

ENERGY AWARE LOAD BALANCING IN FOG COMPUTING

A Thesis

Submitted in partial fulfillment of the requirements

for the award of the degree of

DOCTOR OF PHILOSOPHY

in

COMPUTER SCIENCE AND ENGINEERING

By

Mandeep Kaur

41700092

Supervised By

Dr. Rajni



LOVELY PROFESSIONAL UNIVERSITY

PUNJAB

2021


DECLARATION

I hereby declare that thesis entitled "Energy aware load balancing in fog computing" submitted by me for Degree of Doctor of Philosophy in Computer Science and Engineering is the result of my original and independent research work carried out under the guidance of my Supervisor Dr. Rajni, Associate Professor, School of Computer Science and Engineering, Lovely Professional University, Jalandhar. This work has not been submitted for the award of any degree or fellowship of any other University or Institution.

Mandeep Kaur
School of Computer Science and Engineering,
Lovely Professional University,
Phagwara, Punjab-144411, India
Date:

CERTIFICATE

This is to certify that the thesis entitled "Energy aware Load balancing in Fog computing" submitted by Mandeep Kaur for the award of the degree of Doctor of Philosophy in Computer Science and Engineering, Lovely Professional University, is entirely based on the work carried out by her under my supervision and guidance. The work reported, embodies the original work of the candidate and has not been submitted to any other university or institution for the award of any degree or fellowship, according to the best of my knowledge.



Dr. Rajni

Associate Professor

School of Computer Science and Engineering

Lovely Professional University

Phagwara, Punjab-144411 India

Date:

ABSTRACT

Fog computing was introduced to cope up with the problems faced by cloud computing, i.e., high latency, the distance between data centers and users, locality of data centers. Fog computing brings the services provided by the cloud to the edge of the network. In a fog computing environment, user requests are processed on the fog nodes deployed near to end-user layer. The users are aware of the location of fog nodes, and due to the less distance from end-users and fog nodes, the latency is reduced. Fog computing is a distributed approach that provides an intermediate layer between the end-users and the cloud layer. Hence, it works as a helping hand to the cloud to provide computing, networking, and storage services near the end-user layer. E-healthcare, intelligent traffic management, smart homes, intelligent waste management systems, smart cities, scientific workflows are widespread applications of fog computing in this era of digitization.

Fog computing can play an essential role in executing parallel computational tasks, i.e., scientific workflow applications. To run workflow applications on the cloud datacenters may take more time. So, fog computing can help the cloud to reduce latency issues faced by these applications during execution. While executing large computational tasks, fog computing faces challenges, i.e., load balancing, high energy consumption, less storage, resource management. So, there is a need to develop a resource-utilization-based framework that can help to enhance the performance of fog systems and ensure maximum resource utilization that can be achieved by applying load balancing in the fog layer.

Load balancing and energy consumption are essential issues to be resolved in the fog computing environment. Due to the excess load on the fog servers, the situation can occur in which few resources can become overloaded, and few will remain underloaded. In this scenario, few resources will consume much energy, and others that remain idle will also consume 60% of energy compared to the used ones. Hence, proper load balancing is required in the fog environment so that all the resources can get an equal proportion of load and can be fully utilized. This will also help to reduce energy consumption in a fog environment. So, there is a need for a resource-utilization-based load balancing framework that can help to utilize all the available resources fully.

This research work proposes a resource-utilization-based load balancing framework and an energy-aware load balancing approach to achieve the set objectives. Extensive literature

has been reviewed on load balancing and energy consumption in fog computing. A thorough study of existing load balancing approaches has been conducted. The existing load balancing approaches have been studied and analyzed also. Based on the literature review, it has been observed that load balancing and energy consumption in fog nodes have been emerging issues in fog computing that need to be addressed.

This research work proposes an approach for load balancing in the fog computing environment. Firstly, a resource utilization-based load balancing framework for the fog computing environment has been proposed that helps to enhance resource utilization that can reduce energy consumption in the fog environment. A load-balancing algorithm has been proposed to evaluate the proposed framework. The proposed technique is Tabu-GWO-ACO that is a hybrid approach made up of three different approaches, i.e., Tabu search, Grey Wolf Optimization(GWO), Ant Colony Optimization(ACO). These different approaches are hybridized to get the optimized solution for load balancing in the fog layer. The proposed algorithm has been applied to evaluate the proposed framework.

Furthermore, a resource-utilization-based workflow execution model for fog computing has been proposed. The proposed model tries to execute workflow tasks by balancing the load among all available resources. Along with this, a PSW-fog Clustering algorithm has been proposed to reduce the execution time and energy consumption by executing scientific workflow applications in a fog-cloud environment. The proposed algorithm is named as PSW-Fog Clustering-based Load balancing algorithm that is a hybridized form of Plant Growth Optimization(PGO), Water Cycle Optimization(WCO), and Simulated Annealing Approach(SAA).

This work considers scientific workflow applications to evaluate the proposed resource utilization model and energy-aware load balancing approach. Scientific workflows are the representation of tasks in the form of Directed Acyclic Graphs(DAG). Different sensors generate these tasks, actuators in different applications like astronomy, e-healthcare, intelligent traffic management, and many other application scenarios. In this research, four scientific workflows have been considered, i.e., Genome, Sipt, LIGO, Cybershake, to evaluate proposed approaches.

iFogSim toolkit has been used to evaluate the proposed algorithm. Firstly the performance of the Tabu-GWO-ACO has been analyzed and compared with other existing approaches. Afterward, the PSW-fog clustering-based load balancing approach has been eval-

uated, and its results are compared with other existing approaches and Tabu-GWO-ACO. It has been obtained that the PSW-fog clustering-based load balancing approach outperforms other approaches. This research considers three different parameters, i.e., execution time, energy consumption, and cost, to check the performance of proposed approaches. It has been obtained that the proposed approaches implement proper load balancing in executing scientific workflow applications, and this increases the resource utilization that leads to reduce energy consumption in the fog environment. It has been obtained that the proposed approach reduces the execution time, computational cost, and energy consumption in fog nodes while evaluating large computational tasks. The thesis also proposed a prototype model for implementing FOCALB in real time environment i.e. Smart waste management application.

ACKNOWLEDGEMENT

It is a pleasure for me to thank all those who have helped me accomplish this PhD thesis. Firstly, I wish to express my deepest gratitude to Dr Rajni for guiding me throughout this research work. My supervisor has been a continuous source of knowledge, inspiration, motivation and encouragement during the entire course of this research work.

A special thanks to the management of Lovely Professional University to support me in the best possible manner and facilitate me in balancing my work and my research. The doctoral programme of LPU has made it possible for me to pursue my dream of study and upgrade my knowledge.

Special thanks to Examiners of end-term reports and reviewers of journals who vetted my submissions and gave valuable comments to improve the work further.

I take this opportunity to express my gratitude to all my teachers who have shaped me and have contributed immensely to my knowledge and skill development since childhood. I would like to thank all my colleagues for supporting and guiding me.

