3B in the proposed scheduling framework.



**Figure 5.2:** Interaction between the L2B and L1B

There is a single instance of L3B communicating with all the L2Bs. L3B is provoked only in extreme cases when no suitable FE and FN are located in the current GR. Figure 5.3 shows the interaction between the multiple GRs with the help of all three brokers.

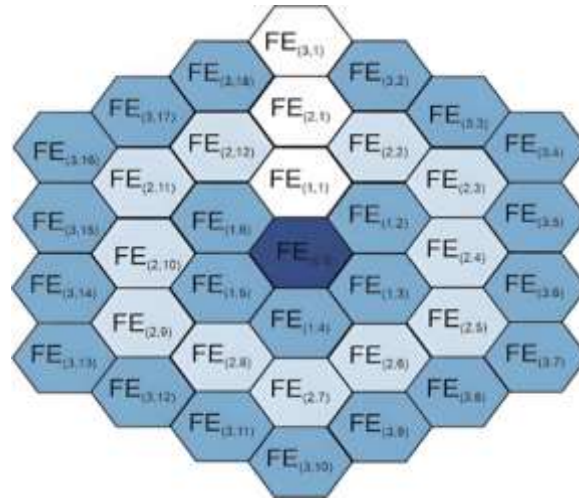


**Figure 5.3:** Interaction between L1B, L2B and L3B.

**5.3.2 The process of selection of an FE**

- **Estimation of the layer count**

The layer count changes with every request and affects the potential wait time for any response from the FEs.



**Figure 5.4:** Increase in the number of FNs with Layer Count

The job is preferably sent to the nearest FE to manage the latency and deadline parameters. Layer count is crucial for traversing all of the FEs in a layer because there are different numbers of FEs in every layer. Using the number provided by the L1Bs, the layer count is calculated as follows:

$$Layer\ count = ((FE\_Count \% 6) > 0) ? FE/6 + 1 : FE/6;$$

**Eq. 5.1**

Navigating between nodes and ascertaining their PP once the layer count is known is simpler.

- **To determine the availability of resources**

After collecting the job resource requirements, the L1B of the FE collects data from FNs regarding their available resources and compares it with the resources required. The L2B receives a positive response if the free resources are sufficient for the job. L2B compiles a list of all the available hexagonal regions by adding the FE IDs of all the FEs that respond positively to an *available []*.

- **Calculation of QoS Score**

The QoS score in the proposed framework depends upon three parameters which are:

- Physical Proximity (PP): Determined by L2B
- Bandwidth (BW): Recorded by L1B
- Quality of the Devices (QD): Recorded by the L1B

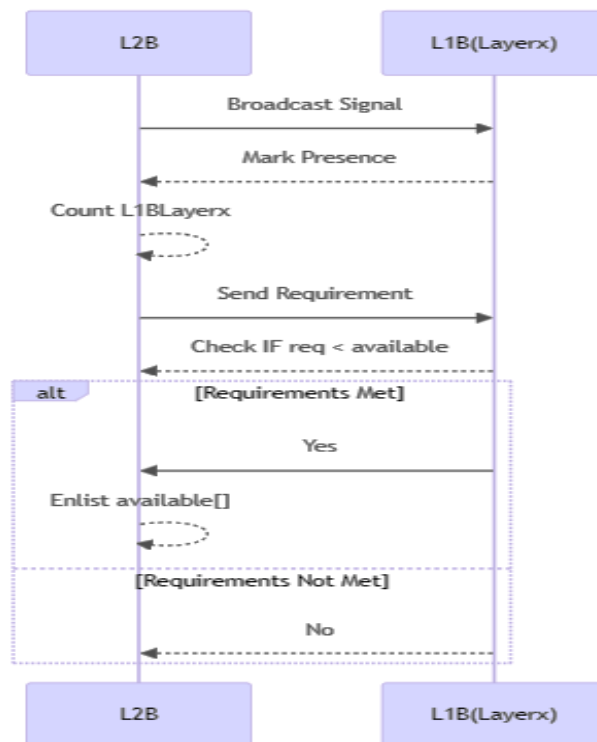
These parameters were chosen because of their noticeable contribution to job latency. QoS score is calculated based on the weightage assigned to all the above parameters, i.e.,  $PP_w$ ,  $BW_w$ , and  $QD_w$ . The weightage values of these parameters are shown in Table 5.3.

$$QoS\ Score = PP * PP_w + BW * BW_w + QD * QD_w$$

**Eq. 5.2**

**Table 5.1:** Weightage of QoS parameters to calculate the QoS score

	High	Medium	Low
<b>PP</b>	1	0.5	0
<b>BW</b>	1	0.5	0
<b>QD</b>	1	0.5	0



**Figure 5.5:** Interaction between L2B and L1Bs of a layer

- **Ranking the FEs**

The current layer's FEs are evaluated according to their Quality of Service (QoS) score. The best solutions are prioritized first in this ranking to execute the job swiftly and effectively. As a result, processing starts with the FE having the highest QoS rating.

- **Calculation of New\_Deadline**

The proposed framework chooses an FN by comparing the actual time the FN needs to finish the application job with the deadlines specified in the SLA. To obtain the New\_Deadline, the total time invested in  $t_{layer}$ ,  $t_{avail}$ ,  $t_{score}$ , and  $t_{transfer}$  is subtracted from the SLA deadline.

$$New\_Deadline = deadline - (t_{layer} + t_{avail} + t_{score} + t_{transfer})$$

**Eq. 5.3**

In Eq. 5.1

- $t_{score}$ : time taken to calculate the QoS score
- $t_{avail}$ : time taken to find the available FEs
- $t_{layer}$ : time taken to count the layers
- $t_{transfer}$ : and transferring data

If none of the FEs in the current layer satisfy the deadline criteria, then the system approaches the FEs in the next layer. The job is routed to the Cloud if the system cannot find a suitable FE in any layers.

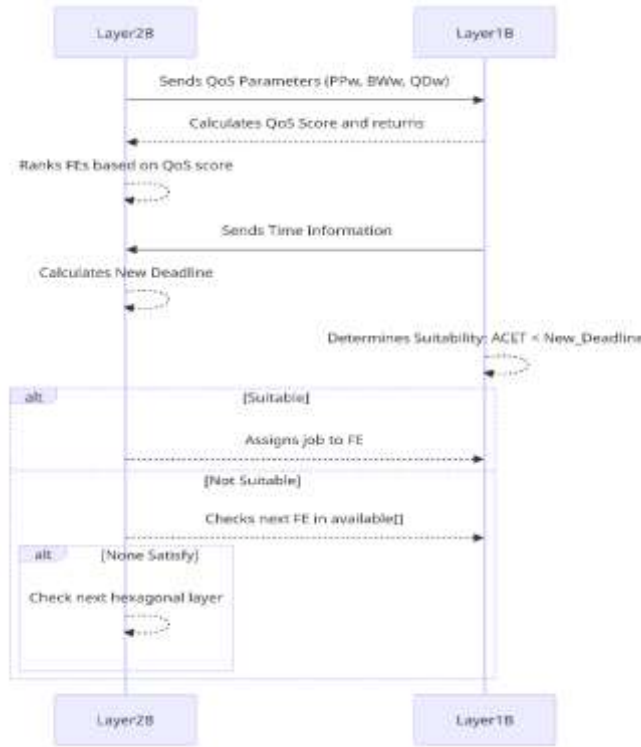
- **Assigning the job to an FE**

For assigning a job to an FE, its New\_Deadline is compared to the actual execution time (ACET) to determine if the current FE meets the latency and efficiency requirements.

$$FE_{selected} = FE_{current}, \text{ IF } ACET_{current} < New\_Deadline$$

**Eq. 5.4**

If not, the next available FE is assessed using the same criteria. If none of the existing FEs meet the latency requirement, the search process restarts in the next hexagonal layer that contains FEs. Figure 5.5 illustrates how the L2B and each L1B work together to select the best FE in a GR.



**Figure 5.6:** Assigning the job to a suitable FE.

- **Finalizing an FN**

The same set of steps that are used for selecting an FE is performed to select an FN within the selected FE, i.e.,

- enlisting the nodes with available resources,
- figuring out their QoS scores,
- ranking each node based on those scores,
- figuring out an FN's New\_Deadline,
- assigning the job to an FN

The L1B performs all these steps. The status of free nodes was calculated during the process of FE selection, which is reused here. Algorithm 5.1 outlines the step-by-step procedure.

---

**Algorithm 5.1: Load scheduling in an FE.**

---

**Inputs:** Execution deadline, ACET, time to calculate layer count, PP, BW, and device quality, layer\_count=0

**Output:** FN

**Step 1:** Start

**Step 2:** The IoT app submits the request

**Step 3:** FE denies due to lack of resources

**Step 4:** Increment the layer count  $k$

**Step 5:** Set time taken to count the layers  $t_{\text{layer}} = t_{\text{layer}} + \text{new\_}t_{\text{layer}}$

**Step 6:** Collect RAV at all FEs by broadcasting a request to all the FEs at layer  $k$

**Step 6.1:** for each  $FE_i$  in layer  $k$

send a request containing *the requirements* to  $FE_i$  to provide *available*

if( $\text{required} < \text{free}$ )

set  $\text{cnt} = \text{cnt} + 1$

add  $FE_i$  to **available** [ $\text{cnt}$ ]

**Step 6.2:** collect **available** []

**Step 6.3:** Set  $t_{\text{avail}} = t_{\text{avail}} + \text{new\_}t_{\text{avail}}$  time taken to collect the availability

**Step 7:** Set  $j=0$

**Step 8:** Calculate the QoS score of FE at *available* [ $j$ ]

**Step 8.1:** Collect  $pp$ ,  $bw$ , and  $qual$  of FE

**Step 8.2:** Set  $\text{QoS\_score} = pp * pp_w + bw * bw_w + qual * qual_w$

**Step 8.3:** Set  $t_{\text{score}}$  = time spent in the calculation of QoS score

**Step 9:** Collect the  $\text{QoS\_score}$  from FEs at layer  $k$  and rank them

**Step 10:** Calculate the  $\text{New\_Deadline}$ .

$\text{newdeadline} = \text{deadline} - (t_{\text{layer}} + t_{\text{avail}} + t_{\text{score}})$

**Step 11:** Compare ACET with a  $\text{New\_Deadline}$

if ( $\text{ACET} > \text{newdeadline}$ )

Set  $j = j + 1$  and go to step 8

else

Search appropriate FN in the current FE

**Step 12:** Use the free resource status of all FNs in  $FE_j$

**Step 12.1:** for each FN  $n$  at  $FE_j$

Compare *required* resources with the free resources at  $FN_n$

if ( $\text{required} < \text{free}$ )

set  $\text{cnt} = \text{cnt} + 1$

Add  $FN_n$  to **available** [ $\text{cnt}$ ]

**Step 12.2:** collect **available** []

**Step 12.3:** Set  $t_{\text{avail}} = t_{\text{avail}} + \text{new\_}t_{\text{avail}}$  time taken to collect the availability

**Step 13:** Calculate the QoS score of  $FN_n$

**Step 13.1:** Collect  $pp$ ,  $bw$ , and  $qual$  of  $FN_n$

**Step 13.2:** Set  $QoS\_score = pp * pp_w + bw * bw_w + qual * qual_w$

**Step 13.3:** Set  $t_{score}$  = time spent in the calculation of QoS score

**Step 14:** Collect the  $QoS\_score$  from  $FNs$  at  $FE_j$  and rank them

**Step 15:** Calculate the  $New\_Deadline$ .

$$newdeadline = deadline - (t_{layer} + t_{avail} + t_{score})$$

**Step 16:** Compare  $ACET$  with  $New\_Deadline$

if ( $ACET > newdeadline$ )

Set  $n = n + 1$  and go to step 13

else

Allocate job requests to  $FN_n$

**Step 17:** If no more  $FE$  is **available** []

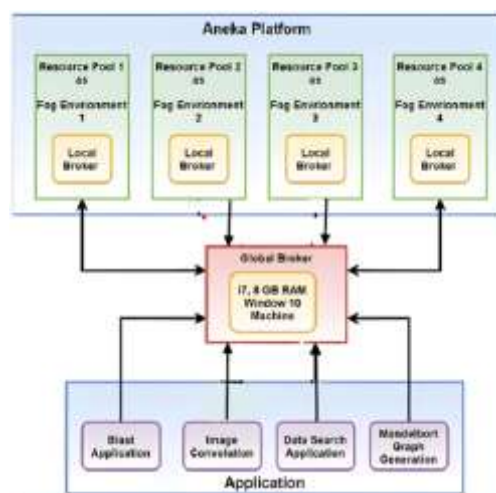
Go to step 4

**Step 18:** If there are no more layers, then send the job to the Cloud

**Step 19:** END

## 5.4 Experimental Results and Discussion

The proposed framework is tested by an emulator, which schedules Fog applications using hexagonal FE and a broker-based strategy. Figure 5.7 displays the emulator's testbed. The proposed testbed and its performance assessment are covered in-depth in this section.



**Figure 5.7:** Testbed for Experimental Setup and Performance Analysis

### 5.4.1 Testbed

The testbed uses four Aneka platform-built applications. As a result, these programs produce several independent jobs that can be assigned to any of the FEs shown in Table 5.5. Number of nodes in an FE and their configuration is as per the Table 5.6. Table 5.7 shows various parameters showing the time spent on the specified operations.

**Table 5.2:** Applications Used in Testbed

S. No.	Name	Tasks per minute	TtC Range
1	Blast Application	1000	5-15 seconds
2	Image Convolution	3600	1-2 seconds
3	Mandlebort	2500	1-5 seconds
4	Data Search	1000	5-15 seconds

**Table 5.3:** Number and configuration of the Nodes

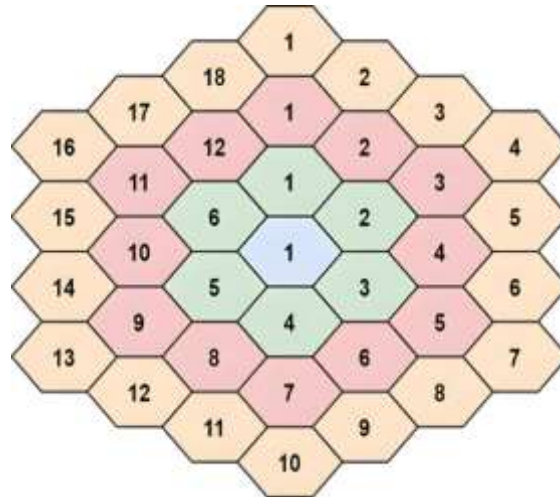
S. No.	FE	Number of Nodes	Node Configuration
1	FE 1	1	1 Core @ 2.9 GHz, 1 L2B RAM, 20 L2B HDD
2	FE 2	6	1 Core @ 2.9 GHz, 1 L2B RAM, 20 L2B HDD
3	FE 3	12	1 Core @ 2.9 GHz, 1 L2B RAM, 20 L2B HDD
4	FE 4	18	1 Core @ 2.9 GHz, 1 L2B RAM, 20 L2B HDD

**Table 5.4:** Parameters for the time used in the proposed framework's testbed.

Parameter	Value
$t_{\text{layer}}$	1-5 seconds (chosen randomly)
$t_{\text{avail}}$	3-8 seconds (chosen randomly)
$t_{\text{score}}$	1-5 seconds (chosen randomly)
Latency per hexagon	5-15 seconds (Poisson distributed)

In the proposed testbed, the L2B is a Windows 10 machine with Intel(R) Core (TM) i7-10,700 CPU 2.90 GHz processor, 16.0 GB (15.7 GB usable) RAM, and a 64-bit operating system. It operates a master Aneka container that controls the job scheduling on the desired FEs. The Aneka PaaS program was used to implement the proposed framework, which controls various scheduling aspects in distributed computing systems.