I would like to take this opportunity to remember my beloved father Late. S. Gurcharan Singh for his motivation and bringing self confidence in me, and my mother Smt. Harmesh Kaur for her endless love and support that helped me to reach upto this level. This thesis would never have been conceived or borne fruit without the unconditional support of my beloved father-in-law S. Nirbhai Singh and mother-in-law Smt. Satvinder Kaur who supported me from beginning to end. Their love, support, and unshakable faith in me provide strength to succeed in all life goals. I am also thankful to my husband, S. Barinder Singh, who has offered full support to me during the entire period of my research work. I would like to thank my daughter Jaivin Kaur for her endless love and support and even for sacrificing her childhood memories with me for me. She supported me on every step. I can not forgot to thank each and every member of my both families who kept on motivating me in downfalls of this research period.

In the end, I would like to thank all my friends for their endless support, and being there when I needed them. Finally, I would like to thank every person who has, directly and indirectly, helped and motivated me in this arduous task.

Mandeep Kaur

TABLE OF CONTENTS

List of Figures	xi
List of Tables	xii
1 INTRODUCTION	1
1.1 Fog computing: Overview	2
1.1.1 Definition	3
1.1.2 Fog architecture	5
1.1.3 Characteristics of fog computing	7
1.1.4 Key areas focused by fog computing	8
1.1.5 Advantages of load balancing in fog computing	10
1.1.6 Open issues and challenges	11
1.2 Scientific workflow applications in fog computing	12
1.3 Load balancing in fog environment	13
1.3.1 Need for load balancing in fog computing	15
1.4 The motivation of study	17
1.5 Thesis contribution	17
1.5.1 Thesis organization	19
2 Literature Survey	22
2.1 Growth of smart devices	23
2.1.1 Motivation	24
2.2 Load balancing techniques	25
2.2.1 Literature review	51
2.2.2 Cost-based load balancing	51
2.2.3 Resource-utilization based load balancing approaches	53
2.2.4 Energy-aware load balancing approaches	55

2.2.5	Year wise review of load balancing techniques	62
2.2.6	Performance measurements that impact load balancing	66
2.3	Open issues and research challenges	69
2.4	Problem formulation	70
2.5	Research assumptions	71
2.6	Research objectives	71
3	Proposed framework for load balancing (FOCALB)	72
3.1	FOCALB: Fog computing architecture of load balancing for scientific workflow application	73
3.1.1	Operating modules of FOCALB	75
3.1.2	Workflow task assignment	77
3.1.3	Workflow models for load balancing in fog computing	78
3.2	Hybridized load balancing algorithm for scientific workflows (Tabu-GWO-ACO)	80
3.2.1	Optimization methods used	81
3.2.2	Proposed hybridized algorithm Tabu-GWO-ACO	84
3.2.3	Flow of execution of Tabu-GWO-ACO	86
3.3	Verification and validation of proposed framework- FOCALB	87
3.3.1	Experimental setup	89
3.3.2	Results and discussion	90
3.3.3	Applications of proposed architecture and approach in real time environment	96
3.4	Conclusion	99
4	An energy-efficient load balancing approach for scientific workflows in fog computing	101
4.1	Scientific workflows	102
4.2	Resource-utilization based Workflow execution model for fog computing . . .	104
4.3	PSW-Fog clustering-based load balancing algorithm	106
4.3.1	Optimization approaches used in our proposed hybrid algorithm . . .	107
4.3.2	PSW-Fog clustering algorithm	110
4.3.3	Flow of execution of proposed algorithm	111
4.3.4	Performance metrics	113

4.4	Analysis of PSW-Fog clustering-based load balancing approach	116
4.4.1	Experimental Requirement	117
4.4.2	Result analysis	118
4.4.3	Applications of proposed energy efficient load balancing algorithm . .	124
4.5	Conclusion	126
5	CONCLUSION AND FUTURE WORK	127
5.1	Conclusion	128
5.2	Future enhancement	130
5.2.1	How limitations can be overcome in future	130
	References	132

List of Figures

1.1	Fog Taxonomy	4
1.2	Layered architecture of fog computing	6
1.3	Key areas of fog computing	9
1.4	Examples of scientific workflows	13
1.5	Load Balancing in fog environment	14
1.6	Flow diagram of load balancing at fog layer	16
2.1	Growth of connected devices from year 2020-2030	23
2.2	Load balancing techniques	26
2.3	Taxonomy of existing load balancing techniques	50
2.4	Evolution of load balancing	65
2.5	Percentage of load balancing metrics considered in reviewed papers	67
3.1	FOCALB	74
3.2	Working of FOCALB	76
3.3	Workflow task assignment	77
3.4	Tabu-GWO-ACO methodology	88
3.5	Cost Analysis of Tabu-GWO-ACO approach	92
3.6	Execution time Analysis of Tabu-GWO-ACO approach	94
3.7	Energy Analysis Tabu-GWO-ACO approach	95
3.8	Applications of FOCALB	97
3.9	Prototype of Waste management system	99
4.1	Example of workflow	103
4.2	Resource-utilization based workflow execution model for fog computing	105
4.3	Flow of execution of PSW-Fog clustering based load balancing algorithm	114

4.4	Cost Analysis of different workflows(PSW-Fog clustering Aproach)	119
4.5	Time delay analysis of LIGO and Cybershake workflows (PSW-Fog clustering)	121
4.6	Time delay analysis of Genome and SIPHT workflows(PSW-Fog clustering Aproach)	122
4.7	Energy Analysis PSW-Fog clustering Aproach	124

List of Tables

2.1	Traditional techniques	29
2.2	Nature-inspired techniques	32
2.3	Agent-based techniques	34
2.4	Real-time based techniques	36
2.5	Hybrid load balancing techniques	39
2.6	Detailed approaches used in existing load balancing algorithms	41
2.7	Comparison of load balancing techniques considered in related work	58
2.8	Comparison of different algorithms on the basis of considered parameters	68
3.1	Required parameters	89
4.1	Notations	107
4.2	Experimental Requirement	117

LIST OF ABBREVIATIONS

IoT	Internet of Things
IoE	Internet of Everything
RMS	Resource Management System
SDLB	Scalable and Dynamic Load Balancer
GWO	Grey Wolf Optimization
PSO	Particle Swarn Optimization
ACO	Ant Colony Optimization
PGO	Plant Growth Optimization
WCO	Water Cycle Optimization
SAA	Simulated Annealing Approach
DAG	Directed Acyclic Graphs
LIGO	Laser Interferometer Gravitational Wave Observatory

CHAPTER 1

INTRODUCTION

With the increase in the Internet of Things (IoT), the amount of data generated by these devices also increases. Cloud computing datacenters provide services for storage and processing services to these IoT devices. Cloud computing provides services on a "pay-as-you-go" basis. Cloud computing has a centralized structure, and its data centers are located far away from the end-users. The time taken to store and process data on data centers is much more than expected.

There are approximately twenty to thirty million edge devices present around the world [1]. These devices generate terabytes to petabytes of data daily. Due to the cloud's centralized storage space, sometimes it becomes difficult for end devices to access the data in emergencies. IoT connects real-world things like smartphones, smart cities, intelligent vehicles, and many other devices to the internet and allows connected devices to exchange data with minimal human interference. In IoT architecture, sensors generate data associated with some applications and send them to the nearest sensor links [2]. The global deployment of different sensors in smart cities requires a computing paradigm to support IoT services, applications, and data analysis.