**Figure 5.8:** Setup of FEs for the Testbed

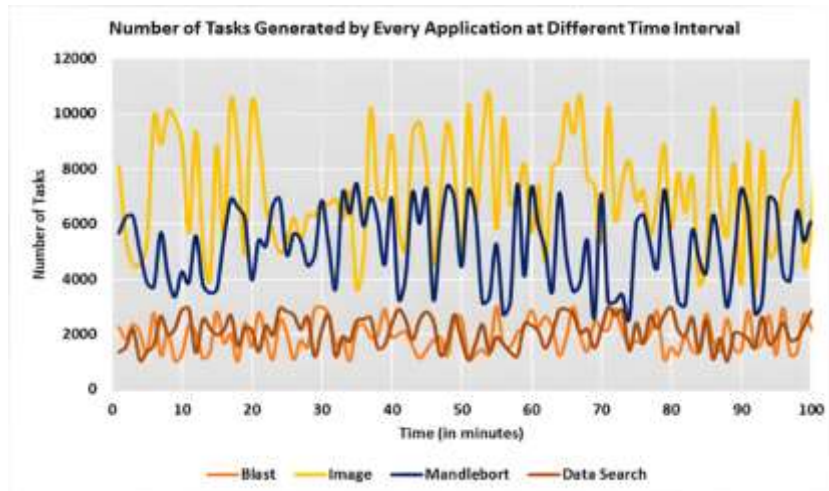
### 5.4.2 Performance Analysis and Comparison

The suitability of the proposed framework to schedule jobs in an FC environment using L1B and L2Bs is evaluated through comparison with its variant.

- **Variation 1:** In Variation-1, no jobs are moved to the nearby FE with available resources. A job is kept waiting for the resources in its FE.
- **Variation 2:** Layering and hexagonal structure are not performed, but the testbed described in Section 4.1 is used, and a job can be transferred to another FE. Jobs change at random.

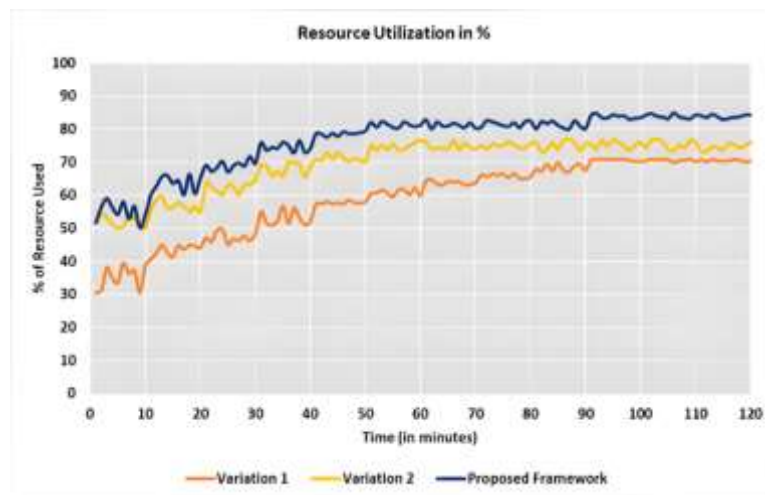
The proposed framework is evaluated for five parameters: RAV, ARU, ART, AWT, and ACT. These parameters are preferred in Fog computing because they measure a system's performance and check if it provides accessible and efficiently utilized resources. Metrics like response time, wait time, and completion time are also helpful in evaluating responsiveness and efficiency and providing valuable insights for improving the user experience.

Figure 5.9 shows the count of jobs produced by these four Aneka applications. Compared to the Blast program, the image convolution application produced many jobs. The magnitude and time required to finish each job varies, as shown in Table 5.5. To effectively populate the framework, a sufficient number of jobs have been generated and checked to verify scalability and dependability.



**Figure 5.9:** Number of Jobs Generated by Every Application at Different Time Intervals

The proposed framework’s ARU is shown in Figure 5.10. After 50 minutes of testing, all of the variations employed in the experiment reached saturation regarding resource utilization. The proposed framework had the highest resource consumption because it dispatches jobs to front-end servers other than the one where they were initiated. Although the resources are distributed equally across all FEs in the proposed framework, there are certain deviations where some FEs are overused while others are underused.



**Figure 5.10:** ARU Comparison

Figure 5.11 depicts the FE’s typical RAV, which is lower for the proposed framework than for other alternatives. The proposed framework has a slightly lower average RAV than Variation-1, which has the highest average RAV. In this experimental setup, RAV reveals the underutilization of resources. While some jobs can be moved to accessible resources, Variation-

1 does not permit this, leaving those resources underutilized. On the contrary, Variation-2 with the proposed framework makes it possible to identify more suitable resources and assign jobs to them, ensuring a fair workload distribution. In the first 60 minutes of the experiment, the proposed framework experiences a 20% lower average RAV. The difference is roughly 15% - 10 % after 60 minutes. The proposed framework can keep resources engaged longer than other alternatives.

Figure 5.12 shows the ART of application jobs. The proposed architecture has a much better response time than other variations. In the beginning, there's little difference in response times for the three approaches, but Variation-1's response time suddenly increases and keeps rising throughout the experiment. Variation-2 has a lower response time than Variation-1 but is still over 45 seconds higher than the proposed framework. By the end of the experiment, Variation-1's response time is 3.5 times higher than that of the proposed framework.

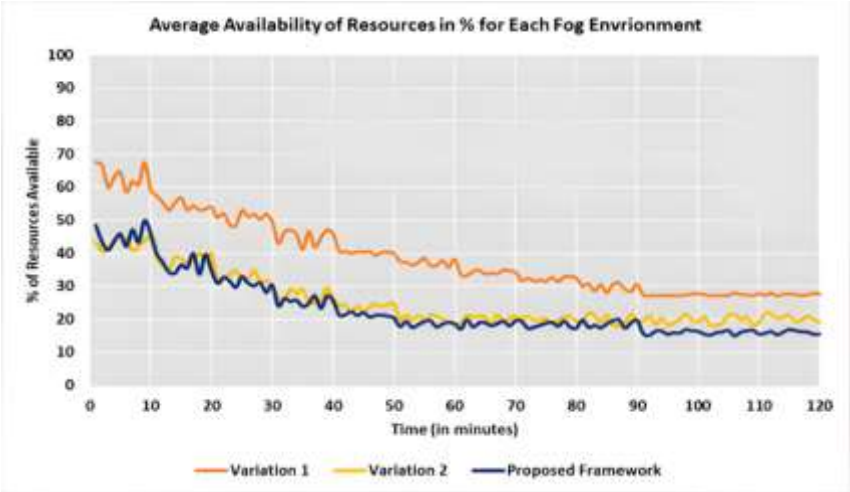


Figure 5.11: RAV Comparison

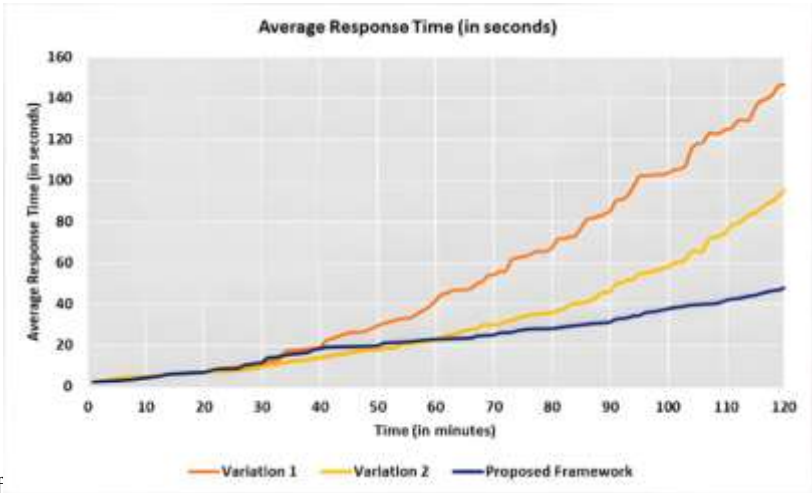
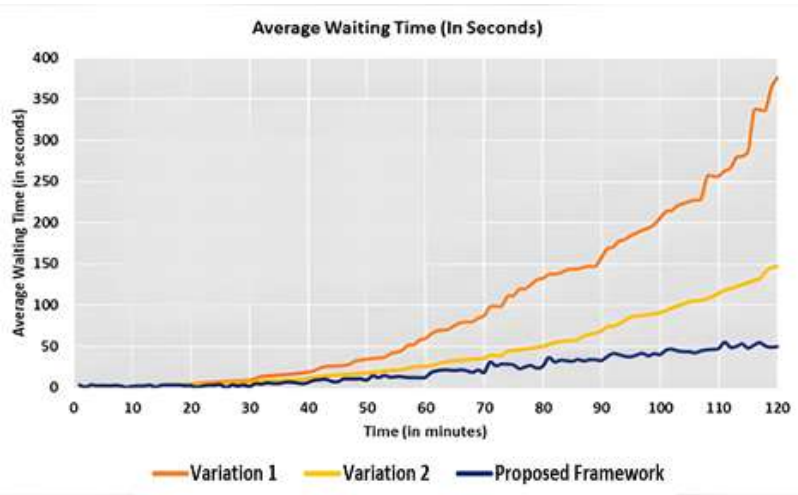


Figure 5.12: ART Comparison

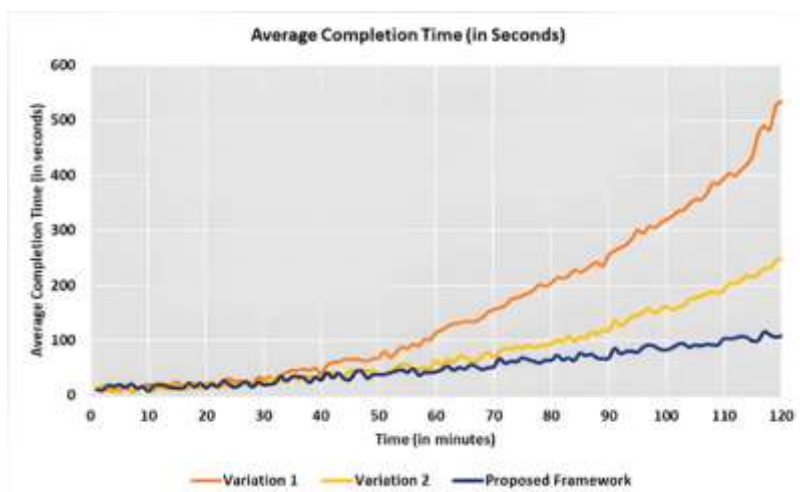
Figure 5.13 shows the AWT for application processes. The proposed framework has significantly reduced waiting times compared to alternative variations. By the end of the trial, Variation-1's waiting time was 370 seconds, Variation-2's was 150 seconds, and the recommended framework's was 50 seconds. Variation-2 had a slower response time than Variation-1, but both exceeded the required framework time of 100 seconds. By the end of the trial, Variation-1's response time was 6.5 times longer than the proposed framework.



**Figure 5.13:** AWT Comparison

The experimental testbed for Fog scheduling considered all significant factors, with fascinating outcomes. Figure 5.9 shows that Image and Mandelbort generate more jobs than other applications. The proposed framework consumes more resources than alternatives but is more efficient. Resource utilization remains consistent after an initial increase. The proposed framework is 20% better than variant 1 and 5-10% better than variant 2 during the trial.

Figure 15.14 shows that the proposed framework performs significantly faster than other versions. Variation-1 shows a sharp rise in completion time, which continues throughout the experiment. Variation-2 has a slower response time than Variation-1, but still slower than the proposed framework. At the end of the trial, Variation-1's response time was 5.5 times longer than that of the proposed framework, whose ACT was 100 seconds.



**Figure 5.14:** ACT Comparison

Table 4.4 summarizes the discussion on the experimental results, comparing all three approaches discussed in this section.

**Table 5.5:** Summary of obtained results.

		<b>ARU</b>	<b>RAV</b>	<b>ART</b>	<b>AWT</b>	<b>ACT</b>
<b>Variation-1</b>		70%	30%	145 sec	370 sec	530 sec
<b>Variation-2</b>		75%	20%	96 sec	148 sec	240 sec
<b>Proposed Framework</b>		84%	18%	48 sec	50 sec	100 sec
<b>Improvement</b>	<b>Variation-1</b>	<b>14%</b>	<b>12%</b>	<b>97 sec</b>	<b>320 sec</b>	<b>430 sec</b>
	<b>Variation-2</b>	<b>9%</b>	<b>2%</b>	<b>48 sec</b>	<b>98 sec</b>	<b>29 sec</b>

**Table 5.6:** Proposed Framework vs Other Solutions

<b>Ref No.</b>	<b>Title</b>	<b>Goal</b>	<b>Limitations</b>
[69]	Distributed computational load balancing for real-time applications	A solution in which scheduling decisions are made by linear programming and a network of queues	1. Missing brokering, 2. Missing hierarchical distribution of load balancing tasks, 3. QoS parameters not considered, 4. No clustering of jobs and resources
[75]	DOTS: Delay-Optimal Task Scheduling Among Voluntary Nodes in Fog Networks	A general analytical model of task scheduling that divides FNs into two categories, which are	1. Missing brokering, 2. Missing hierarchical distribution of load balancing tasks, 3. QoS parameters not considered, 4.No ML technique is used

		voluntary nodes (VN) and task nodes (TN)	
[76]	Folo: Latency and Quality Optimized Task Allocation in Vehicular Fog Computing	A dynamic, event-triggered framework for task allocation using linear programming-based optimization and binary particle swarm optimization	1. Missing brokering, 2. Missing hierarchical distribution of load balancing tasks, 3. QoS parameters not considered, 4.No ML technique is used, 5.No clustering of jobs and resources
[80]	Ranking Fog Nodes for Tasks Scheduling in Fog-Cloud Environments: A Fuzzy Logic Approach	A linguistic and fuzzy quantified proposition for ranking the FNs from most satisfactory to least satisfactory ranking	1. Missing brokering, 2. Missing hierarchical distribution of load balancing tasks, 3. QoS parameters not considered, 4. No clustering of jobs and resources

## 5.5 Summary

The proposed framework offers an efficient QoS-based scheduling approach for IoT application jobs on reachable FNs. Hexagonal FEs cover all potential FNs in the network, and every hexagonal GR has a predetermined number of resources. The PP, BW, and QD in a particular FN are the variables used to calculate each FN's QoS score. Additionally, an ALTERNATE node is given the job only if it satisfies the deadline for execution and has the necessary resources. A job can only be successfully finished if it is purposefully assigned to the most advantageous combination of processing resources, so Fog is mainly employed to fulfill time-sensitive tasks.

Our research focuses on ensuring successful broker communication without involving networking concepts. Improvements can be made by considering various factors that could affect the Quality of Service (QoS) during the execution of jobs for Fog applications. One such improvement could be to utilize a flexible set of QoS criteria that aligns with the type and specifications of each application job. Additionally, the framework could benefit from including scalability as another feature.

# **CHAPTER 6**

## **CONCLUSION AND FUTURE SCOPE**

This thesis aims to study and propose the usage of broker-based load balancing and job scheduling techniques in DSs, mainly CC and FC. As a result of this study, three frameworks are proposed and implemented to achieve efficient resource allocation to the available set of resources.

### **6.1 Thesis Summary**

Chapter 1 introduces this thesis's concept, background, open issues, and objectives. Chapter 2 presents the existing literature describing the methodologies and techniques proposed by the authors in load balancing and job scheduling in the fields of CC and FC. This chapter is divided into multiple sections, each discussing the literature details directly related to one of the objectives. Chapter 3 discusses an SA-based broker framework for load balancing in a PPC where SAs are utilized to implement the autonomous behavior in load-balancing brokers. Chapter 4 discusses an ML classification-based dual broker framework by integrating QoS parameters for scheduling applications' jobs in FEs. In this framework, FNs are segregated based on resource availability, and jobs are mapped on the appropriate set of resources. Chapter 5 discusses a multilevel hierarchical broker for job scheduling in hexagonal FEs for better global geographical coverage of distributed and ad-hoc Fog services based upon the QoS Score and average execution time of the FNs.

### **6.2 Concluding Remarks**

This chapter summarizes the results obtained from Chapters 3, 4 and 5. These chapters contain a detailed description of the proposed frameworks, experimental setup, and results obtained.

#### **6.2.1 A SA-based broker framework for load balancing in a PPC**

To reduce the search space in the Public Cloud (PC), to reduce the congestion at a single job scheduler, and to insert autonomous behavior in the load balancing technique, an SA-based broker framework is introduced for a single Cloud, which resulted in a reduced number of detained jobs, improved makespan, better average execution time (AET), maximum execution time (MET) and total execution time (TET). This method yielded good outcomes, but it can be

enhanced by incorporating machine learning (ML) classification methods for automated decision-making and considering Quality of Service (QoS) parameters when evaluating system performance.

### **6.2.2 ML Classification-based dual broker framework with integrating QoS parameters**

An ML classification-based dual broker framework is proposed by integrating QoS parameters. Two brokers are used: Global Broker (GBR) deployed at the centralized level and Local Brokers (LBRs) deployed at each FE. This framework considers three QoS parameters: Physical Proximity (PP), Bandwidth (BW), and Device Quality (QD). A combination of Naïve Bayes Theorem and Self-Organizing Map (SOM) is used in this approach to classify and map jobs on a suitable set of resources (Generic, Compute Intensive, GPU Intensive, and Memory Intensive). This approach improves average resource utilization (ARU), resource availability (RAV), average response time (ART), average waiting time (AWT), and average completion time (ACT). However, this approach does not address geographical resource coverage, due to which some of the pervasively scattered nodes may be left unattended or overlapped. Congestion issues may still arise at dual-level hierarchical brokers when considering globally scattered Fog resources.

### **6.2.3 Multilevel hierarchical broker for job scheduling in hexagonal FEs for better global geographical coverage of distributed and ad-hoc Fog services**

The third approach is a multilevel hierarchical broker-based framework for job scheduling in hexagonal FEs for better global geographical coverage of distributed and ad-hoc Fog services. The RoI (Region of Interest) in this chapter is divided into hexagonal regions due to the equal distance between their centers and ensuring complete coverage of FNs. This approach considers global-level Fog resources across the Geographical Regions (GRs) and is flexible enough to add multiple hierarchical brokers to manage the increased congestion along with the expansion to multiple GRs. As an output of this approach, we could achieve higher RAV, ARU, ART, AWT, and ACT.



## 6.3 Contribution

This section discusses the significant contributions of the research as part of the thesis.

- **Reduced Search Space**

PC cloud accommodated a large number of resources, which raised manageability issues. In PPC, the main objective is to reduce the search space for load balancing and job allocation. Deploying independent brokers can further speed up the load-balancing tasks.

- **Autonomous Behavior**

SAs possess a unique property of adapting to the behavior of the environment where they are deployed. It adds autonomous behavior to the job scheduling entities, consequently automating the load balancing tasks as per the workload existing in a DS at a time.

- **Improved Average Resource Utilization**

Framework-2 produces 5% and -10 % improved resource utilization compared to [16] and [80]. Framework-3 produces 14% and 9% improved resource utilization compared to Variation-1 and Variation-2.

- **Improved Average Resource Availability**

Framework-2 produces 3% and -10 % resource utilization compared to [16] and [80]. Framework-3 produces 12% and 2% improved resource availability compared to variation-1 and variation-2, respectively.

- **Reduced Average Response Time**

Framework-2 responded 7 sec and 22 sec faster than [16] and [80] respectively. Framework-3 responds 97 sec and 48 sec faster than variation-1 and variation-2, respectively.

- **Reduced Average Waiting Time**

Framework-2 reduced average waiting time by 5 sec and 10 sec compared to [16] [80], respectively. Framework-3 reduced average waiting time by 320 sec and 98 sec compared to variation-1 and variation-2, respectively.

- **Reduced Average Completion Time**

Framework-2 reduced average completion time by 11 sec and 29 sec compared to [16] [80], respectively. Framework-3 reduced average completion time by 320 sec and 98 sec compared to variation-1 and variation-2, respectively.

- **Better Geographical Coverage of Resources**

Hexagonal FE structures are known for better geographic coverage because of their equal distance from the radius of their neighboring hexagons. Framework-3 utilizes this property of hexagonal shape to cover the fog resources better, avoiding overlapping or chances of left-out nodes.

## **6.4 Future Scope**

This thesis is focused on service orchestration in the Cloud and FC environments and has proposed frameworks based on chosen parameters only. There is a long list of the network (number of hops, BW, packet loss, network throughput, network delay, network jitter) and infrastructure (CPU count, amount of memory, size of disk space, percentage of CPU, percentage of memory, free disk) related QoS parameters which contribute significantly in service orchestration. [113]. Experimentation can be performed upon this set of resources to verify if the proposed frameworks produce significantly improved results in the expanded set of parameters. More ML techniques can be explored along with expanding the list of parameters, as the same technique cannot benefit every parameter. Another field of future research to explore is to scale the hierarchy levels in a multilevel broker platform where hierarchical levels can be added or removed per the geographical size and resource requirements of fluctuating request volume. Scaling in terms of hexagonal FEs, including their size and number of hexagonal GRs, is also a dimension that can be experimented with.

## **6.5 Summary**

In the end, the research attempts to propose and verify different approaches to achieve better execution timing, better utilization of deployed resources, and reduce load balancing complexities due to the large search space in the Distributed Systems. Improved operational and QoS parameters improve the user experience and the service provider's goodwill. Efficient

resource allocation is the major contributor to the evolution of modern computing paradigms, and this thesis is an attempt to contribute in the same direction.

<b>Broker Based Scheduling Framework for Partitioned Public Cloud</b>	<b>Technique:</b>	Software Agents
	<b>Platform:</b>	CloudSim
	<b>Performance Metrics:</b>	Average Max Execution Time, Average Time Comparison, Total Time Comparison, Jobs Detained, Makespan
<b>Dual Broker Based, QoS Integrated Scheduling Framework for IoT Applications' Job.</b>	<b>Technique:</b>	Naïve Bays Algorithm, Self-Organizing Maps
	<b>Platform:</b>	Aneka
	<b>Performance Metrics:</b>	Average Resource Utilization, Availability of Resources, Average Response Time, Average Waiting Time, Average Completion Time
<b>Multi-level Hierarchical Broker Based Platform for Job Scheduling in Hexagonal Fog Environments</b>	<b>Technique:</b>	Hexagonal Fog Environments
	<b>Platform:</b>	Aneka
	<b>Performance Metrics:</b>	Number of Tasks Generated, Average Resource Utilization, Average Resource Availability, Average Response Time, Average Waiting Time, Average Completion Time

Figure 6.1: Thesis Summary.

## REFERENCES

- [1] R. Vaithiyathan and T. A. Govindharajan, “User preference-based automatic orchestration of web services using a multi-agent,” *Computers & Electrical Engineering*, vol. 45, pp. 68–76, Jul. 2015, doi: 10.1016/j.compeleceng.2015.03.021.
- [2] R. Sandhu, N. Kaur, S. K. Sood, and R. Buyya, “TDRM: tensor-based data representation and mining for healthcare data in cloud computing environments,” *J Supercomput*, vol. 74, no. 2, pp. 592–614, Feb. 2018, doi: 10.1007/s11227-017-2163-y.
- [3] C. Carrión, “Kubernetes Scheduling: Taxonomy, Ongoing Issues and Challenges,” *ACM Comput. Surv.*, vol. 55, no. 7, pp. 1–37, Jul. 2023, doi: 10.1145/3539606.
- [4] N. Greneche, T. Menouer, C. Cérin, and O. Richard, “A Methodology to Scale Containerized HPC Infrastructures in the Cloud,” in *Euro-Par 2022: Parallel Processing*, vol. 13440, J. Cano and P. Trinder, Eds., in Lecture Notes in Computer Science, vol. 13440. , Cham: Springer International Publishing, 2022, pp. 203–217. doi: 10.1007/978-3-031-12597-3\_13.
- [5] N. Greneche and C. Cerin, “Autoscaling of Containerized HPC Clusters in the Cloud,” in *2022 IEEE/ACM International Workshop on Interoperability of Supercomputing and Cloud Technologies (SuperCompCloud)*, Dallas, TX, USA: IEEE, Nov. 2022, pp. 1–7. doi: 10.1109/SuperCompCloud56703.2022.00006.
- [6] B. Sharma, R. Prabhakar, S.-H. Lim, M. T. Kandemir, and C. R. Das, “MROrchestrator: A Fine-Grained Resource Orchestration Framework for MapReduce Clusters,” in *2012 IEEE Fifth International Conference on Cloud Computing*, Honolulu, HI, USA: IEEE, Jun. 2012, pp. 1–8. doi: 10.1109/CLOUD.2012.37.
- [7] M. Firoz Ali, R. Zaman Khan, *Distributed Computing: An Overview*. Int. J. Advanced Networking and Applications, 2015.
- [8] M. Van Steen and A. S. Tanenbaum, “A brief introduction to distributed systems,” *Computing*, vol. 98, no. 10, pp. 967–1009, Oct. 2016, doi: 10.1007/s00607-016-0508-7.
- [9] A. D. Kshemkalyani and M. Singhal, *Distributed Computing: Principles, Algorithms, and Systems*, 1st ed. Cambridge University Press, 2008. doi: 10.1017/CBO9780511805318.
- [10] A. S. Tanenbaum and M. van Steen, *Distributed systems: principles and paradigms*, Second edition, Adjusted for digital publishing. The Netherlands? Maarten van Steen, 2016.
- [11] I. Ahmad, A. Ghafoor, and G. C. Fox, “Hierarchical Scheduling of Dynamic Parallel Computations on Hypercube Multicomputers,” *Journal of Parallel and Distributed Computing*, vol. 20, no. 3, pp. 317–329, Mar. 1994, doi: 10.1006/jpdc.1994.1030.
- [12] A. Grama, Ed., *Introduction to parallel computing*, 2. ed., [Reprint.]. Harlow: Pearson, 2011.
- [13] K. A. Nuaimi, N. Mohamed, M. A. Nuaimi, and J. Al-Jaroodi, “A Survey of Load Balancing in Cloud Computing: Challenges and Algorithms,” in *2012 Second Symposium on Network Cloud Computing and Applications*, London, United Kingdom: IEEE, Dec. 2012, pp. 137–142. doi: 10.1109/NCCA.2012.29.
- [14] Junjie Pang Gaochao Xu and Xiaodong Fu, “A load balancing model based on cloud partitioning for the public cloud,” *TSINGHUA SCIENCE AND TECHNOLOGY*, pp. 34–39, 2013.
- [15] D. Mohandass, S. R, and R. R, “A novel approach for dynamic cloud partitioning and load balancing in cloud computing environment,” *Journal of Theoretical and Applied Information Technology*, pp. 662–667, 2014.

- [16] R. Sandhu and S. K. Sood, "Scheduling of big data applications on distributed cloud based on QoS parameters," *Cluster Comput.*, vol. 18, no. 2, pp. 817–828, Jun. 2015, doi: 10.1007/s10586-014-0416-6.
- [17] R. Deng, R. Lu, C. Lai, T. H. Luan, and H. Liang, "Optimal Workload Allocation in Fog-Cloud Computing Towards Balanced Delay and Power Consumption," *IEEE Internet Things J.*, pp. 1–1, 2016, doi: 10.1109/JIOT.2016.2565516.
- [18] S. Tuli, R. Sandhu, and R. Buyya, "Shared data-aware dynamic resource provisioning and task scheduling for data intensive applications on hybrid clouds using Aneka," *Future Generation Computer Systems*, vol. 106, pp. 595–606, May 2020, doi: 10.1016/j.future.2020.01.038.
- [19] H. Tran-Dang, S. Bhardwaj, T. Rahim, A. Musaddiq, and D.-S. Kim, "Reinforcement learning based resource management for fog computing environment: Literature review, challenges, and open issues," *J. Commun. Netw.*, vol. 24, no. 1, pp. 83–98, Feb. 2022, doi: 10.23919/JCN.2021.000041.
- [20] Z. J. Al-Araji, S. S. S. Ahmad, N. Kausar, A. Farhani, E. Ozbilge, and T. Cagin, "Fuzzy Theory in Fog Computing: Review, Taxonomy, and Open Issues," *IEEE Access*, vol. 10, pp. 126931–126956, 2022, doi: 10.1109/ACCESS.2022.3225462.
- [21] D. H. Abdulazeez and S. K. Askar, "Offloading Mechanisms Based on Reinforcement Learning and Deep Learning Algorithms in the Fog Computing Environment," *IEEE Access*, vol. 11, pp. 12555–12586, 2023, doi: 10.1109/ACCESS.2023.3241881.
- [22] H. Tran-Dang, K.-H. Kwon, and D.-S. Kim, "Bandit Learning-Based Distributed Computation in Fog Computing Networks: A Survey," *IEEE Access*, vol. 11, pp. 104763–104774, 2023, doi: 10.1109/ACCESS.2023.3314889.
- [23] T.-A. N. Abdali, R. Hassan, A. H. M. Aman, and Q. N. Nguyen, "Fog Computing Advancement: Concept, Architecture, Applications, Advantages, and Open Issues," *IEEE Access*, vol. 9, pp. 75961–75980, 2021, doi: 10.1109/ACCESS.2021.3081770.
- [24] U. Mir, U. Abbasi, T. Mir, S. Kanwal, and S. Alamri, "Energy Management in Smart Buildings and Homes: Current Approaches, a Hypothetical Solution, and Open Issues and Challenges," *IEEE Access*, vol. 9, pp. 94132–94148, 2021, doi: 10.1109/ACCESS.2021.3092304.
- [25] S. Khan, T. Arslan, and T. Ratnarajah, "Digital Twin Perspective of Fourth Industrial and Healthcare Revolution," *IEEE Access*, vol. 10, pp. 25732–25754, 2022, doi: 10.1109/ACCESS.2022.3156062.
- [26] Science and Research Branch, Islamic Azad University, Tehran, Iran, M. Mesbahi, and A. Masoud Rahmani, "Load Balancing in Cloud Computing: A State of the Art Survey," *IJMECS*, vol. 8, no. 3, pp. 64–78, Mar. 2016, doi: 10.5815/ijmecs.2016.03.08.
- [27] Shavan Askar, Kurdistan Ali, and T. A. Rashid, "Fog Computing Based IoT System: A Review," Aug. 2021, doi: 10.5281/ZENODO.5222392.
- [28] S. D. J. Sunaina, "Service Broker Policy Algorithm for Logistics over Cloud," *International Journal of Innovations in Engineering and Technology*, vol. 7, pp. 2319–1058.
- [29] G. Pierre and M. van Steen, "Globule: a collaborative content delivery network," *IEEE Commun. Mag.*, vol. 44, no. 8, pp. 127–133, Aug. 2006, doi: 10.1109/MCOM.2006.1678120.
- [30] B. Beverly Yang and H. Garcia-Molina, "Designing a super-peer network," in *Proceedings 19th International Conference on Data Engineering (Cat. No.03CH37405)*, Bangalore, India: IEEE, 2003, pp. 49–60. doi: 10.1109/ICDE.2003.1260781.