Cisco launched a new architecture in 2012, fulfilling these IoT requirements, i.e., Fog computing. Fog computing can be said to as an extension of cloud computing to the edge of the network. Fog performs latency-sensitive and energy-aware tasks efficiently on powerful computing nodes at the network's intermediate. Fog computing concept was introduced to meet the demands from different segments of IoT, Internet of Everything (IoE) or Internet of Me (IoM) from start to end, e.g., consumer, wearable, industrial, enterprise, automobile, healthcare, building, energy [3].

This chapter provides a high-level view of this research work by providing an overview of fog computing, its architecture, characteristics, key focus areas, applications, advantages,

and open issues and challenges faced by fog. Further, scientific workflow applications have been described. Furthermore, the need for load balancing in a fog environment has been described. In addition to this, the chapter also motivates to propose resource-utilization-based load balancing in fog computing. In the end, the organization of the rest chapters has been provided along with the thesis contributions.

1.1 Fog computing: Overview

Cisco has introduced fog computing as an extension of the cloud by providing its services near to end-users. Fog computing is defined as an environment where many ubiquitous devices communicate without third-party intervention [4] [5]. Satyanarayanan et al. [6] introduced the term "Cloudlets" for fog computing, which Cisco popularized as an intermediate complementary resource-rich layer between the edge device and the cloud. Fog computing is a highly virtualized platform that provides compute, storage, and networking services between end devices and traditional cloud computing data centers, typically but not exclusively located at the edge of the network. The primary aim of fog computing is to solve the problems faced by cloud computing during IoT data processing. The fog layer acts as an intermediate layer between the cloud and IoT devices [1] [7]. Fog brings the compute, communication, and decision making closer to where data originates from, speeding process times and lower costs.

Fog computing comes up to conquer the problems faced by IoT users. It is a powerful technology that provides different way-outs to the issues experienced in cloud computing. Fog renders storage, computing along with networking services, but in a decentralized manner [8], unlike centralized cloud. In fog computing environments, nodes are deployed in various places and collected processed data from sensors installed at nearby locations. The real-time data received at fog nodes are immediately processed and stored at the fog layer itself. End users are responded after processing data at the fog layer. The data which is rarely to be used is sent to the cloud for further storage and processing.

In this way, fog is not a substitution for cloud computing, but it can work well along with the cloud in a combined form [9].

1.1.1 Definition

Various researchers have proposed several definitions, but Cisco introduced the term in the year 2012. According to Cisco, fog brings cloud services closer to edge devices. Fog computing is an architectural deployment of computing resources where unique nodes are enabled for communication and data transmission between IoT devices rather than backing up their data on the data centers of the cloud [10]. It provides a pool of infrastructure resources of the cloud system, which forms a networking system between the resource and end-user device. [2]. The service layer of fog is characterized as a collaboration of local networks that consists of numerous network nodes like gateways and routers, having finite computing ability [11]. F. Bonomi et al. [1] described fog computing as a distributive and hierarchical computing platform for providing compute, storage, and network service delivery to the end-users.

According to [12], fog computing is near to IoT devices; it is more suitable for lightweight processes. The execution and storage of data between the smart IoT data sensors and the cloud data centers are extended using a fog layer in between. Fog computing broadens the cloud services to compete with challenges of conventional cloud computing [13]. Fog nodes, sitting at the network edge, focus on collecting data, command users, and control IoT devices [10]. Fog computing provides a hierarchical and distributed architecture supporting combination of technical components and services like smart healthcare, cities, homes, and connected e-vehicles. [14].

The fog node is the fundamental entity in the fog computing environment, which helps execute IoT applications. Fog layer provides few characteristics like mobility, geographical distribution, and location awareness [15].

Fog computing is a decentralized computing technology in which data is processed and stored between the source of origin and cloud infrastructure. The fog computing paradigm is primarily motivated by a continuous increase in IoT devices. An ever-increasing amount of data concerning volume, variety, and velocity is generated from an ever-growing array of devices [5].

Figure 1.1 shows the taxonomy of fog computing that describes the technology used in fog, its applications, architectures and frameworks, its features, and its related technologies. The technology used in fog is further divided into three different categories. First is a wireless

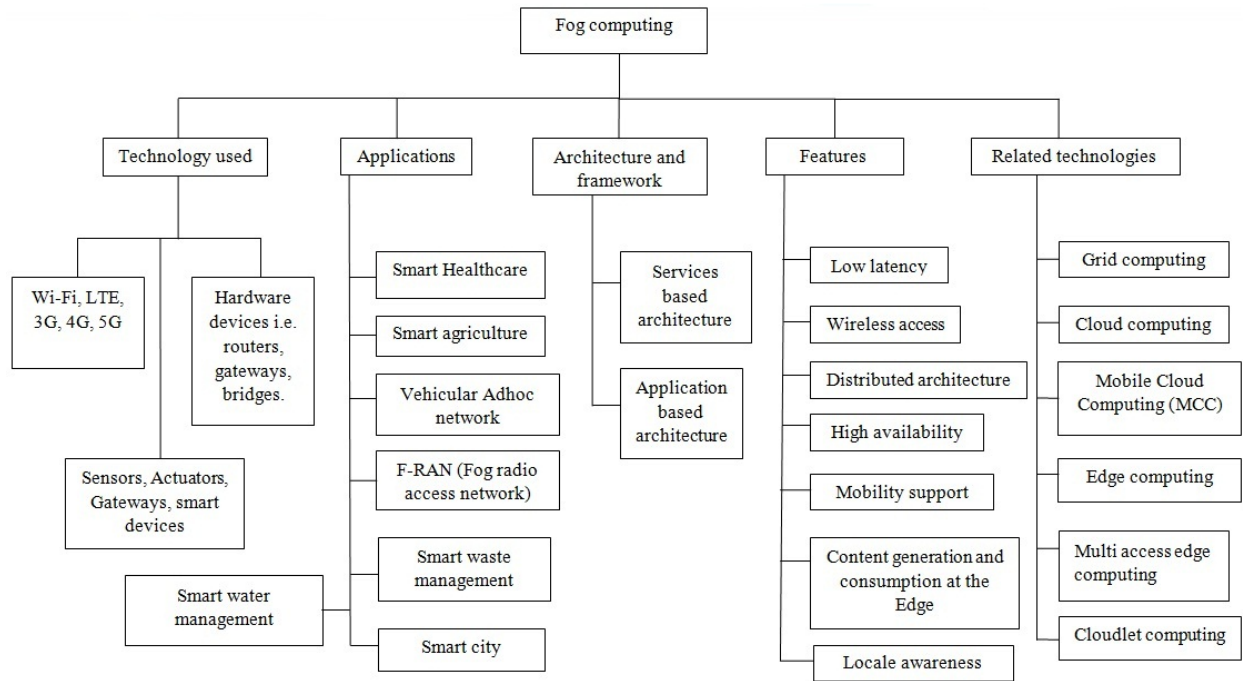


Figure 1.1: Fog Taxonomy

technology connecting IoT with the fog layer, i.e., Wi-Fi, LTE, 3G, 4G, and 5G technologies. The second category is hardware devices, i.e., routers, gateways, and bridges that act as processing and storage devices in the fog layer. The last category is intelligent devices that can become fog nodes, i.e., sensors, actuators, and intelligent gateways. Applications of fog are innovative healthcare, Vehicular Adhoc network, Smart agriculture, Fog Radio Access Network (F-RAN), Smart waste management, smart city, and scientific workflow applications.