- [31] F. Li, B. Luo, P. Liu, D. Lee, and C.-H. Chu, “Enforcing Secure and Privacy-Preserving Information Brokering in Distributed Information Sharing,” *IEEE Trans. Inform. Forensic Secur.*, vol. 8, no. 6, pp. 888–900, Jun. 2013, doi: 10.1109/TIFS.2013.2247398.
- [32] H. Benitez Perez, I. Lopez Juarez, P. C. Garza Alanis, R. Rios Cabrera, and A. Duran Chavesti, “Reconfiguration Distributed Objects in an Intelligent Manufacturing Cell,” *IEEE Latin Am. Trans.*, vol. 14, no. 1, pp. 136–146, Jan. 2016, doi: 10.1109/TLA.2016.7430073.
- [33] N. Ferdosian, M. Othman, B. M. Ali, and K. Y. Lun, “Fair-QoS Broker Algorithm for Overload-State Downlink Resource Scheduling in LTE Networks,” *IEEE Systems Journal*, vol. 12, no. 4, pp. 3238–3249, Dec. 2018, doi: 10.1109/JSYST.2017.2702109.
- [34] K. Hwang, J. M. Lee, I. H. Jung, and D.-H. Lee, “Modification of Mosquito Broker for Delivery of Urgent MQTT Message,” in *2019 IEEE Eurasia Conference on IOT, Communication and Engineering (ECICE)*, Yunlin, Taiwan: IEEE, Oct. 2019, pp. 166–167. doi: 10.1109/ECICE47484.2019.8942800.
- [35] J. Mei, K. Li, Z. Tong, Q. Li, and K. Li, “Profit Maximization for Cloud Brokers in Cloud Computing,” *IEEE Trans. Parallel Distrib. Syst.*, vol. 30, no. 1, pp. 190–203, Jan. 2019, doi: 10.1109/TPDS.2018.2851246.
- [36] H. Oh, S. Park, G. M. Lee, J. K. Choi, and S. Noh, “Competitive Data Trading Model With Privacy Valuation for Multiple Stakeholders in IoT Data Markets,” *IEEE Internet Things J.*, vol. 7, no. 4, pp. 3623–3639, Apr. 2020, doi: 10.1109/JIOT.2020.2973662.
- [37] W. Zhang, X. Li, L. Zhao, X. Yang, T. Liu, and W. Yang, “Service Pricing and Selection for IoT Applications Offloading in the Multi-Mobile Edge Computing Systems,” *IEEE Access*, vol. 8, pp. 153862–153871, 2020, doi: 10.1109/ACCESS.2020.3018166.
- [38] B. Mishra and A. Kertesz, “The Use of MQTT in M2M and IoT Systems: A Survey,” *IEEE Access*, vol. 8, pp. 201071–201086, 2020, doi: 10.1109/ACCESS.2020.3035849.
- [39] A. Sathish, D. Dsouza, K. Ballal, A. M, T. Singh, and G. Monteiro, “Advanced Mechanism to Achieve QoS and Profit Maximization of Brokers in Cloud Computing,” *EAI Endorsed Transactions on Cloud Systems*, p. 170244, Jun. 2021, doi: 10.4108/eai.23-6-2021.170244.
- [40] M. Razian, M. Fathian, H. Wu, A. Akbari, and R. Buyya, “SAIoT: Scalable Anomaly-Aware Services Composition in CloudIoT Environments,” *IEEE Internet Things J.*, vol. 8, no. 5, pp. 3665–3677, Mar. 2021, doi: 10.1109/JIOT.2020.3023938.
- [41] S. Gruener, H. Koziolk, and J. Ruckert, “Towards Resilient IoT Messaging: An Experience Report Analyzing MQTT Brokers,” in *2021 IEEE 18th International Conference on Software Architecture (ICSA)*, Stuttgart, Germany: IEEE, Mar. 2021, pp. 69–79. doi: 10.1109/ICSA51549.2021.00015.
- [42] A. Gruner, A. Muhle, and C. Meinel, “ATIB: Design and Evaluation of an Architecture for Brokered Self-Sovereign Identity Integration and Trust-Enhancing Attribute Aggregation for Service Provider,” *IEEE Access*, vol. 9, pp. 138553–138570, 2021, doi: 10.1109/ACCESS.2021.3116095.
- [43] J. R, A. Urs C J, A. S, D. H M, and S. B K, “Cloud Service Orchestration,” *Journal of Emerging Technologies and Innovative Research*, vol. 8, no. 7, pp. 344–349, 2021.
- [44] S. K. Dhurandher, M. S. Obaidat, I. Woungang, P. Agarwal, A. Gupta, and P. Gupta, “A cluster-based load balancing algorithm in cloud computing,” in *2014 IEEE International Conference on Communications (ICC)*, Sydney, NSW: IEEE, Jun. 2014, pp. 2921–2925. doi: 10.1109/ICC.2014.6883768.
- [45] H. Zhang, G. Jiang, K. Yoshihira, and H. Chen, “Proactive Workload Management in Hybrid Cloud Computing,” *IEEE Trans. Netw. Serv. Manage.*, vol. 11, no. 1, pp. 90–100, Mar. 2014, doi: 10.1109/TNSM.2013.122313.130448.

- [46] P. Sharma and P. Singh, "Load degree calculation for the public cloud based on cloud partitioning model using turnaround time," *International Journal of Computer Science and Information Technologies*, 2015.
- [47] S. Sebastio and A. Scala, "A Workload-Based Approach to Partition the Volunteer Cloud," in *2015 IEEE Conference on Collaboration and Internet Computing (CIC)*, Hangzhou, China: IEEE, Oct. 2015, pp. 210–218. doi: 10.1109/CIC.2015.27.
- [48] A. Nahir, A. Orda, and D. Raz, "Replication-Based Load Balancing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 27, no. 2, pp. 494–507, Feb. 2016, doi: 10.1109/TPDS.2015.2400456.
- [49] M. Pantazoglou, G. Tzortzakis, and A. Delis, "Decentralized and Energy-Efficient Workload Management in Enterprise Clouds," *IEEE Trans. Cloud Comput.*, vol. 4, no. 2, pp. 196–209, Apr. 2016, doi: 10.1109/TCC.2015.2464817.
- [50] A. Lakhan, Q.-U.-A. Mastoi, M. Elhoseny, M. S. Memon, and M. A. Mohammed, "Deep neural network-based application partitioning and scheduling for hospitals and medical enterprises using IoT assisted mobile fog cloud," *Enterprise Information Systems*, vol. 16, no. 7, p. 1883122, Jul. 2022, doi: 10.1080/17517575.2021.1883122.
- [51] A. C. Zhou, J. Luo, R. Qiu, H. Tan, B. He, and R. Mao, "Adaptive Partitioning for Large-Scale Graph Analytics in Geo-Distributed Data Centers," in *2022 IEEE 38th International Conference on Data Engineering (ICDE)*, Kuala Lumpur, Malaysia: IEEE, May 2022, pp. 2818–2830. doi: 10.1109/ICDE53745.2022.00256.
- [52] Y. Xiao, Y. Yao, K. Chen, W. Tang, and F. Zhu, "A simulation task partition method based on cloud computing resource prediction using ensemble learning," *Simulation Modelling Practice and Theory*, vol. 119, p. 102595, Sep. 2022, doi: 10.1016/j.simpat.2022.102595.
- [53] D. Lan *et al.*, "Task Partitioning and Orchestration on Heterogeneous Edge Platforms: The Case of Vision Applications," *IEEE Internet Things J.*, vol. 9, no. 10, pp. 7418–7432, May 2022, doi: 10.1109/JIOT.2022.3153970.
- [54] S. T. Selvi, C. Valliyammai, and V. N. Dhatchayani, "Resource allocation issues and challenges in cloud computing," in *2014 International Conference on Recent Trends in Information Technology*, Chennai, India: IEEE, Apr. 2014, pp. 1–6. doi: 10.1109/ICRTIT.2014.6996213.
- [55] J. O. Gutierrez-Garcia and A. Ramirez-Nafarrate, "Collaborative Agents for Distributed Load Management in Cloud Data Centers Using Live Migration of Virtual Machines," *IEEE Trans. Serv. Comput.*, vol. 8, no. 6, pp. 916–929, Nov. 2015, doi: 10.1109/TSC.2015.2491280.
- [56] R. Thapa, L. Jiao, B. J. Oommen, and A. Yazidi, "A Learning Automaton-Based Scheme for Scheduling Domestic Shiftable Loads in Smart Grids," *IEEE Access*, vol. 6, pp. 5348–5361, 2018, doi: 10.1109/ACCESS.2017.2788051.
- [57] I. Azimi *et al.*, "HiCH: Hierarchical Fog-Assisted Computing Architecture for Healthcare IoT," *ACM Trans. Embed. Comput. Syst.*, vol. 16, no. 5s, pp. 1–20, Oct. 2017, doi: 10.1145/3126501.
- [58] R. Mahmud, K. Ramamohanarao, and R. Buyya, "Latency-Aware Application Module Management for Fog Computing Environments," *ACM Trans. Internet Technol.*, vol. 19, no. 1, pp. 1–21, Feb. 2019, doi: 10.1145/3186592.
- [59] J. Wan, B. Chen, S. Wang, M. Xia, D. Li, and C. Liu, "Fog Computing for Energy-Aware Load Balancing and Scheduling in Smart Factory," *IEEE Trans. Ind. Inf.*, vol. 14, no. 10, pp. 4548–4556, Oct. 2018, doi: 10.1109/TII.2018.2818932.

- [60] M. Soualhia, F. Khomh, and S. Tahar, "A Dynamic and Failure-Aware Task Scheduling Framework for Hadoop," *IEEE Trans. Cloud Comput.*, vol. 8, no. 2, pp. 553–569, Apr. 2020, doi: 10.1109/TCC.2018.2805812.
- [61] J. Zhu, Y. Song, D. Jiang, and H. Song, "A New Deep-Q-Learning-Based Transmission Scheduling Mechanism for the Cognitive Internet of Things," *IEEE Internet Things J.*, vol. 5, no. 4, pp. 2375–2385, Aug. 2018, doi: 10.1109/JIOT.2017.2759728.
- [62] Y. Wei, L. Pan, S. Liu, L. Wu, and X. Meng, "DRL-Scheduling: An Intelligent QoS-Aware Job Scheduling Framework for Applications in Clouds," *IEEE Access*, vol. 6, pp. 55112–55125, 2018, doi: 10.1109/ACCESS.2018.2872674.
- [63] X. Chen, S. Leng, K. Zhang, and K. Xiong, "A machine-learning based time constrained resource allocation scheme for vehicular fog computing," *China Commun.*, vol. 16, no. 11, pp. 29–41, Nov. 2019, doi: 10.23919/JCC.2019.11.003.
- [64] G. Zhu, J. Zan, Y. Yang, and X. Qi, "A Supervised Learning Based QoS Assurance Architecture for 5G Networks," *IEEE Access*, vol. 7, pp. 43598–43606, 2019, doi: 10.1109/ACCESS.2019.2907142.
- [65] G. Henri and N. Lu, "A Supervised Machine Learning Approach to Control Energy Storage Devices," *IEEE Trans. Smart Grid*, vol. 10, no. 6, pp. 5910–5919, Nov. 2019, doi: 10.1109/TSG.2019.2892586.
- [66] Y. Zhao, Y. Li, X. Zhang, G. Geng, W. Zhang, and Y. Sun, "A Survey of Networking Applications Applying the Software Defined Networking Concept Based on Machine Learning," *IEEE Access*, vol. 7, pp. 95397–95417, 2019, doi: 10.1109/ACCESS.2019.2928564.
- [67] M. M. Amiri and D. Gunduz, "Computation Scheduling for Distributed Machine Learning with Straggling Workers," in *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Brighton, United Kingdom: IEEE, May 2019, pp. 8177–8181. doi: 10.1109/ICASSP.2019.8682911.
- [68] D. Zeng, L. Gu, S. Guo, Z. Cheng, and S. Yu, "Joint Optimization of Task Scheduling and Image Placement in Fog Computing Supported Software-Defined Embedded System," *IEEE Trans. Comput.*, vol. 65, no. 12, pp. 3702–3712, Dec. 2016, doi: 10.1109/TC.2016.2536019.
- [69] S. Sthapit, J. R. Hopgood, and J. Thompson, "Distributed computational load balancing for real-time applications," in *2017 25th European Signal Processing Conference (EUSIPCO)*, Kos, Greece: IEEE, Aug. 2017, pp. 1385–1189. doi: 10.23919/EUSIPCO.2017.8081436.
- [70] D. Rahbari, S. Kabirzadeh, and M. Nickray, "A security aware scheduling in fog computing by hyper heuristic algorithm," in *2017 3rd Iranian Conference on Intelligent Systems and Signal Processing (ICSPIS)*, Shahrood: IEEE, Dec. 2017, pp. 87–92. doi: 10.1109/ICSPIS.2017.8311595.
- [71] Y.-C. Chen, Y.-C. Chang, C.-H. Chen, Y.-S. Lin, J.-L. Chen, and Y.-Y. Chang, "Cloud-fog computing for information-centric Internet-of-Things applications," in *2017 International Conference on Applied System Innovation (ICASI)*, Sapporo, Japan: IEEE, May 2017, pp. 637–640. doi: 10.1109/ICASI.2017.7988506.
- [72] Y. Yang, S. Zhao, W. Zhang, Y. Chen, X. Luo, and J. Wang, "DEBTS: Delay Energy Balanced Task Scheduling in Homogeneous Fog Networks," *IEEE Internet Things J.*, vol. 5, no. 3, pp. 2094–2106, Jun. 2018, doi: 10.1109/JIOT.2018.2823000.
- [73] L. Yin, J. Luo, and H. Luo, "Tasks Scheduling and Resource Allocation in Fog Computing Based on Containers for Smart Manufacturing," *IEEE Trans. Ind. Inf.*, vol. 14, no. 10, pp. 4712–4721, Oct. 2018, doi: 10.1109/TII.2018.2851241.



- [74] O. Casquero, A. Armentia, I. Sarachaga, F. Perez, D. Orive, and M. Marcos, “Distributed scheduling in Kubernetes based on MAS for Fog-in-the-loop applications,” in *2019 24th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, Zaragoza, Spain: IEEE, Sep. 2019, pp. 1213–1217. doi: 10.1109/ETFA.2019.8869219.
- [75] G. Zhang, F. Shen, N. Chen, P. Zhu, X. Dai, and Y. Yang, “DOTS: Delay-Optimal Task Scheduling Among Voluntary Nodes in Fog Networks,” *IEEE Internet Things J.*, vol. 6, no. 2, pp. 3533–3544, Apr. 2019, doi: 10.1109/JIOT.2018.2887264.
- [76] C. Zhu *et al.*, “Folo: Latency and Quality Optimized Task Allocation in Vehicular Fog Computing,” *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4150–4161, Jun. 2019, doi: 10.1109/JIOT.2018.2875520.
- [77] S. F. Abedin, A. K. Bairagi, Md. S. Munir, N. H. Tran, and C. S. Hong, “Fog Load Balancing for Massive Machine Type Communications: A Game and Transport Theoretic Approach,” *IEEE Access*, vol. 7, pp. 4204–4218, 2019, doi: 10.1109/ACCESS.2018.2888869.
- [78] S. Sthapit, J. Thompson, N. M. Robertson, and J. R. Hopgood, “Computational Load Balancing on the Edge in Absence of Cloud and Fog,” *IEEE Trans. on Mobile Comput.*, vol. 18, no. 7, pp. 1499–1512, Jul. 2019, doi: 10.1109/TMC.2018.2863301.
- [79] G. L. Stavrinides and H. D. Karatza, “Cost-Effective Utilization of Complementary Cloud Resources for the Scheduling of Real-Time Workflow Applications in a Fog Environment,” in *2019 7th International Conference on Future Internet of Things and Cloud (FiCloud)*, Istanbul, Turkey: IEEE, Aug. 2019, pp. 1–8. doi: 10.1109/FiCloud.2019.00009.
- [80] M. A. Benblidia, B. Brik, L. Merghem-Boulaiah, and M. Esseghir, “Ranking Fog nodes for Tasks Scheduling in Fog-Cloud Environments: A Fuzzy Logic Approach,” in *2019 15th International Wireless Communications & Mobile Computing Conference (IWCMC)*, Tangier, Morocco: IEEE, Jun. 2019, pp. 1451–1457. doi: 10.1109/IWCMC.2019.8766437.
- [81] J.-P. Yang, “Elastic Load Balancing Using Self-Adaptive Replication Management,” *IEEE Access*, vol. 5, pp. 7495–7504, 2017, doi: 10.1109/ACCESS.2016.2631490.
- [82] P. Zhao, W. Yu, S. Yang, X. Yanga, and J. Lin, “On Minimizing Energy Cost in Internet-scale Systems with Dynamic Data,” *IEEE Access*, pp. 1–1, 2017, doi: 10.1109/ACCESS.2017.2725939.
- [83] Z. Qiu and J. F. Perez, “Evaluating Replication for Parallel Jobs: An Efficient Approach,” *IEEE Trans. Parallel Distrib. Syst.*, vol. 27, no. 8, pp. 2288–2302, Aug. 2016, doi: 10.1109/TPDS.2015.2496593.
- [84] J.-P. Yang, “Intelligent Offload Detection for Achieving Approximately Optimal Load Balancing,” *IEEE Access*, vol. 6, pp. 58609–58618, 2018, doi: 10.1109/ACCESS.2018.2873287.
- [85] S. Souravlas and A. Sifaleras, “Trends in data replication strategies: a survey,” *International Journal of Parallel, Emergent and Distributed Systems*, vol. 34, no. 2, pp. 222–239, Mar. 2019, doi: 10.1080/17445760.2017.1401073.
- [86] N. Mostafa, I. Al Ridhawi, and A. Hamza, “An intelligent dynamic replica selection model within grid systems,” in *2015 IEEE 8th GCC Conference & Exhibition*, Muscat, Oman: IEEE, Feb. 2015, pp. 1–6. doi: 10.1109/IEEEGCC.2015.7060061.
- [87] R. Yadav and A. S. Sidhu, “Fault tolerant algorithm for Replication Management in distributed cloud system,” in *2015 IEEE 3rd International Conference on MOOCs, Innovation and Technology in Education (MITE)*, Amritsar, India: IEEE, Oct. 2015, pp. 78–83. doi: 10.1109/MITE.2015.7375292.

- [88] M. Viana, P. Alencar, E. Guimaraes, E. Cirilo, and C. Lucena, “Creating a Modeling Language Based on a New Metamodel for Adaptive Normative Software Agents,” *IEEE Access*, vol. 10, pp. 13974–13996, 2022, doi: 10.1109/ACCESS.2022.3147144.
- [89] A. S. R. M.P. Georgeff, “BDI agents: From theory to practice,” *Proc. ICMAS*, vol. 95, pp. 312–319, 1995.
- [90] N. Huber, A. Van Hoorn, A. Koziolk, F. Brosig, and S. Kounev, “Modeling run-time adaptation at the system architecture level in dynamic service-oriented environments,” *SOCA*, vol. 8, no. 1, pp. 73–89, Mar. 2014, doi: 10.1007/s11761-013-0144-4.
- [91] R. De Lemos *et al.*, “Software Engineering for Self-Adaptive Systems: A Second Research Roadmap,” in *Software Engineering for Self-Adaptive Systems II*, vol. 7475, R. De Lemos, H. Giese, H. A. Müller, and M. Shaw, Eds., in Lecture Notes in Computer Science, vol. 7475. , Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 1–32. doi: 10.1007/978-3-642-35813-5\_1.
- [92] T. Laroum and B. Tighiouart, “A Multi-agent System for the Modelling of the HIV Infection,” in *Agent and Multi-Agent Systems: Technologies and Applications*, vol. 6682, J. O’Shea, N. T. Nguyen, K. Crockett, R. J. Howlett, and L. C. Jain, Eds., in Lecture Notes in Computer Science, vol. 6682. , Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 94–102. doi: 10.1007/978-3-642-22000-5\_11.
- [93] S. H. H. Madni, M. S. Abd Latiff, M. Abdullahi, S. M. Abdulhamid, and M. J. Usman, “Performance comparison of heuristic algorithms for task scheduling in IaaS cloud computing environment,” *PLoS ONE*, vol. 12, no. 5, p. e0176321, May 2017, doi: 10.1371/journal.pone.0176321.
- [94] T. Islam and M. S. Hasan, “A performance comparison of load balancing algorithms for cloud computing,” in *2017 International Conference on the Frontiers and Advances in Data Science (FADS)*, Xi’an: IEEE, Oct. 2017, pp. 130–135. doi: 10.1109/FADS.2017.8253211.
- [95] R. Elavarasi and S. Silas, “Survey on Job Scheduling in Fog Computing,” in *2019 3rd International Conference on Trends in Electronics and Informatics (ICOEI)*, Tirunelveli, India: IEEE, Apr. 2019, pp. 580–583. doi: 10.1109/ICOEI.2019.8862651.
- [96] P. Narayana, P. Parvataneni, and K. Keerthi, “A Research on Various Scheduling Strategies in Fog Computing Environment,” in *2020 International Conference on Emerging Trends in Information Technology and Engineering (ic-ETITE)*, Vellore, India: IEEE, Feb. 2020, pp. 1–6. doi: 10.1109/ic-ETITE47903.2020.261.
- [97] Z. Liu, Q. Zheng, L. Xue, and X. Guan, “A distributed energy-efficient clustering algorithm with improved coverage in wireless sensor networks,” *Future Generation Computer Systems*, vol. 28, no. 5, pp. 780–790, May 2012, doi: 10.1016/j.future.2011.04.019.
- [98] A. Brahmachari, P. Kanti, P. Dutta, S. Pal, and P. Choudhury, “Processing iot data: From cloud to fogits time to be down to earth,” *Research Gate*, pp. 124–148, 2018.
- [99] C. Barros, V. Rocio, A. Sousa, and H. Paredes, “Survey on Job Scheduling in Cloud-Fog Architecture,” in *2020 15th Iberian Conference on Information Systems and Technologies (CISTI)*, Sevilla, Spain: IEEE, Jun. 2020, pp. 1–7. doi: 10.23919/CISTI49556.2020.9141156.
- [100] P. P. G. Gopinath and S. K. Vasudevan, “An In-depth Analysis and Study of Load Balancing Techniques in the Cloud Computing Environment,” *Procedia Computer Science*, vol. 50, pp. 427–432, 2015, doi: 10.1016/j.procs.2015.04.009.
- [101] N. Mohan and J. Kangasharju, “Edge-Fog cloud: A distributed cloud for Internet of Things computations,” in *2016 Cloudification of the Internet of Things (CIoT)*, Paris, France: IEEE, Nov. 2016, pp. 1–6. doi: 10.1109/CIOT.2016.7872914.

- [102] A. Botta, W. de Donato, V. Persico, and A. Pescapé, “Integration of Cloud computing and Internet of Things: A survey,” *Future Generation Computer Systems*, vol. 56, pp. 684–700, Mar. 2016, doi: 10.1016/j.future.2015.09.021.
- [103] L. Atzori, A. Iera, and G. Morabito, “The Internet of Things: A survey,” *Computer Networks*, vol. 54, no. 15, pp. 2787–2805, Oct. 2010, doi: 10.1016/j.comnet.2010.05.010.
- [104] T. Alam, “A reliable communication framework and its use in internet of things (IoT),” *International Journal of Scientific Research in Computer Science, Engineering and Information Technology*, vol. 3, no. 5, pp. 450–456.
- [105] X. Masip-Bruin, E. Marín-Tordera, G. Tashakor, A. Jukan, and G.-J. Ren, “Foggy clouds and cloudy fogs: a real need for coordinated management of fog-to-cloud computing systems,” *IEEE Wireless Commun.*, vol. 23, no. 5, pp. 120–128, Oct. 2016, doi: 10.1109/MWC.2016.7721750.
- [106] X. Chu, K. Nadiminti, C. Jin, S. Venugopal, and R. Buyya, “Aneka: Next-Generation Enterprise Grid Platform for e-Science and e-Business Applications,” in *Third IEEE International Conference on e-Science and Grid Computing (e-Science 2007)*, Bangalore, India: IEEE, 2007, pp. 151–159. doi: 10.1109/E-SCIENCE.2007.12.
- [107] M. A. Benblidia, B. Brik, L. Merghem-Boulahia, and M. Esseghir, “Ranking Fog nodes for Tasks Scheduling in Fog-Cloud Environments: A Fuzzy Logic Approach,” in *2019 15th International Wireless Communications & Mobile Computing Conference (IWCMC)*, Tangier, Morocco: IEEE, Jun. 2019, pp. 1451–1457. doi: 10.1109/IWCMC.2019.8766437.
- [108] H. El-Sayed *et al.*, “Edge of Things: The Big Picture on the Integration of Edge, IoT and the Cloud in a Distributed Computing Environment,” *IEEE Access*, vol. 6, pp. 1706–1717, 2018, doi: 10.1109/ACCESS.2017.2780087.
- [109] B. Ali, M. Adeel Pasha, S. ul Islam, H. Song, and R. Buyya, “A Volunteer-Supported Fog Computing Environment for Delay-Sensitive IoT Applications,” *IEEE Internet Things J.*, vol. 8, no. 5, pp. 3822–3830, Mar. 2021, doi: 10.1109/JIOT.2020.3024823.
- [110] S. Misra, S. Chatterjee, and M. S. Obaidat, “On Theoretical Modeling of Sensor Cloud: A Paradigm Shift From Wireless Sensor Network,” *IEEE Systems Journal*, vol. 11, no. 2, pp. 1084–1093, Jun. 2017, doi: 10.1109/JSYST.2014.2362617.
- [111] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, “Fog computing and its role in the internet of things,” in *Proceedings of the first edition of the MCC workshop on Mobile cloud computing - MCC '12*, Helsinki, Finland: ACM Press, 2012, p. 13. doi: 10.1145/2342509.2342513.
- [112] S. K. Sood, R. Sandhu, K. Singla, and V. Chang, “IoT, big data and HPC based smart flood management framework,” *Sustainable Computing: Informatics and Systems*, vol. 20, pp. 102–117, Dec. 2018, doi: 10.1016/j.suscom.2017.12.001.
- [113] S. Taherizadeh, V. Stankovski, and M. Grobelnik, “A Capillary Computing Architecture for Dynamic Internet of Things: Orchestration of Microservices from Edge Devices to Fog and Cloud Providers,” *Sensors*, vol. 18, no. 9, p. 2938, Sep. 2018, doi: 10.3390/s18092938.

## List of Publications

### JOURNAL PUBLICATIONS

- [1] M. Kaur and R. Mohana, “Static Load Balancing Technique for Geographically Partitioned Public Cloud,” *SCPE*, vol. 20, no. 2, pp. 299–316, May 2019, doi: 10.12694/scpe.v20i2.1520.
- [2] M. Kaur, R. Sandhu, and R. Mohana, “A framework for scheduling IoT application jobs on fog computing infrastructure based on QoS parameters,” *IJPCC*, Oct. 2021, doi: 10.1108/IJPCC-08-2020-0108.
- [3] Kaur, M., Sandhu, R. & Mohana, R. A Framework for QoS Parameters-Based Scheduling for IoT Applications on Fog Environments. *Wireless Pers Commun* **132**, 2709–2736 (2023). <https://doi.org/10.1007/s11277-023-10740-6>

### CONFERENCE PUBLICATIONS

- [4] M. Kaur and Dr. Rajni Mohana, “A Software Agent Based Technique for Load Balancing in Partitioned Cloud,” *IJET*, vol. 7, no. 4.12, p. 13, Oct. 2018, doi: 10.14419/ijet.v7i4.12.20984.
- [5] M. Kaur, R. Sandhu, and R. Mohana, “Fog Load Balancing Broker (FLBB),” in *2021 Sixth International Conference on Image Information Processing (ICIIP)*, Shimla, India: IEEE, Nov. 2021, pp. 332–337. doi: 10.1109/ICIIP53038.2021.9702669.

# SYNOPSIS

## 1. Introduction

This thesis addresses the complexities of orchestrating services in Cloud and Fog computing environments by proposing Broker-Based Frameworks (BBFs). These frameworks leverage intelligent brokering strategies to optimize resource management, service orchestration, and deployment in heterogeneous computing environments. The thesis outlines the architecture and components of the BBF, emphasizing its potential to enhance responsiveness, resource efficiency, resource utilization, and resource availability in Distributed Computing Systems. Overall, the proposed BBFs offer a solution-oriented approach to the challenges associated with service orchestration in dispersed and dynamic computing ecosystems.

### 1.1 Orchestration

The term orchestration describes managing and synchronizing services, apps, and resources in Cloud and Fog Computing (FC) contexts. Cloud orchestration is the process of overseeing services and resources in a distant, centralized data center. It usually manages storage, networking, Virtual Machines (VMs), containers, and other Cloud infrastructure resources. Applications in Cloud settings can be automated, managed, and scaled using orchestration tools such as OpenStack, Docker Swarm, and Kubernetes. FC involves orchestrating computing, storage, and networking resources in edge devices, gateways, and local servers. Orchestration tools such as Cisco IOx, OpenFog, or edge computing (EC) platforms can manage and automate the services in FC environments. [1][2][3]

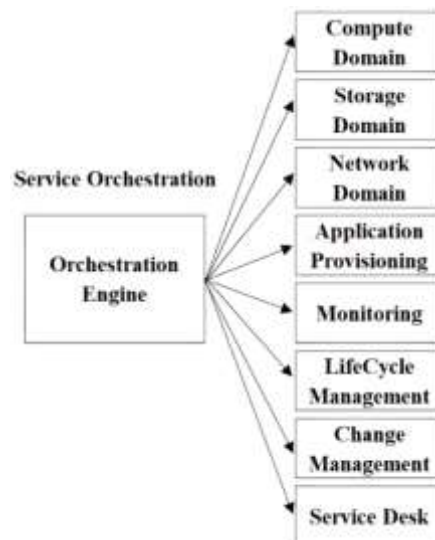
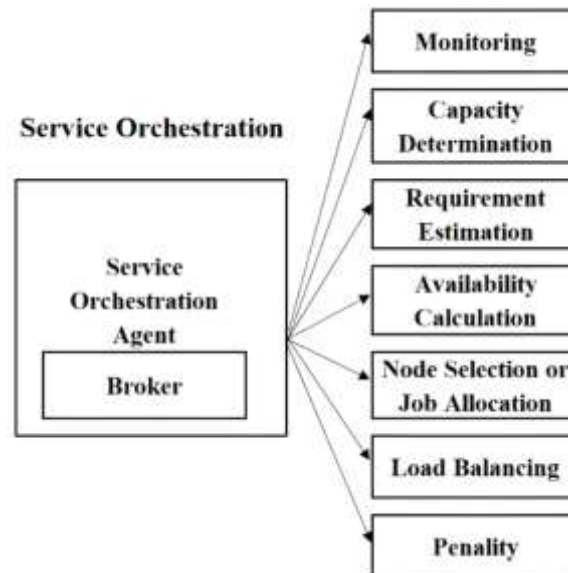


Figure 1: Roles played by the orchestration.

A service orchestration engine plans, directs, and automates the interactions between various services or systems in a network or application environment.



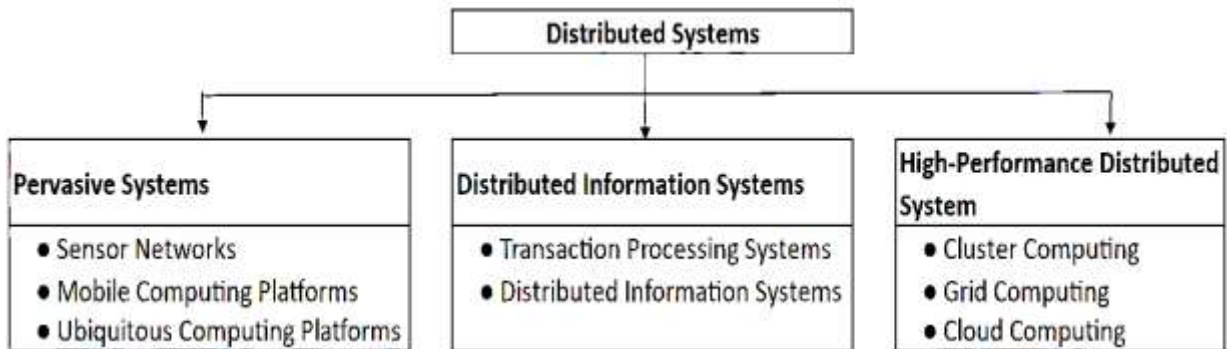
**Figure 2:** Broker as a Service Orchestration Agent

Orchestration is crucial in Cloud Computing (CC) and FC to simplify and automate application and service deployment, scaling, and management. It ensures reliable, scalable, and efficient resource utilization in complex and distributed computing environments. [4][5][6]

## 1.2 Distributed Systems

Distributed Systems (DS) refers to two or more computers working together through a network to finish a single job, which is distributed over a group of computers, heterogeneous in operating systems, node characteristics, network design, and communication medium [7]. The generic quality of DSs in a group of autonomous computer components is that these seem like a single cohesive system to the users [8]. DS comprises autonomous computing nodes for better accessibility, transparency, openness, and scalability [9].

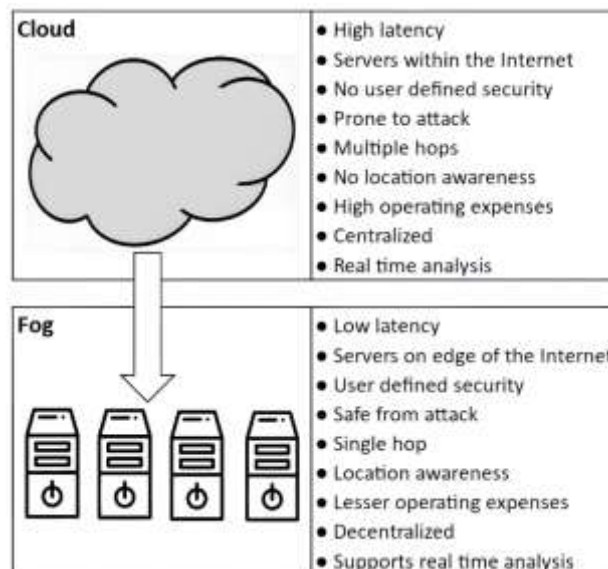
To optimize autonomous communication between nodes, a DS's performance can be measured through execution time, throughput, efficiency, system utilization, turnaround time, waiting time, response time, overheads, and reliability [9][10][11][12]. Figure 3 demonstrates the categories and relevant examples of DS.



**Figure 3:** Categories of distributed systems.

### 1.3 High-Performance Distributed Systems

This category includes computing paradigms such as cluster computing, grid computing, and CC. FC is another paradigm that enables delivering Cloud application services closer to the Internet of Things (IoT) devices at the edge rather than relying on a distance Cloud. In contrast to transferring IoT data to the Cloud, which processes and stores it remotely on IoT computers, Fog provides better latency-sensitive resources. Thus, FC is the ideal choice for enabling the IoT to provide efficient and trustworthy services to many clients. Figure 4 highlights the fundamental differences between Cloud and FC paradigms.