The architecture and framework of fog computing are divided into two categories, i.e., service-based and application-based architecture. Fog computing has various features, i.e., low latency, wireless access, distributed architecture, high availability, mobility support, local awareness, and content generation and consumption at the user end. Fog computing has various related technologies, i.e., grid computing, cloud computing, mobile cloud computing, edge computing, multi-access edge computing, cloudlet computing.

1.1.2 Fog architecture

The traditional architecture of fog computing contains three layers. The first layer is the IoT layer containing different smart devices. The second layer is the fog computing layer which contains fog nodes having low computing and storage capacities. The Cloud layer is the top layer containing massive data centers. The data produced by IoT devices is processed by the fog layer, and then after processing the data, users are responded immediately. The fog layer is deployed near the IoT users. Hence in the case of real-time applications, users can get an immediate response. Cloud data centers store the data for a long time that is received from the fog layer. This section represents fog architecture according to different layers' working. The layered architecture contains these layers, i.e., IoT layer, service layer, connectivity layer, fog layer, fog service layer, virtualization, and cloud layer. Working of all these layers is explained as follows:

- **IoT layer:** This layer contains different IoT-based smart devices. These devices are daily usage devices that help users in their daily routine, i.e., smartphones, smart vehicles, smart homes. These devices produce data in bulk within a fraction of seconds per their usage by users. The smart devices in IoT layers contain smart sensors, and these sensors generate data sent to the upper layers for its processing. IoT layer has a large number of service requirements that need to be processed by computing nodes according to their time requirement and availability of resources [16].
- **Service layer:** The service layer differentiates the task generated by the IoT layer into two forms, i.e., firstly, those tasks that need immediate processing. Secondly, those tasks that do not require rapid processing or less urgent tasks. The critical tasks require rapid processing, so they are sent to fog computing nodes in the fog layer. The less critical tasks can be assigned to other intermediate computing nodes.
- **Connectivity layer:** There are various components required to accept data from IoT devices in fog computing architecture, i.e., gateways, wireless and wired connectivity endpoints (i.e., switches, fully rugged routers). There can be client site equipment and gateways installed to access fog computing nodes on the other aspect. In fog architecture, edge devices can be connected with core networks and access cloud servers and services.

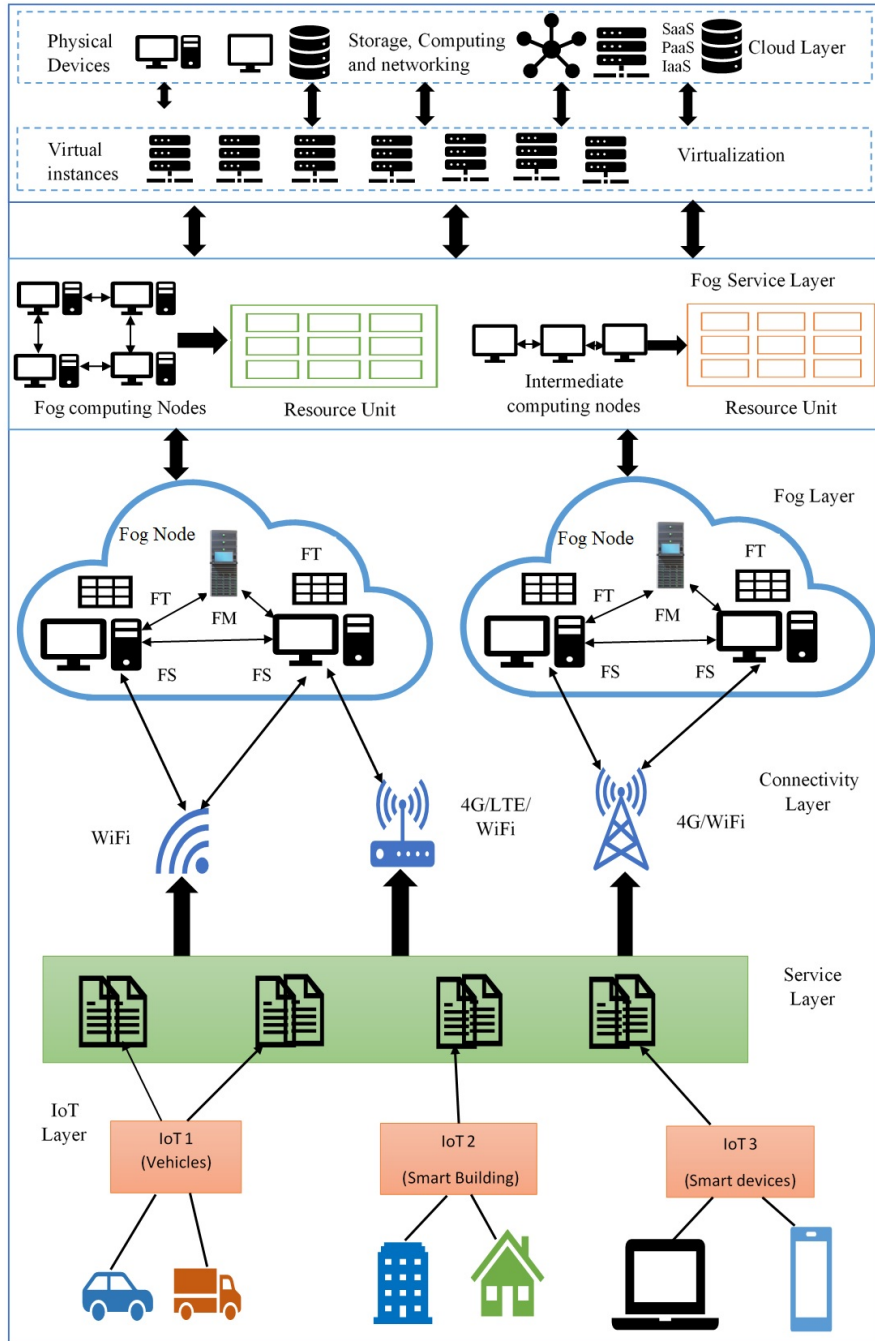


Figure 1.2: Layered architecture of fog computing

- **Fog layer:** The fog layer is divided into two layers, i.e., fog layer and fog service layer. Different fog nodes are deployed in the fog layer that receives tasks for processing from the IoT layer. Different Fog Servers(FS) are connected to each other in the fog layer, and these FS are managed by Fog Manager(FM). Each FS contains its Fog Table(FT) that includes the information about the number of tasks generated, the number of resources available, and several resources occupied. Based on the FT of each FS, FM decides about tasks received from the IoT layer. Most essential tasks are passed to fog computing nodes of the fog service layer containing their resource unit. The least important tasks are executed on intermediate computing nodes.
- **Cloud layer:** In Figure 1.2 cloud layer is divided into two different parts in which the first one is virtualization, and the second one is cloud layer. Virtual Machines(VMs) are available in virtualization on each Physical Machines(PM) in the cloud layer. Virtualization generates virtual servers on physical machines for the execution of tasks. The Cloud layer contains physical machines, networking devices, enormous data centers. The Cloud layer receives data from the lower layer and stores it for any future processing.

1.1.3 Characteristics of fog computing

As the fog sector develops, virtualization will be its key to success. The fog layer becomes necessary to connect the cloud to things because it has latency, mobility, bandwidth, and security issues. Simultaneously, not everything can run from the edge with intelligent endpoints because of space, energy, and security. Fog computing is an emerging technology that has many characteristics. A few of them are discussed below:

- **Geographically distributed:** Fog nodes are geographically deployed on different locations, unlike the cloud data centers that work in a centralized manner. Fog computing nodes can be deployed at any place and can be accessed at anytime that helps in maintaining real-time applications, for example, pipeline monitoring [15].
- **Mobile application support:** Fog nodes can not only be deployed at fixed locations but mobile also. Fog computing supports mobility by providing computational offloading and expansion in storage. For example, fog nodes can be deployed in connected rails.

- **Heterogeneous end-user support:** Fog nodes can be located near to the end-users at the network edge to provide immediate support in case of an emergency [15].
- **Mobility support:** Any mobile node can become a fog node, for example, cars, mobile phones, etc.

Fog computing aims to enhance efficiency and decrease data to be transferred to the cloud for processing. The fog layer can easily manage all the resources by sitting in between the cloud and IoT [17]. The fog layer helps in preparing a viable business model like the cloud layer. Fog computing can be used in transportation, smart cities, surveillance, health care, and intelligent buildings [18] [18]. Fog computing can be used in different types of IoT services [19]. First, e-Health gateway can be used for patients to monitoring their health status [20]. Second, Smart homes can be improved by detecting the temperature, and air conditioning system [21]. The emergency alarm can be activated and send warnings to the owner. Third, Smart cities can be monitored traffic, and transport systems by IoT [17, 18].

In a network, few systems remain under-loaded at some interval, while the others carry the entire load of the network. To maintain the load in a balanced scheme, "Load Balancing" becomes necessary. "Load Balancing strives to distribute the load in identical proportions throughout resources depending on recourse capability so that every useful resource is not overload or underutilized in a cloud device". Data centers in the cloud required minimizing and distributing the workload among all resources for the proper functioning at their higher degree of potential and controlling the magnitude of load distribution. Load balancing must also be done to avoid deadlock and suppress the server overflow problem [22].

1.1.4 Key areas focused by fog computing

Fog computing has three fundamental areas, i.e. Resource management, Data management, and Security. Figure 1.3 represents the main areas in which fog computing focuses. They are discussed below:

- **Resource management:** In the fog computing environment, users have to face the problems of allocation of resources due to uncertainty and distribution of resources, which causes heterogeneity, dynamism, and failures [23] [24]. Automatic resource management is required to consider all QoS parameters like availability, execution time,

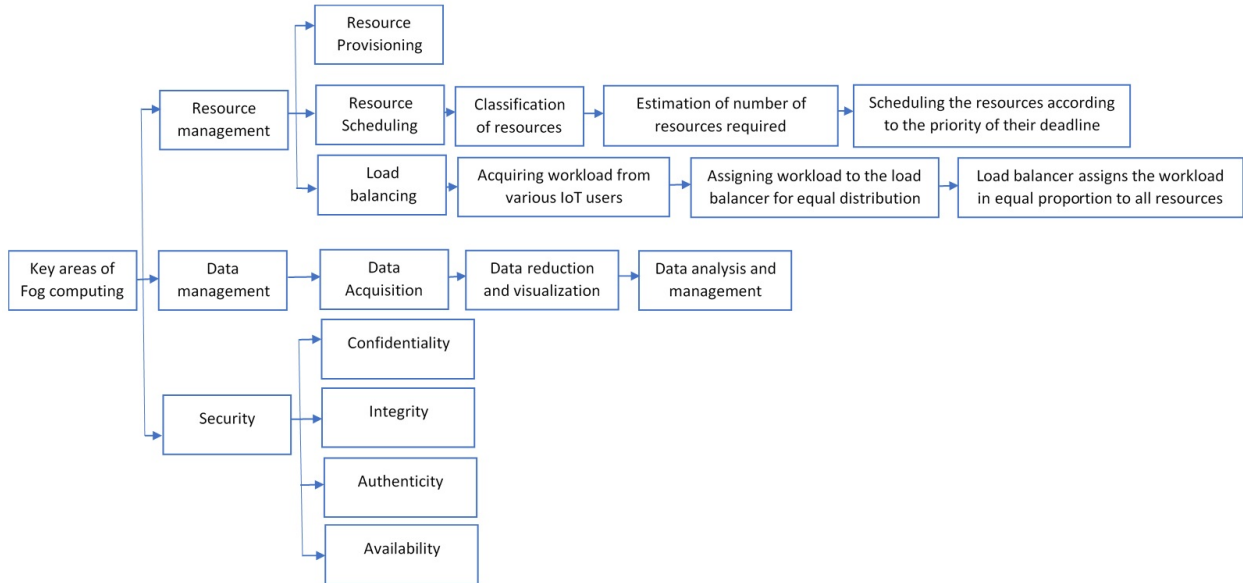


Figure 1.3: Key areas of fog computing

security application performance [25]. Resource Management System(RMS) contains three essential components:

Resource provisioning: Resource provisioning is required in fog computing to select and maintain all fog resources. Resource provisioning can be of two types: static provisioning, dynamic provisioning. In resource provisioning, software and hardware resources such as DBMS, different load balancers, CPUs, hard disks are selected, deployed, and managed. This ensures to enhance system performance [26].

Resource scheduling: Resource scheduling plays an essential role in administering the data centers that contribute to improving resource utilization. When resource provisioning is completed, resources are organized and scheduled by the resource scheduler for different users [27].

Load balancing: Load balancing is necessary to distribute the workload in equal proportion. Load balancing disseminates a similar proportion of workload amongst all the resources based upon their capability to utilize every resource efficiently. Load balancing ensures that there should not be any resources that are over-utilized or under-utilized. Appropriate load balancing helps to maximize resource utilization, reduce the response time and resource consumption [16] [28].

- **Data management:** The fog computing layer acquires the data from its lower layer, i.e., IoT devices. This data is validated and reduced at the fog layer to be processed or easily transferred to the cloud layer. Different analytical techniques are applied for data analysis [1].
- **Security:** Fog computing helps to provide secure data transfer over the network. The fog layer ensures data integrity and confidentiality to provide secure access to data. User authentication and authorization helps to avoid any unauthorized access to the data in the network [29].