**Figure 4:** Cloud vs Fog computing paradigms.

### 1.4 Load, Load Balancing and Job Scheduling

In the context of CC, load refers to the volume of work or demand on a server, network, or system at a specific time. It mainly concerns how much resource is used or consumed in a Cloud-based infrastructure. Compute, network and storage loads are just a few examples of the components that comprise the term load in distributed systems [13]. Load balancing in DS evenly distributes network traffic or computational workload across multiple servers that optimize resources, scalability, fault tolerance, performance enhancement, and cost efficiency [14] [15]. On the other hand, job scheduling in cloud and fog computing efficiently allocates computing resources and manages task execution to manage resource utilization, prioritization, fairness, adaptability, flexibility, and latency [16][17][18].

### 1.5 Brokers

In Cloud and FC environments, Cloud Brokers (CBs) and Fog Brokers are significant for resource management, communication facilitation, and operational optimization. To help choose and handle Cloud services, Cloud brokerage involves mediators who link consumers or businesses with Cloud service providers. Cloud service brokers (CSBs), such as Right Scale and Gravitant, are instances of CBs. They offer platforms for managing Cloud services from various vendors. Figure 5 demonstrates the positioning and contribution of the brokers in DS.

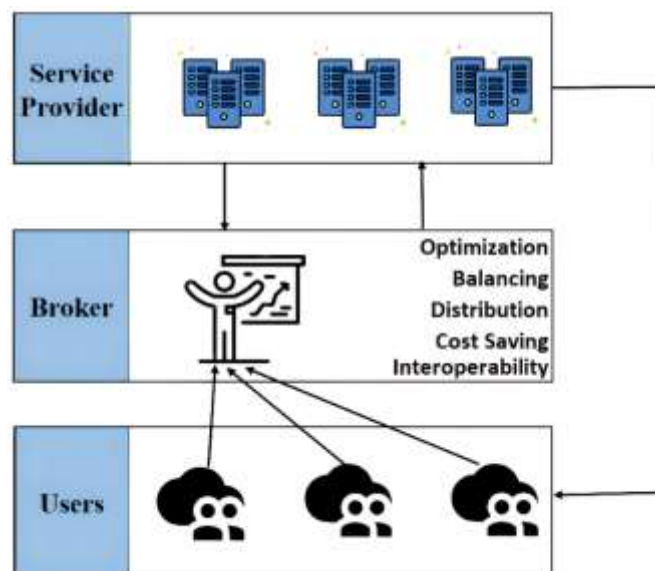


Figure 5: Overview of a broker-based system.



Companies that work with FC concepts include Cisco, IBM, and Microsoft. Some of these companies may also develop or offer Fog brokerage services. The brokerage model inevitably changes as technology develops and the demand for effective resource management in various computing environments increases.

## **2. Research Gaps**

To the best of our knowledge, some of the existing gaps in the field of study are presented in this section:

1. Broker-based load balancing technique using software agents for PPC.
2. ML-based dual broker for selection and allocation of fog services considering QoS parameters.
3. Multilevel hierarchical broker for job scheduling in hexagonal FEs for better global geographical coverage of distributed and ad-hoc fog services.

## **3. Objectives**

Objectives derived from the research gaps are as follows:

1. To design a software agent-based broker framework for load balancing in PPC.
2. To develop an ML-based dual broker framework by integrating QoS parameters for scheduling applications' jobs in the FEs.
3. To develop a multilevel hierarchical broker for job scheduling in hexagonal FEs for better global geographical coverage of distributed and ad-hoc fog services.

## **4. Outline**

Chapter 1 introduces the concept, background, open issues, and objectives of this thesis. Chapter 2 presents the existing literature describing the methodologies and techniques proposed by the authors in load balancing and job scheduling in the fields of CC and FC. This chapter is divided into multiple sections, each discussing the literature details directly related to one of the objectives. Chapter 3 discusses an SA-based broker framework for load balancing in a PPC where SAs are utilized to implement the autonomous behavior in load-balancing brokers. Chapter 4 discusses an ML classification-based dual broker framework by integrating QoS parameters for scheduling applications' jobs in FEs. In this framework, FNs are segregated based on resource availability, and jobs are mapped on the appropriate set of resources. Chapter

5 discusses a multilevel hierarchical broker for job scheduling in hexagonal FEs for better global geographical coverage of distributed and ad-hoc Fog services based upon the QoS Score and average execution time of the FNs.

## 5. Broker-based load balancing technique using software agents for partitioned public cloud

CC empowers resource flexibility with minimal upfront costs, facilitating scalability and cost-effectiveness [19] [20]. According to CISCO, Cloud traffic will reach 14.1 ZB annually soon [21]. The PC is a shared space for users to outsource their jobs [22], where Load-balancing approaches are required to handle under loaded and overloaded resources [23][24][25][26]. Cloud Partitioning is one of the commonly used methods for load balancing in the PC [27] [28] [29]. This chapter discusses a software agent-based load-balancing framework for partitioned public Cloud.

A PC has nodes spread across various geographic locations [14], causing challenges such as complex management, resource allocation, optimization, cost management, data management, and latency management.

### 5.1 Architecture

The brokers in this architecture are practical representations of the SAs:

- Client Agent                      =>    Client
- Partition Agent                   =>    Partition Broker (PB)
- Controller Agent                =>    Controller Broker (CCB)

The major components in the architecture are:

- **Client:** The primary duty of the client is to submit requests, along with the required set of resources, to a Partition Broker (*PB*).
- **PB:** Each partition has a local PB to receive the client requests. PB records the load status of all the nodes inside its partition and shares these details with the Central Controller Broker (*CCB*) whenever required.
- **CCB:** CCB interacts with all the PBs to collect the nodes' load status. CCB is invoked for load balancing across the partitions.

- **Node:** Nodes contain the physical resources, such as storage, computing capacity, graphical resources, memory, and other such resources, to be utilized for processing user requests. Each partition is assumed to have m nodes ( $n_1, n_2, n_3 \dots n_m$ ).

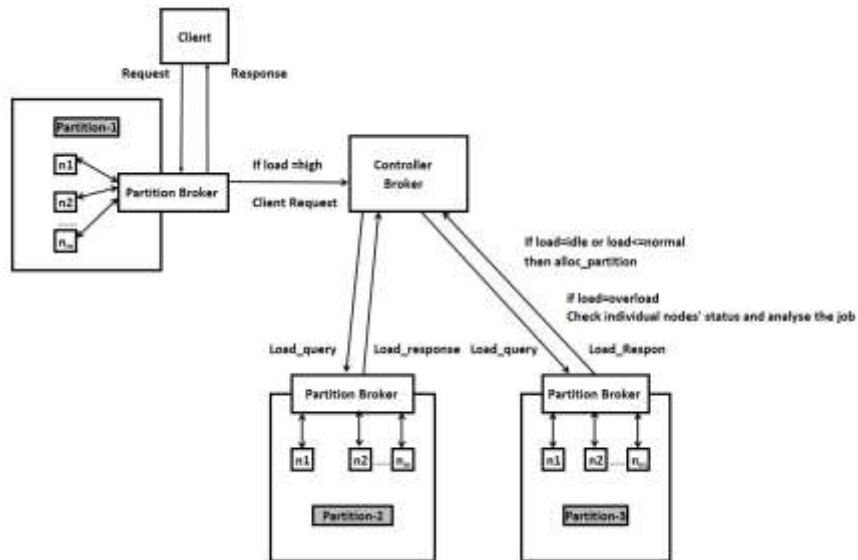


Figure 6: Load Balancing in Partitioned Cloud

## 5.2 Assessing the Load Status of Partitions and Nodes

The number of jobs allocated to a node at a given time is known as its workload. The load status of the Cloud includes the load status of partitions and their nodes and is used to determine the availability of resources for job allocation. The system needs to assess the availability at two levels listed below:

- Choosing the  $P_i$  partition in which a node can be searched.
- Finding a node  $N_j$  to which a job can be allocated.

The proposed framework defines four load states, which are:

- idle,
- normal,
- overloaded and
- full

These states are determined based on the static threshold values pre-defined for the partitions by the CCB and the nodes by the PBs. At any given instance of time, the load state of the

partition depends upon the load state of its nodes. Assuming that there are  $i$  number of partitions,  $j$  number of nodes, given below is the formulation of these load states:

### 5. Load State='idle'

c) Node State:

A  $j^{th}$  node  $N_j$  is considered to be in an idle state if resources available with  $j^{th}$  node  $R_{n,j}$  are beyond a predefined threshold  $t_{n,idle}$ . i.e.

$$N_j = \text{'idle'} \quad \text{IF} \quad R_{n,j} > t_{n,idle} \quad \text{Eq. 1}$$

d) Partition State:

An  $i^{th}$  partition  $P_i$  is considered to be in idle state if the number of idle nodes in that partition are beyond a predefined threshold  $t_{p,idle}$ , i.e.,

$$P_i = \text{'idle'} \quad \text{IF} \quad C_i > t_{p,idle} \quad \text{Eq. 2}$$

where  $C_i$  is no. of idle nodes in a partition.

### 6. Load State='normal'

c) Node State:

A  $j^{th}$  node  $N_j$  is considered to be in a normal state, if resources available with  $j^{th}$  node  $R_{n,j}$  are beyond a predefined threshold  $t_{n,normal}$ . i.e.

$$N_j = \text{'normal'} \quad \text{IF} \quad R_{n,j} > t_{n,normal} \quad \text{Eq. 3}$$

d) Partition State:

An  $i^{th}$  partition  $P_i$  is considered to be in a normal state if the number of normal nodes in that partition are beyond a predefined threshold  $t_{p,normal}$ , i.e.

$$P_i = \text{'normal'} \quad \text{IF} \quad C_i > t_{p,normal} \quad \text{Eq. 4}$$

where  $C_i$  is no. of normal nodes in a partition.

### 7. Load State='overload'

c) Node State:

A  $j^{th}$  node  $N_j$  is considered to be in an overloaded state, if resources available with  $j^{th}$  node  $R_{n,j}$  are beyond a predefined threshold  $t_{n,ovld}$ . i.e.

$$N_j = \text{'overloaded'} \quad \text{IF} \quad R_{n,j} > t_{n,ovld} \quad \text{Eq. 5}$$

d) Partition State:

An  $i^{th}$  partition  $P_i$  is considered to be in an overloaded state if the number of overloaded nodes in that partition are beyond a predefined threshold  $t_{p,ovld}$  i.e.

$$P_i = \text{'overloaded'} \text{ IF } C_i > t_{p,ovld} \quad \text{Eq. 6}$$

where  $C_i$  is the number of ovld nodes in a partition.

### 8. Load State='full'

An  $i^{th}$  partition  $P_i$  is considered to be in full load state if the number of overloaded nodes  $C_i$  in that partition is equal to  $j$ , i.e.,

$$P_i = \text{'full'} \text{ IF } C_i = j \quad \text{Eq. 7}$$

where  $j$  is the total number of nodes in that partition.

## 5.3 Job Assignment

Following this load status review, three outcomes are possible:

- **Case 1:** If a single node  $N_j$  is available with an idle or normal state, then allocate the job to  $N_j$  and update the load status of the partition ' $S_{p,i}$ ' and the resources at node  $R_{n,j}$
- **Case 2:** If more than one node is available with an idle or normal state, then make a list of all these nodes as  $N_{capable[]}$
- **Case 3:** If No node is available with sufficient resources, PB transfers the request to the CCB to search for a node in other partitions.

The CCB calls all other partition's PBs and adds the partition's ID to a list of  $P_{capable[]}$ , capable partitions whenever it discovers a  $P_{normal}$  or  $P_{idle}$  load state. A similar process is followed to search a node  $N_j$  in all the partitions in  $P_{capable[]}$ .

## 5.4 Complexity Analysis

Time complexity varies according to the stages of the process. When the job was initially submitted, just one node's availability status was assessed. Consequently, the complexity of this phase is  $O(1)$ . The complexity class  $O(N)$  is applicable in all other scenarios, and the value of  $N$  varies according to the number of nodes in a partition and their RAV. Table 3.2 compiles the complexity for different possible scenarios.  $P$  denotes a Partition, and  $N$  is a Node in Table 3.2. The number of partitions and nodes within each partition impacts the overall complexity.

**Table 1:** Complexity Analysis of Algorithms and its s

Sr. No.	Step	Time Complexity
1.	Checking the current node for the free resources	$O(1)$
2.	Checking other nodes in the current partition	$O(N_P)$ : $N_P$ is the number of nodes in the partition P
3.	Checking other partitions	
	a) Every partition has the same number of nodes	$O(P*N_P)$
	b) Partitions have different number of nodes	$O(N_{P1} + N_{P2} + \dots)$
4.	Checking across multiple partitions	$O(\text{Count}(P_{capable}))$
5.	Checking across multiple nodes	$O(\text{Count}(P_{capable}(N)))$

## 5.6 Experimental Results and Discussion

This section presents experimental results and their discussion. This technique is implemented in CloudSim.

The AET, MET, and TET are compared with FCFS and SJF algorithms within the same setup to assess the performance of the proposed framework. Each algorithm is run five times as part of the experiment. Table 3.3 details the experimental setup with the heterogeneous sizes of VMs and jobs.

**Table 2:** Experimental setup

Sr. No.	Parameter Name	Values
1.	Partitions	2
2.	Brokers	2
3.	Datacenters	4 (2 Datacenters/partition)
4.	Host	2 hosts for each Datacenter
5.	VMs	6 VMs in each Datacenter
6.	Tasks	40 tasks with each broker

All these parameters are considered to be the basic evaluation parameters because the users of DS, such as Cloud or Fog, are very particular towards the time-based parameters. The execution and response time of the application jobs are significant and helpful in assessing a site's performance. Other parameters, such as accuracy, reliability, and QoSs, are considered to be taken care of by default. However, time-oriented parameters require experimentation to devise and follow load-balancing schemes that offer better results in terms of such parameters. According to the results, the proposed framework performs better than the FCFS algorithm, while the SJF algorithm performs well overall.

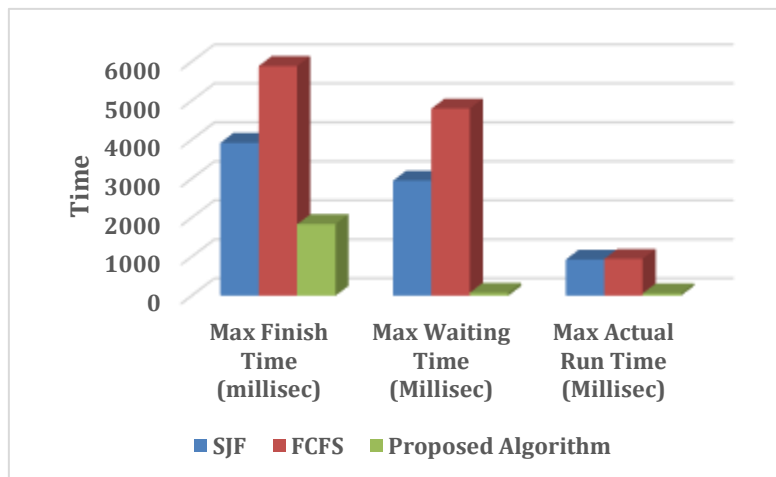
## 5.6 Experimental Setup and Results

The algorithm is implemented in CloudSim with the following setup details. Implementation is done on this small set of attributes for sampling purposes with certain assumptions.