1.1.5 Advantages of load balancing in fog computing

As discussed in earlier sections, there is a need for load balancing in fog computing to enhance resource utilization. This chapter has studied various load balancing techniques, and on the basis of reviewed articles, a load balancing-based framework for fog computing has been proposed. Here are some advantages of combining load balancing with fog computing, as discussed below:

- High availability: With load balancing, fog resources become highly available. In case when there is no response from one resource, others will be available for processing. Load balancer keeps track of under-loaded and overloaded resources to ensure the availability of resources in case of latency-sensitive applications.
- Flexibility: The load balancer distributes the workload among all the resources, so in case one resource fails, others will be available to process the data and respond to the users.
- Reduced energy consumption: All resources, whether executing task or in idle mode, consumes energy. With the help of load balancing, all the resources run tasks approximately in an equal proportion. So, by avoiding overloading and under-loading of resources, energy consumption in hardware nodes can be reduced.

- **Reduced Cost:** Fog load balancers are not expensive, as the cost depends upon the resources consumed. By applying load balancing in fog computing hardware cost, several resources may be used only according to the user requests.
- **Resource utilization:** By applying load balancing, all the resources in the fog computing environment can be fully utilized. Because load balancers avoid under-utilization and over-utilization of resources by distributing workload in equal proportion in all the resources.

1.1.6 Open issues and challenges

Various open challenges can be further worked upon. The following can be further explored:

- **Communication between fog nodes:** It can be further explored to find a framework through which fog nodes can interface with each other. If the fog nodes start communicating with each other's, they can also share their resources.
- **Detection of stolen Devices:** Smart devices are costly than other simple devices in the market. These can be stolen easily, so further work is required to protect these gadgets.
- **Security:** Security is the biggest challenge in today's world. Fog nodes need more security algorithms to be implemented. Any malicious person can try to enter into the system and temper the necessary information.
- **Load balancing:** Load balancing has become necessary in a fog environment. Because it will cause a problem if the few servers will be overloaded and the others will be under-loaded, if the load is not balanced in the system, it may cause a deadlock in the system. So, there is a need to develop efficient load balancing algorithms.
- **Energy efficiency:** Energy efficiency can be a huge challenge in fog computing. The power consumption of fog needs to be reduced. As fog nodes are more in the system, more energy is consumed. Need to manage the energy efficiently.

1.2 Scientific workflow applications in fog computing

This research work considered a few scientific workflows that are executed using the proposed technique. This research tried to reduce execution time, cost, and energy consumption in implementing these workflows. The overall efficiency of fog computing can be significantly increased by improving load distribution in scientific workflow applications. Since workflow scheduling is an NP-complete challenge, meta-heuristic methods are a better choice for optimizing it [30]. The existence of workflow benchmarks will significantly aid the design and assessment of workflow management systems. Following are the examples of scientific workflows that are considered to evaluate the proposed framework and algorithms.

Cybershake: The Cybershake computational pathway is modeled as a workflow and implemented in the grid-based SCEC environment [31]. The SCEC first used it to characterize earthquake hazards in the area. An MPI-based differential simulation is used to produce Strain Green Tensors (SGT) given the region of interest. Synthetic seismograms are created from the SGT data with each of the expected ruptures. The generation of probabilistic hazard curves follows this [32–34]

Genome: The Epigenome Center uses the genomic, a CPU-intensive program, to process output DNA methylation and histone modification results. The data is initially collected in DNA sequence lanes from the ISGA ("Illumina-Solexa Genetic Analyzer"). Each Solexa computer generates multiple DNA sequences. The ISGA is divided into various sections. Every chunk's data is translated into a file that the Maq framework can understand [35]. The workflow then performs the mapping of DNA sequences to specific positions in a genome. This generates a map that depicts the sequence density. Filtering out noisy sequences, translating sequences into the proper location in a genome, producing a global map, and determining sequencing abundance at each point in the genome are the remaining operations [30].

SIPHT: The Harvard University bioinformatics project was looking for small and untranslated RNAs (sRNA) to control various bacterial processes such as secretion and virulence. The sRNA recognition protocol using technological innovation software [36] employs a workflow to simplify the hunt for sRNA encoding genes in the NCBI database

for all bacterial replicons. Condor DAGMan’s [37] capabilities are used to perform a sequence of individual programs in the appropriate order for annotation and prediction of sRNA encoding genes. These include forecasting Rho-independent transcriptional function.

LIGO: Large-scale interferometers gather data for the Laser Interferometer Gravitational-Wave Observatory (LIGO) workflows used to look for gravitational wave signatures. The observer aims to quantify and detect waves in the manner expected by relativity. The workflow produces a subset of output waveform from the spatial domain for each section and evaluates the matching filter output. A trigger is created if a true inspiral is detected, which could then be compared to stimuli for all the other detectors [38]. This workflow [39] is used to interpret data from small binary systems, including binary neutron stars as well as black holes. Figure 1.4 represents a few scientific workflows, out of which LIGO, SIPHT, GENOME, and Cybershake have been considered for this research work.

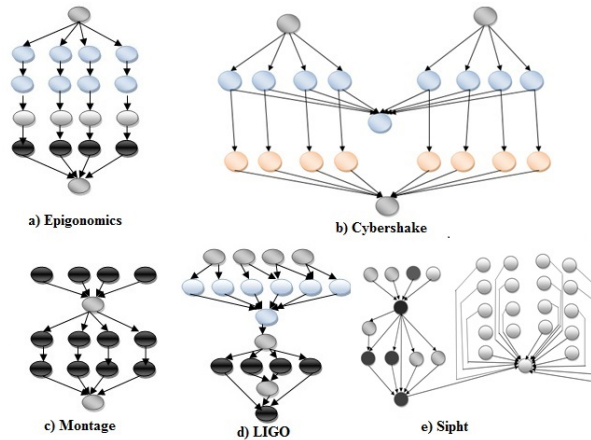


Figure 1.4: Examples of scientific workflows

1.3 Load balancing in fog environment

In computing networks, at different time intervals, few nodes remain under-loaded, while other nodes carry the entire load of the network. Due to the load imbalance on

the servers, there may be problems like system failure, network failure, energy consumption, increased execution time. [40]. Load balancing becomes necessary to control the load in computing nodes. Load balancing seeks to distribute the load equally throughout all resources according to their capability. With this, every useful resource does not become overloaded or underutilized in a fog environment [41]. Data centers in the cloud must minimize and equally distribute the workload for the proper functioning at the highest degree of potential and control the magnitude of load distribution. Load balancing also becomes necessary to avoid overflowing, and deadlock problems in server [42].

Load balancing is required to distribute a large volume of data on servers. With the equal distribution of the workload in the network, resources can be efficiently utilized. Load balancing has few characteristics, i.e., work is equally distributed in all nodes, efficient resource utilization, improved system performance, reduced energy consumption, more user satisfaction, reduced response time [43]. A few load balancing functions are given below:

- It distributes network load or the requests of the clients efficiently across multiple servers.
- Active nodes receive requests from end-users and ensure high availability and reliability by immediately processing their requests.
- It provides flexibility to the servers so that any server can be added to the network whenever required.

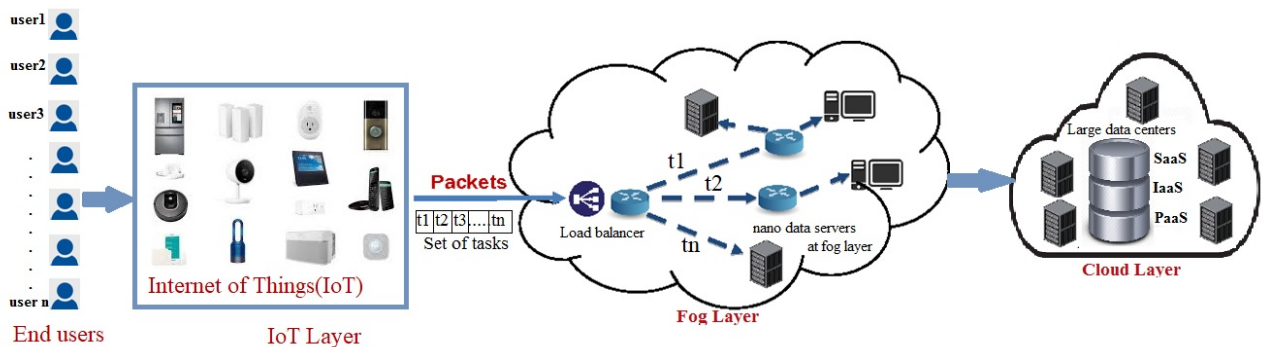


Figure 1.5: Load Balancing in fog environment

The Figure 1.5 shows how a load balancer receives the workload from various users and assigns this workload to the computing servers. A load balancer continuously keeps track of available servers in the network. When it receives the workload from different clients, it checks the availability of resources. It then distributes the load among all the computing resources to avoid overloading situations in the network.

1.3.1 Need for load balancing in fog computing

With the expansion in IoT in the digital world and a gradual increase in real-time applications, the need for equal distribution of workload in the fog environment has increased [42]. Load balancing helps to achieve high resource utilization and more user satisfaction. With this, the overall performance of the system and the resource utility will also improve. It ensures that no resource either becomes overloaded or remains underutilized. By distributing the workload among all the processors equally, the overall operational cost of the system can be reduced, and user requests can be optimally balanced. In cloud-fog-based architecture, users continuously pass their requests in bulk amount to the fog nodes, which needs to be minimized to make exclusive use of fog nodes [44]. The load balancer at the fog layer receives the task processing requests from IoT devices and distributes the tasks among all processing nodes.

The main requirement of load balancing is to surmount the problems faced by overloaded resources at the fog layer. The tasks assigned to virtual machines can either be reliant or independent of VMs. The load is divided into different types, i.e., CPU load, storage devices, network load. These loads are divided into three categories based upon the place and requirement of the user, i.e., a) critical load, b) Non-critical load, c) Highly critical load. Load balancing is the process of detecting overburden and small loaded nodes and then balancing the workload among all of them. By proper utilization of fog resources, system performance can improve. Fog resources may be hardware resources or virtual resources. The load balancer does these tasks for proper load balancing: 1) Allocating tasks to the physical machines, which are further assigned to the virtual machines. 2) Transferring workload between two physical machines or virtual machines.

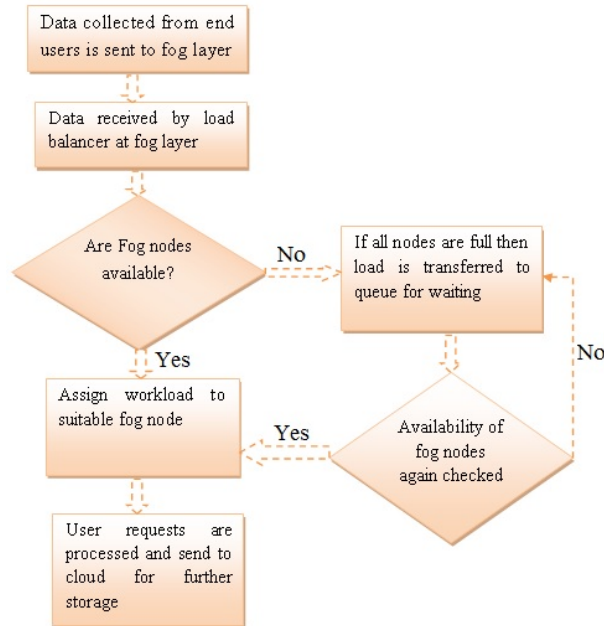


Figure 1.6: Flow diagram of load balancing at fog layer

The flow diagram in Figure 1.6 explains how load balancing works in fog computing. In a combined cloud-fog-based computing environment, workload management becomes compulsory at both the fog layer and the cloud layer. The fog layer helps to manage the workload transmitted to the cloud layer. Fog layer receives data from smart devices, then load balancer at fog layer checks the availability of fog nodes, so that workload can be assigned to free fog node. If the fog nodes are available, the workload is equally assigned to the free fog nodes in the computing environment. If all the nodes of the fog layer are occupied, then the workload received from the smart devices has to wait in the queue until it gets any free fog node. The load balancer continuously checks for the availability of fog nodes. Whenever free fog node is found, then the workload is assigned to the fog node. Highly time-sensitive requests are processed at the fog layer itself, and their corresponding results are transmitted to the cloud layer for more processing and storage.

1.4 The motivation of study

With the increased number of IoTs, the workload on the fog layer has been gradually increased. Due to more workload on fog devices, the power consumption of nodes is also high. Micro data centers at the fog layer are aided by load balancers responsible for distributing the workload amongst multiple fog nodes to optimize resource utilization and enhance response time. The load balancers ensure that the workload is equally divided among the idle fog nodes to avoid the overloading of a few fog nodes. If there is an imbalance of workload at the fog layer, it directly impacts the user response and real-time event detection. Micro data centers of the fog layer are deployed in remote environments, due to which the need for secure authentication is of significant importance. An efficient load balancing framework must identify the idle edge data centers so that workload can be equally and efficiently divided among all idle fog nodes. Load balancing is required at the fog layer to achieve resource efficiency, avoid overload in the network, maintain system firmness, improve system performance, and protect the system against failures. Fog load balancing provides the following benefits to the computing environment, which motivates towards the study of fog load balancing:

- High scalability.
- High energy efficiency.
- High resource utilization.
- Reduced cost.
- Reduced latency
- More flexibility

1.5 Thesis contribution

The significant contributions of this research work are described as follows:

- It has analyzed and provided detailed literature on load balancing in the fog computing environment. A qualitative comparison between existing load balancing

- techniques in fog computing has been provided based on parameters considered, i.e., execution time, energy consumption, response time, processing time, etc.
- Different parameters, i.e., Time delay, cost, and energy consumption, have been identified that helps to propose and design a resource-utilization-based load balancing framework for the fog computing environment.
 - A framework has been proposed for a fog computing environment that helps to improve the utilization of existing resources by equally distributing the workload among all resources.
 - Fog computing architecture is provided, showing load balancing and scheduling in the fog layer. There are many existing fog computing architectures, but this research enhanced the traditional architecture by providing load scheduling and load balancing at the fog layer. Generally, traditional architectures implement load scheduling or load balancing only. However, this work provided scheduling and load balancing, which will improve system performance by speeding up execution.
 - A resource utilization-based load balancing framework is proposed for the fog computing environment that is named as FOCALB. A detailed flow diagram has been provided that explains the whole procedure of load balancing done in the fog layer in the proposed architecture. When tasks get resources for processing in time, it helps reduce wastage of resources, which will help reduce the energy consumption of those resources.
 - Hybridized load balancing algorithm for scientific workflows (Tabu-GWO-ACO) has been proposed, which helps to utilize fog resources properly and reduce the cost, execution time, and energy consumption in fog nodes. The proposed approach's obtained results are compared with the existing tabu search method, ACO, GWO, and Artificial Bee Colony(ABC) algorithms to analyze and compare its efficiency.
 - A prototype model has been proposed for implementing proposed architecture FOCALB in real time application i.e. Smart waste management.
 - Resource-utilization based workflow execution model has been proposed for fog

computing. This model helps to apply load balancing to enhance resource utilization in fog computing.