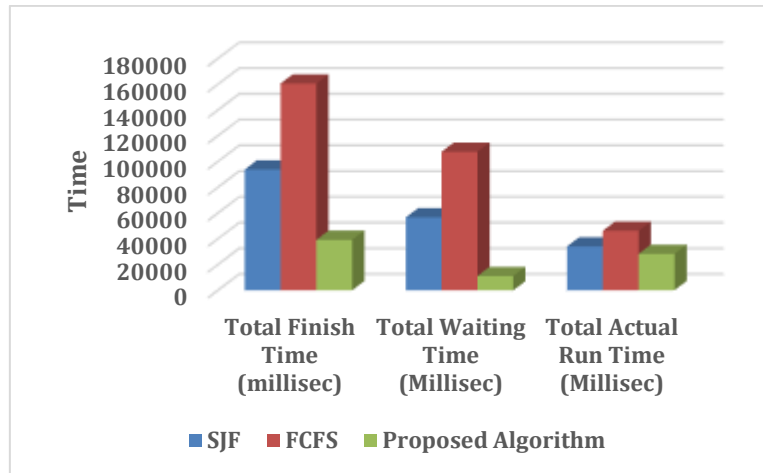
**Table 3:** Implementation Setup Parameters

Sr. No.	Parameter Name	Parameter
1.	No. of Partitions	2
2.	No. of Brokers	2
3.	No. Of Datacenters	4 (2 Datacenters with each partition)
4.	No. of Hosts	2 Hosts for each Datacenter
5.	No. of VMs	6 VMs in each Datacenter
6.	No. of Cloudlets	40 Cloudlets with each broker

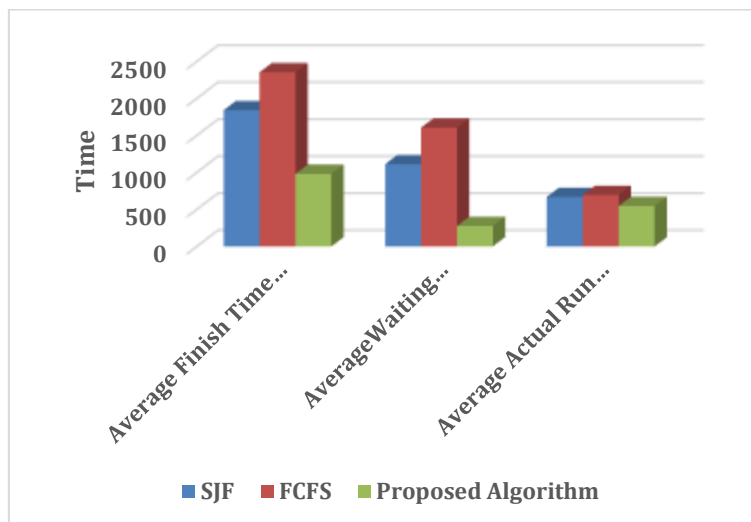
Results are compared for the average finish time of cloudlets, waiting for the time of cloudlets, and actual run time. Figure 7 shows the comparison of MET of a cloudlet during simulation. In Figure 8, a comparison is made between the average total waiting time, average total finish time, and the average maximum actual time of all the cloudlets. Figure 9 compares the actual run time of the cloudlets. Results show that the proposed framework produces better results than FCFS and SJF algorithms.



**Figure 7:** MET Comparison



**Figure 8: TET Comparison**

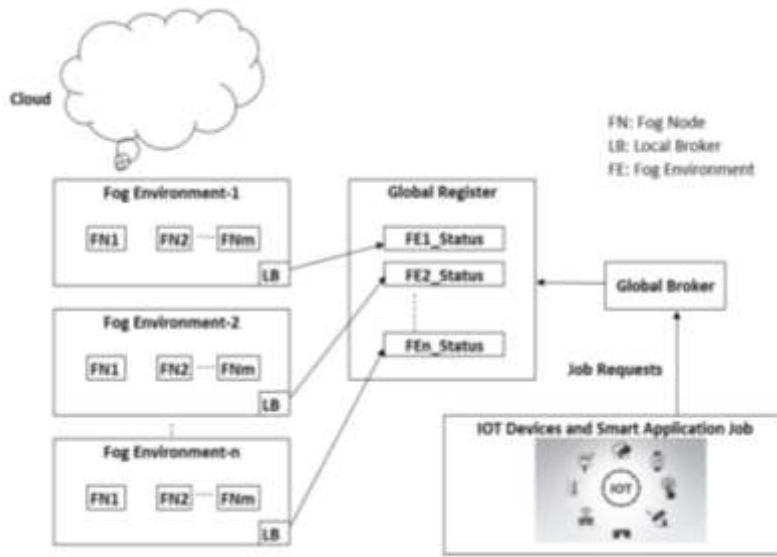


**Figure 9: AET Comparison**

## 9. Machine learning-based dual broker for selection and allocation of fog services considering QoS parameters

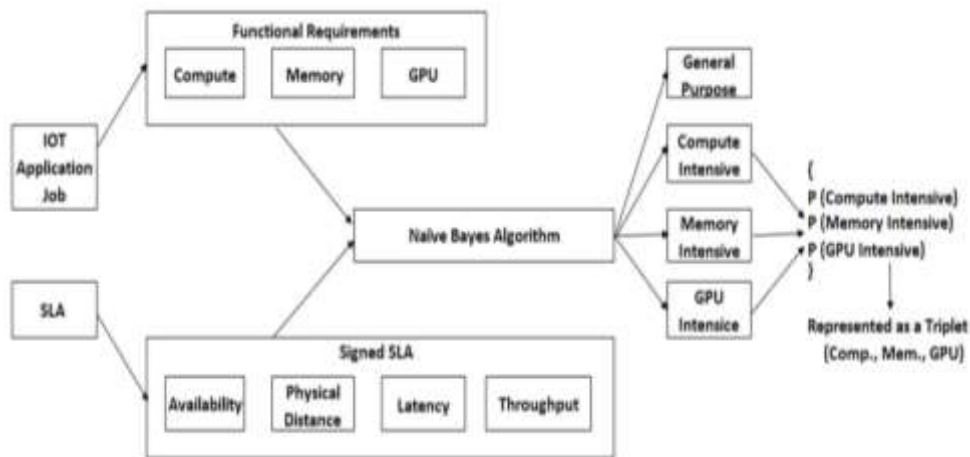
As seen in the previous chapter, efficient resource allocation to the requested jobs significantly improves the performance of DSs. Modern domains are being researched and implemented in job scheduling, including classification techniques in ML. ML classification is a versatile and powerful tool that enables automation, pattern recognition, and predictive modeling across various applications and industries. It helps extract valuable insights from data, improve decision-making processes, and enhance efficiency. The current chapter discusses an ML-based dual broker framework integrating QoS parameters for scheduling applications' jobs.





**Figure 10:** High-Level Architecture of the Dual BBF.

Two ML-based approaches are used in this framework: Naïve Bays Algorithm: At the fine-grained (node selection) level, a probability triad (C, M, G) is anticipated using the Naïve Bayes algorithm, which provides a probability of a newly submitted application’s job to fall in either of the categories Compute (C) intensive, Memory (M) intensive and GPU (G) intensive. Self-Organizing Map: SOM maps the application’s jobs onto the corresponding resource cluster.

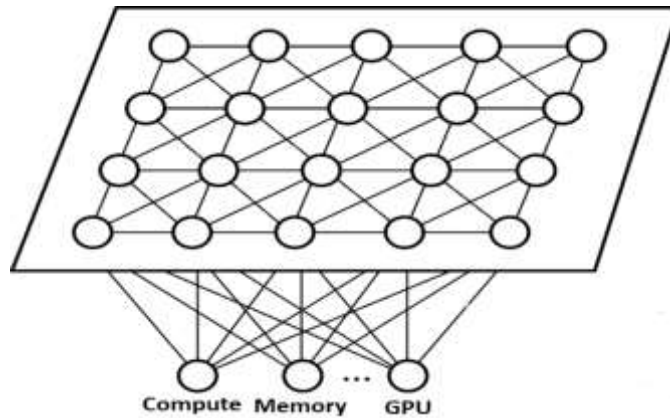


**Figure 11:** QoS Parameters and Functional Requirement-Based Triplet Generation.

Each FN in an environment is virtually clustered, corresponding to these four categories using SOM. Any newly submitted request in the proposed framework is allocated to one of these.

**Table 4:** Node types and application's job types.

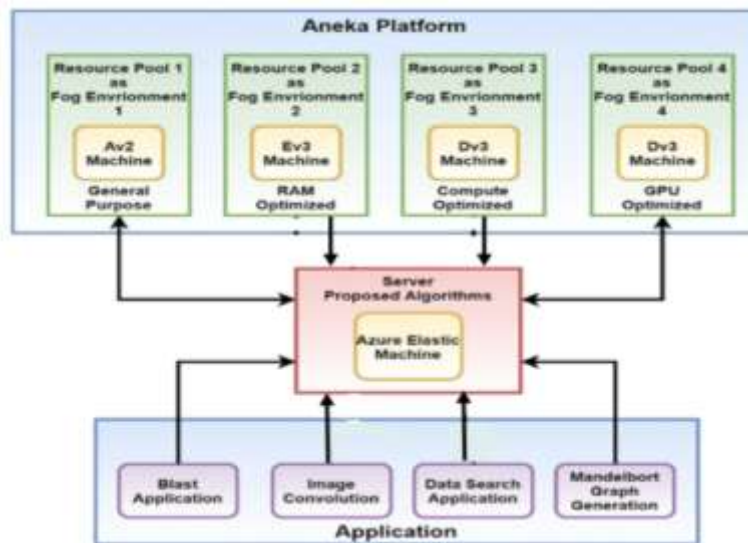
Node Type	GPU Requirement	Memory Requirement	Compute Requirement
General Purpose	Normal	Normal	Normal
GPU Intensive	High	Normal	Normal
Memory Intensive	Normal	High	Normal
Compute Intensive	Normal	Normal	High



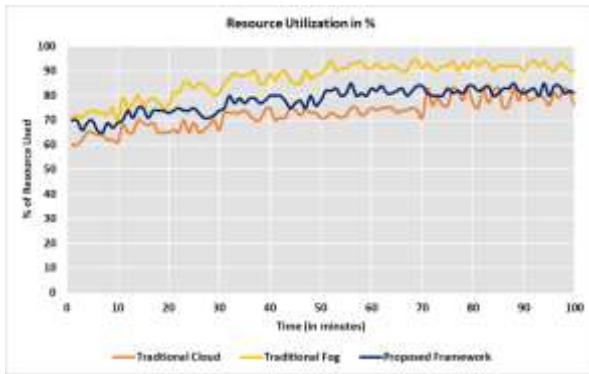
**Figure 12:** Use of SOM for mapping triplets onto the resource clusters.

### 5.1 Experimental Setup and Results

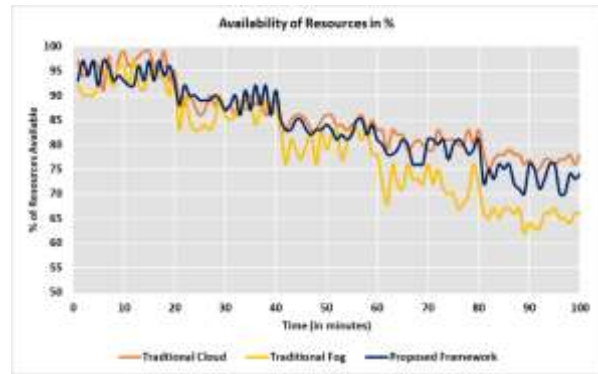
The experiment was run for 100 minutes because saturation was observed after this time. Five metrics were recorded in the experiment for three installed infrastructures: resource utilization, availability, response time, waiting time, and completion time.



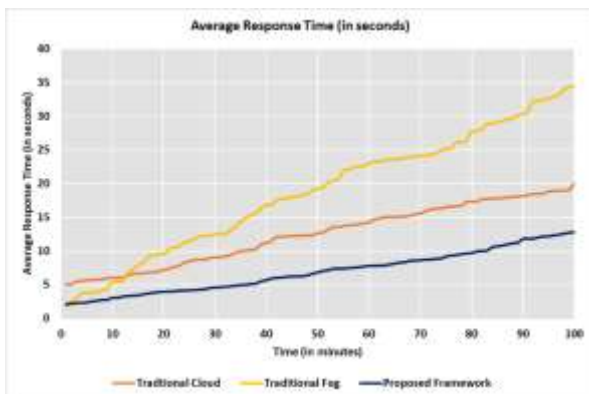
**Figure 13:** Testbed for the performance evaluation



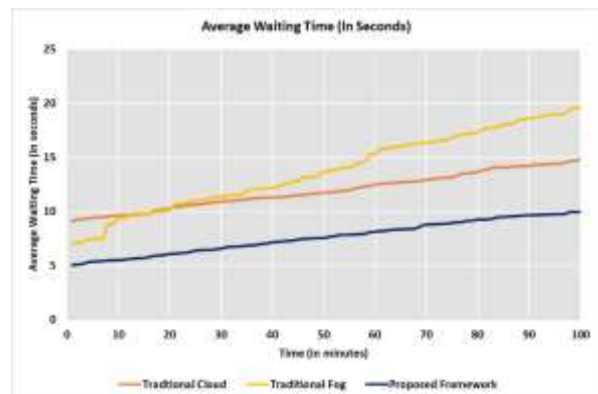
**Figure 14:** ARU of Available Computing Infrastructure



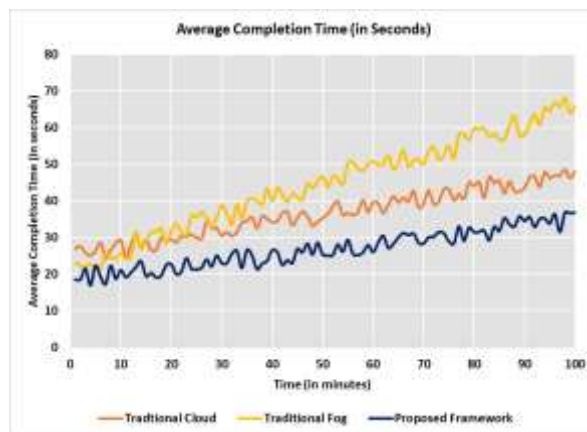
**Figure 15:** Comparison of availability of resources from installed infrastructure



**Figure 16:** Comparison of average response time (ART) in Installed Infrastructure



**Figure 17:** AWT of Installed Infrastructure



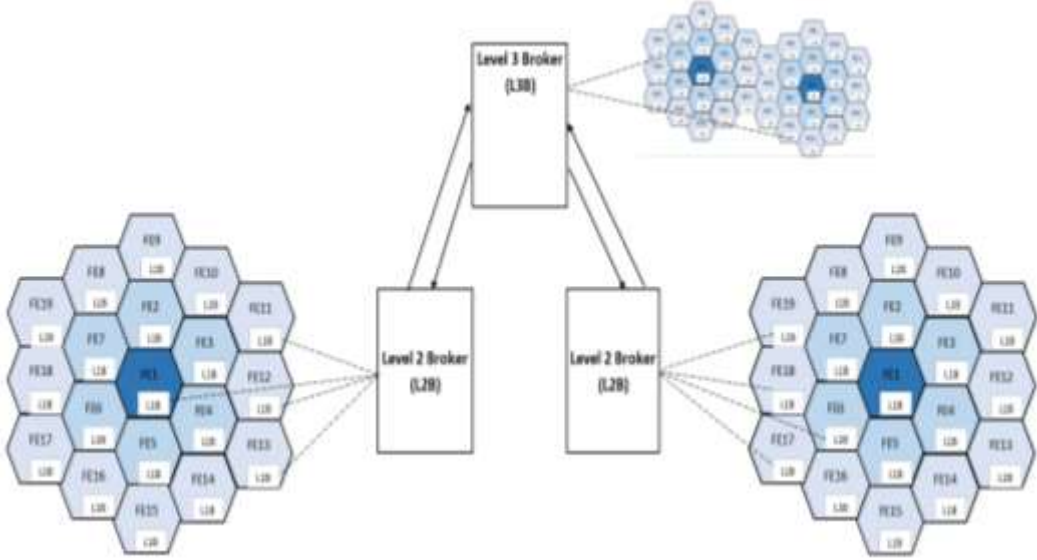
**Figure 18:** ACT of Installed Infrastructure

After careful analysis of available results, it can be concluded that selecting specific resources for each type of application results in better performance. Although the proposed framework performed close to the cloud computing platform for utilization and availability, it is better in all parameters related to the timely execution of jobs submitted to the application. The latency parameter was the same in the case of the traditional FC environment, but it failed

to perform because resources were and many applications had to wait for their execution to start. The adverse result of traditional FC proved the viability and usage of the proposed framework.

**10. Multilevel hierarchical broker for job scheduling in hexagonal fog environments for better global geographical coverage of distributed and ad-hoc fog services.**

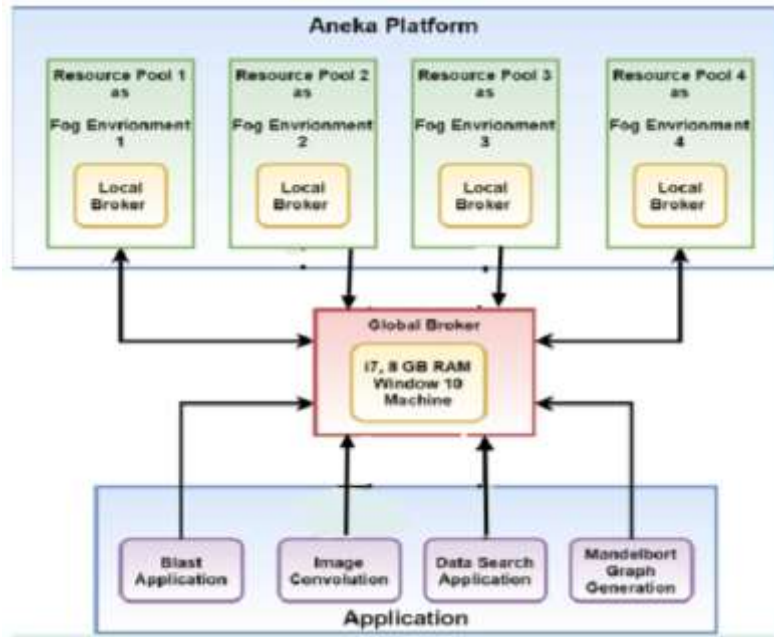
The previous Chapter elaborates on a dual-broker implementation in FC. That solution refers to a single FE. This chapter presents a hierarchical broker-based technique for job scheduling in hexagonal FEs for better global geographical coverage of distributed and ad-hoc Fog services.



**Figure 19:** Global geographical coverage using multilevel brokers

In the proposed framework, every FE has an L1B deployed inside it, which accounts for the available resources of all the FNs inside that FE. Are L2Bs deployed at the upper level of the hierarchy, representing FEs of a GR, and it is responsible for communicating with the L1Bs of all FEs. For global coverage of resources, another level of hierarchy has an L3B, which coordinates all the GRs by communicating with all L2 Brokers at the global level.

## 6.1 Experimental Setup and Results



**Figure 20:** Testbed for experimental setup and performance analysis

Figure 20 shows the overall components of the testbed created to evaluate the proposed framework, which contains four different applications, four FEs with a variable number of nodes, and a machine that handles the scheduling. Four applications used in the testbed are already available in the Aneka platform, and the authors have not created them. These applications generate a series of independent tasks that can be allocated to any FE, as listed in Table 3. All tasks generated from these applications are submitted to the GBR, which decides which task to run on a given FN. In the testbed, the GBR is a Windows 10 machine with Intel(R) Core (TM) i7-10700 CPU 2.90 GHz processor, 16.0 GB (15.7 GB usable) RAM, and a 64-bit operating system. It runs an Aneka master container that manages the scheduling of jobs on the desired FEs. The proposed algorithm, which manages different aspects of scheduling in any distributed computing environment, is developed in Aneka PaaS software.

**Table 5:** Applications used in the testbed

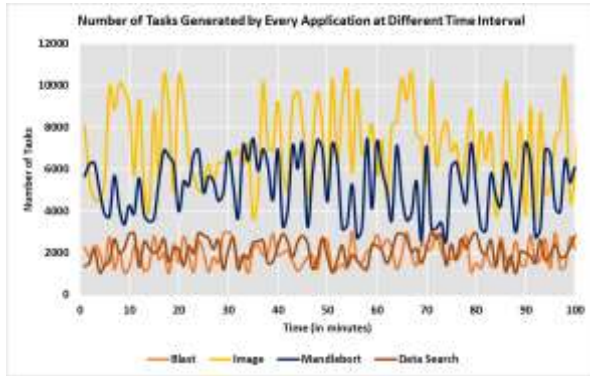
S. No.	Name	Tasks per minute	TtC Range
1	Blast Application	1000	5-15 seconds
2	Image Convolution	3600	1-2 seconds
3	Mandlebort	2500	1-5 seconds
4	Data Search	1000	5-15 seconds

**Table 6:** Node number and configuration of FEs

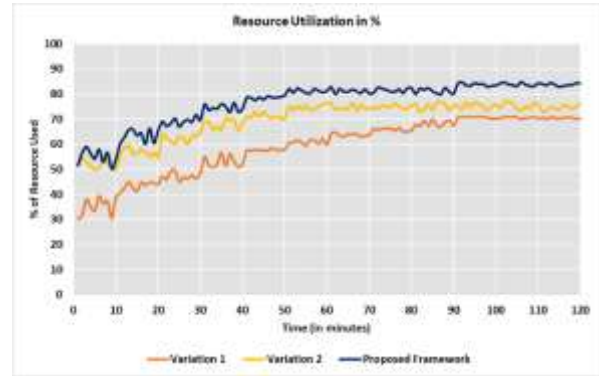
S. No.	FE	Number of Nodes	Node Configuration
1	FE 1	1	1 Core @ 2.9 GHz, 1 GB RAM, 20 GB HDD
2	FE 2	6	1 Core @ 2.9 GHz, 1 GB RAM, 20 GB HDD
3	FE 3	12	1 Core @ 2.9 GHz, 1 GB RAM, 20 GB HDD
4	FE 4	18	1 Core @ 2.9 GHz, 1 GB RAM, 20 GB HDD

**Table 7:** Parameters for Time used in the Proposed Framework's Testbed.

Parameter	Value
$t_{layer}$	1-5 seconds (chosen randomly)
$t_{avail}$	3-8 seconds (chosen randomly)
$t_{score}$	1-5 seconds (chosen randomly)
Latency per hexagon	5-15 seconds (Poisson distributed)



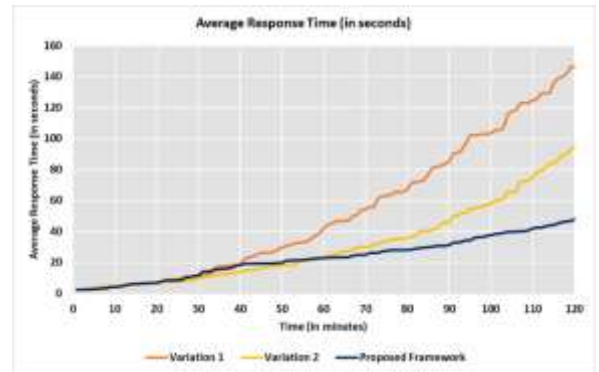
**Figure 21:** Number of tasks generated by every application at different time intervals



**Figure 22:** ARU of FEs



**Figure 23:** Comparison of the average resource availability of FEs.

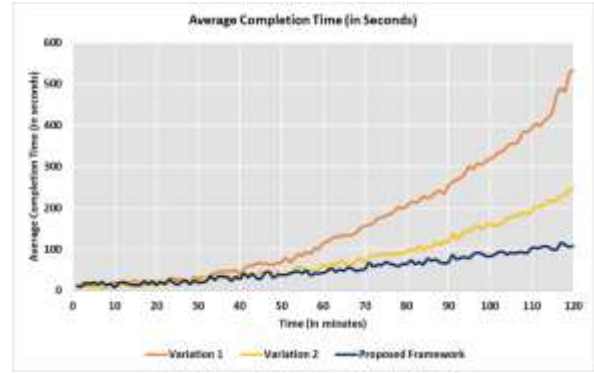


**Figure 24:** Comparison of average response time (ART) in Application Tasks





**Figure 25:** AWT of Application Tasks.



**Figure 26:** ACT of application tasks.

A comparison of experimental results shows that the proposed framework performs much better regarding resource utilization, resource availability, waiting time, response time, and job completion. In the future, this model can be expanded by considering a flexible set of QoS parameters as per the requirements of the application’s job to be executed.

## 11. Conclusion

In the end, the research attempts to propose and verify different approaches to achieve better execution timing, better utilization of deployed resources, and reducing the complexities of load balancing due to the large search space in the Distributed Systems. Improved operational and QoS parameters improve the user experience and the service provider’s goodwill. Efficient resource allocation is the major contributor to the evolution of modern computing paradigms, and this thesis is an attempt to contribute in the same direction.

## 12. Future Scope

This thesis is focused on service orchestration in the Cloud and FC environments and has proposed frameworks based on chosen parameters only. There is a long list of the network (number of hops, BW, packet loss, network throughput, network delay, network jitter) and infrastructure (CPU count, amount of memory, size of disk space, percentage of CPU, percentage of memory, free disk) related QoS parameters which contribute significantly in service orchestration. [30]. Experimentation can be performed upon this set of resources to verify if the proposed frameworks produce significantly improved results in the expanded set of parameters. More ML techniques can be explored along with expanding the list of parameters, as the same technique cannot benefit every parameter. Another field of future research to explore is to scale the hierarchy levels in a multilevel broker platform where

hierarchical levels can be added or removed per the geographical size and resource requirements of fluctuating request volume. Scaling in terms of hexagonal FEs, including their size and number of hexagonal GRs, is also a dimension that can be experimented with.

## References

- [1] R. Vaithyanathan and T. A. Govindharajan, “User preference-based automatic orchestration of web services using a multi-agent,” *Computers & Electrical Engineering*, vol. 45, pp. 68–76, Jul. 2015, doi: 10.1016/j.compeleceng.2015.03.021.
- [2] R. Sandhu, N. Kaur, S. K. Sood, and R. Buyya, “TDRM: tensor-based data representation and mining for healthcare data in cloud computing environments,” *J Supercomput*, vol. 74, no. 2, pp. 592–614, Feb. 2018, doi: 10.1007/s11227-017-2163-y.
- [3] C. Carrión, “Kubernetes Scheduling: Taxonomy, Ongoing Issues and Challenges,” *ACM Comput. Surv.*, vol. 55, no. 7, pp. 1–37, Jul. 2023, doi: 10.1145/3539606.
- [4] N. Greneche, T. Menouer, C. Cérin, and O. Richard, “A Methodology to Scale Containerized HPC Infrastructures in the Cloud,” in *Euro-Par 2022: Parallel Processing*, vol. 13440, J. Cano and P. Trinder, Eds., in Lecture Notes in Computer Science, vol. 13440. , Cham: Springer International Publishing, 2022, pp. 203–217. doi: 10.1007/978-3-031-12597-3\_13.
- [5] N. Greneche and C. Cerin, “Autoscaling of Containerized HPC Clusters in the Cloud,” in *2022 IEEE/ACM International Workshop on Interoperability of Supercomputing and Cloud Technologies (SuperCompCloud)*, Dallas, TX, USA: IEEE, Nov. 2022, pp. 1–7. doi: 10.1109/SuperCompCloud56703.2022.00006.
- [6] B. Sharma, R. Prabhakar, S.-H. Lim, M. T. Kandemir, and C. R. Das, “MROrchestrator: A Fine-Grained Resource Orchestration Framework for MapReduce Clusters,” in *2012 IEEE Fifth International Conference on Cloud Computing*, Honolulu, HI, USA: IEEE, Jun. 2012, pp. 1–8. doi: 10.1109/CLOUD.2012.37.
- [7] M. Firoz Ali, R. Zaman Khan, *Distributed Computing: An Overview*. Int. J. Advanced Networking and Applications, 2015.
- [8] M. Van Steen and A. S. Tanenbaum, “A brief introduction to distributed systems,” *Computing*, vol. 98, no. 10, pp. 967–1009, Oct. 2016, doi: 10.1007/s00607-016-0508-7.
- [9] A. D. Kshemkalyani and M. Singhal, *Distributed Computing: Principles, Algorithms, and Systems*, 1st ed. Cambridge University Press, 2008. doi: 10.1017/CBO9780511805318.
- [10] A. S. Tanenbaum and M. van Steen, *Distributed systems: principles and paradigms*, Second edition, Adjusted for digital publishing. The Netherlands? Maarten van Steen, 2016.
- [11] I. Ahmad, A. Ghafoor, and G. C. Fox, “Hierarchical Scheduling of Dynamic Parallel Computations on Hypercube Multicomputers,” *Journal of Parallel and Distributed Computing*, vol. 20, no. 3, pp. 317–329, Mar. 1994, doi: 10.1006/jpdc.1994.1030.
- [12] A. Grama, Ed., *Introduction to parallel computing*, 2. ed., [Reprint.]. Harlow: Pearson, 2011.



- [13] K. A. Nuaimi, N. Mohamed, M. A. Nuaimi, and J. Al-Jaroodi, "A Survey of Load Balancing in Cloud Computing: Challenges and Algorithms," in *2012 Second Symposium on Network Cloud Computing and Applications*, London, United Kingdom: IEEE, Dec. 2012, pp. 137–142. doi: 10.1109/NCCA.2012.29.
- [14] Junjie Pang Gaochao Xu and Xiaodong Fu, "A load balancing model based on cloud partitioning for the public cloud," *TSINGHUA SCIENCE AND TECHNOLOGY*, pp. 34–39, 2013.
- [15] D. Mohandass, S. R., and R. R., "A novel approach for dynamic cloud partitioning and load balancing in cloud computing environment," *Journal of Theoretical and Applied Information Technology*, pp. 662–667, 2014.
- [16] R. Sandhu and S. K. Sood, "Scheduling of big data applications on distributed cloud based on QoS parameters," *Cluster Comput*, vol. 18, no. 2, pp. 817–828, Jun. 2015, doi: 10.1007/s10586-014-0416-6.
- [17] R. Deng, R. Lu, C. Lai, T. H. Luan, and H. Liang, "Optimal Workload Allocation in Fog-Cloud Computing Towards Balanced Delay and Power Consumption," *IEEE Internet Things J.*, pp. 1–1, 2016, doi: 10.1109/JIOT.2016.2565516.
- [18] S. Tuli, R. Sandhu, and R. Buyya, "Shared data-aware dynamic resource provisioning and task scheduling for data intensive applications on hybrid clouds using Aneka," *Future Generation Computer Systems*, vol. 106, pp. 595–606, May 2020, doi: 10.1016/j.future.2020.01.038.
- [19] A. Nahir, A. Orda, and D. Raz, "Replication-Based Load Balancing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 27, no. 2, pp. 494–507, Feb. 2016, doi: 10.1109/TPDS.2015.2400456.
- [20] J.-P. Yang, "Elastic Load Balancing Using Self-Adaptive Replication Management," *IEEE Access*, vol. 5, pp. 7495–7504, 2017, doi: 10.1109/ACCESS.2016.2631490.
- [21] P. Zhao, W. Yu, S. Yang, X. Yanga, and J. Lin, "On Minimizing Energy Cost in Internet-scale Systems with Dynamic Data," *IEEE Access*, pp. 1–1, 2017, doi: 10.1109/ACCESS.2017.2725939.
- [22] Z. Qiu and J. F. Perez, "Evaluating Replication for Parallel Jobs: An Efficient Approach," *IEEE Trans. Parallel Distrib. Syst.*, vol. 27, no. 8, pp. 2288–2302, Aug. 2016, doi: 10.1109/TPDS.2015.2496593.
- [23] J.-P. Yang, "Intelligent Offload Detection for Achieving Approximately Optimal Load Balancing," *IEEE Access*, vol. 6, pp. 58609–58618, 2018, doi: 10.1109/ACCESS.2018.2873287.
- [24] S. Souravlas and A. Sifaleras, "Trends in data replication strategies: a survey," *International Journal of Parallel, Emergent and Distributed Systems*, vol. 34, no. 2, pp. 222–239, Mar. 2019, doi: 10.1080/17445760.2017.1401073.
- [25] N. Mostafa, I. Al Ridhawi, and A. Hamza, "An intelligent dynamic replica selection model within grid systems," in *2015 IEEE 8th GCC Conference & Exhibition*, Muscat, Oman: IEEE, Feb. 2015, pp. 1–6. doi: 10.1109/IEEEGCC.2015.7060061.
- [26] R. Yadav and A. S. Sidhu, "Fault tolerant algorithm for Replication Management in distributed cloud system," in *2015 IEEE 3rd International Conference on MOOCs*,

- Innovation and Technology in Education (MITE)*, Amritsar, India: IEEE, Oct. 2015, pp. 78–83. doi: 10.1109/MITE.2015.7375292.
- [27] P. Sharma and P. Singh, “Load degree calculation for the public cloud based on cloud partitioning model using turnaround time,” *International Journal of Computer Science and Information Technologies*, 2015.
- [28] M. Pantazoglou, G. Tzortzakis, and A. Delis, “Decentralized and Energy-Efficient Workload Management in Enterprise Clouds,” *IEEE Trans. Cloud Comput.*, vol. 4, no. 2, pp. 196–209, Apr. 2016, doi: 10.1109/TCC.2015.2464817.
- [29] D. Lan *et al.*, “Task Partitioning and Orchestration on Heterogeneous Edge Platforms: The Case of Vision Applications,” *IEEE Internet Things J.*, vol. 9, no. 10, pp. 7418–7432, May 2022, doi: 10.1109/JIOT.2022.3153970.
- [30] S. Taherizadeh, V. Stankovski, and M. Grobelsnik, “A Capillary Computing Architecture for Dynamic Internet of Things: Orchestration of Microservices from Edge Devices to Fog and Cloud Providers,” *Sensors*, vol. 18, no. 9, p. 2938, Sep. 2018, doi: 10.3390/s18092938.