- A hybrid load balancing approach, PSW-Fog Clustering tries to reduce energy consumption in a fog computing environment while executing scientific workflows. The work considers both the availability of tasks and workflow structure. The work tries to reduce system overheads, minimize resource wastage, and enhance tasks' execution speed.
- A framework has been proposed for the implementation of workflows. The proposed work considers scientific workflows for execution in fog environments, i.e., LIGO, Genome, SIPHT, Cybershake workflows.

1.5.1 Thesis organization

After giving an introduction to the thesis in chapter 1, the rest of the thesis is structured as follows:

Chapter 2 presents the literature survey on fog computing and load balancing techniques. Load balancing techniques in fog computing have been thoroughly studied, and taxonomy has been provided representing existing load balancing techniques. Moreover, a year-wise review of load balancing techniques has been done and delivered in taxonomy. A comparison of different load balancing techniques has been described based on various performance parameters. The chapter is concluded by formulating the problem and providing its relative solution. Chapter 2 is partially derived from:

- Saroa, M. K., Aron, R. (2018, December). Fog computing and its role in the development of smart applications. In 2018 IEEE Intl Conf on Parallel Distributed Pro-cessing with Applications, Ubiquitous Computing Communications, Big Data CloudComputing, Social Computing Networking, Sustainable Computing Communications(ISPA/IUCC/ BDCLOUD/SocialCom/SustainCom) (pp. 1120-1127). IEEE
- Kaur, M., Aron, R. (2021). A systematic study of load balancing approaches in the fog computing environment. The Journal of Supercomputing, 1-46.

Chapter 3 describes the proposed architecture for load balancing in scientific workflow applications in the fog computing environment. The proposed architecture is named FOCALB. Existing frameworks have been studied and proposed an enhanced architecture implementing load balancing in a fog environment to enhance resource utilization and reduce energy consumption and execution time. The working framework has been described in the form of a flow diagram that explains the whole procedure of load balancing done in the fog layer in the proposed architecture. Along with this, a hybridized load balancing algorithm for scientific workflows (Tabu-GWO-ACO) has been submitted, which mainly helps to utilize fog resources properly and reduce the cost, execution time, and energy consumption in fog nodes. The proposed approach's obtained results are compared with the existing tabu search method, ACO, GWO, and Artificial Bee Colony(ABC) algorithms to analyze and compare its efficiency. The proposed framework has been validated by comparing it with existing frameworks based on considered parameters, i.e., execution time, energy consumption, and resource utilization. The experimental results have been obtained by implementing the proposed approach in the iFogSim toolkit. Chapter 3 partially derives from:

- Kaur, M., Aron, R. (2020). Equal Distribution Based Load Balancing Technique for Fog-Based Cloud Computing. In International Conference on Artificial Intelligence: Advances and Applications 2019 (pp. 189-198). Springer, Singapore.
- Kaur, M., Aron, R. FOCALB: Fog Computing Architecture of Load Balancing for Scientific Workflow Applications. J Grid Computing 19, 40 (2021).
<https://doi.org/10.1007/s10723-021-09584-w>
- A novel load balancing technique for smart application in the fog computing environment, International Journal of Grid and High-performance computing(IJGHPC), IGI Global [Accepted]

Chapter 4 describes the proposed resource-utilization-based workflow execution model for fog computing. This model helps to apply load balancing to enhance resource utilization in fog computing. Along with this, the chapter also describes the proposed hybrid load balancing approach PSW-Fog Clustering that tries to reduce energy consumption in a fog computing environment while executing scientific workflows. Further,

this chapter analyses the proposed algorithm, and obtained results are compared with other existing algorithms. Chapter 4 derives from:

- Kaur, M., Aron, R. (2020). Energy-aware load balancing in fog cloud computing. *Materials Today: Proceedings*. Elsevier.
- An energy-efficient load balancing approach for scientific workflows in fog computing, *Wireless Personal Computing*, Springer [Communicated]

Chapter 5 finally concludes the thesis and provides the future scope of the proposed work.

CHAPTER 2

Literature Survey

Fog computing is still not adopted as a mature computing paradigm, as some of its areas still need to be explored areas as load balancing, energy, resource utilization. Much research works in the area of load balancing in cloud computing, but lesser in fog computing. More focus is provided on scheduling in scientific workflows, but load balancing in scientific workflows also seeks attention. Fog computing has a broad scope in its application areas, so many intelligent devices generate more data which causes load imbalance in the fog layer. So, load balancing also becomes necessary in fog computing.

The fog layer requires to balance a workload among all the resources. Requests received from users are assigned to the fog nodes for an immediate response, and if there are many user requests, they should be equally distributed among all fog computing nodes. Hence, load balancing has become a necessity on the fog layer. It is required in both physical nodes and virtual machines as well. The load balancing mechanism disseminates the load fairly among host and virtual machines. This chapter reviews and summarises the existing load balancing algorithms. Firstly, the growth of smart devices has been discussed, and then the motivation behind this chapter has been provided. The other section covers the load balancing related surveys that various researchers conduct. Furthermore, load balancing techniques have been described thoroughly. The load balancing techniques have been explained, and their comparison has been provided based on multiple parameters. Moreover, related work in fog computing has been provided based on three categories, i.e., cost-based, resource-utilization-based, and energy-aware load balancing approaches. In addition to this, a year-wise review of existing load balancing approaches has been conducted. Taxonomy has been provided as an explaining focus of the current studies. Moreover, performance measured those impact load balancing has been discussed. Further, open issues and research challenges have been described. In the last, the problem has been formulated, and research objectives have been provided.

2.1 Growth of smart devices

Internet of Things(IoT) daily produces a huge volume of data that is called Big data [45], that needs analysis [46]. Since 2015 total of 15 billion IoT devices has emerged that include sensors. The connected devices in IoT contain various smart devices, i.e., smart wearable devices (helping aids, glasses, and smartwatches), intelligent electric grids [47] smart-cities [48], that measures energy consumption in smart homes, sensor networks [49], self-driving vehicles [50].

Figure 2.1 shows the anticipated growth of connected devices in the coming years. Cisco Systems, Inc. reports that the connected devices will grow up to 50 billion in 2020 [51]. But, in the current scenario, many researchers have analyzed the available number of fog nodes in the current year 2020, i.e., approximately 31.73 billion devices are available till the end of 2020 [52]. This ratio will start increasing from 15-25 percent to 50 percent within the coming years and may reach up to 150 billion by 2030 [53].

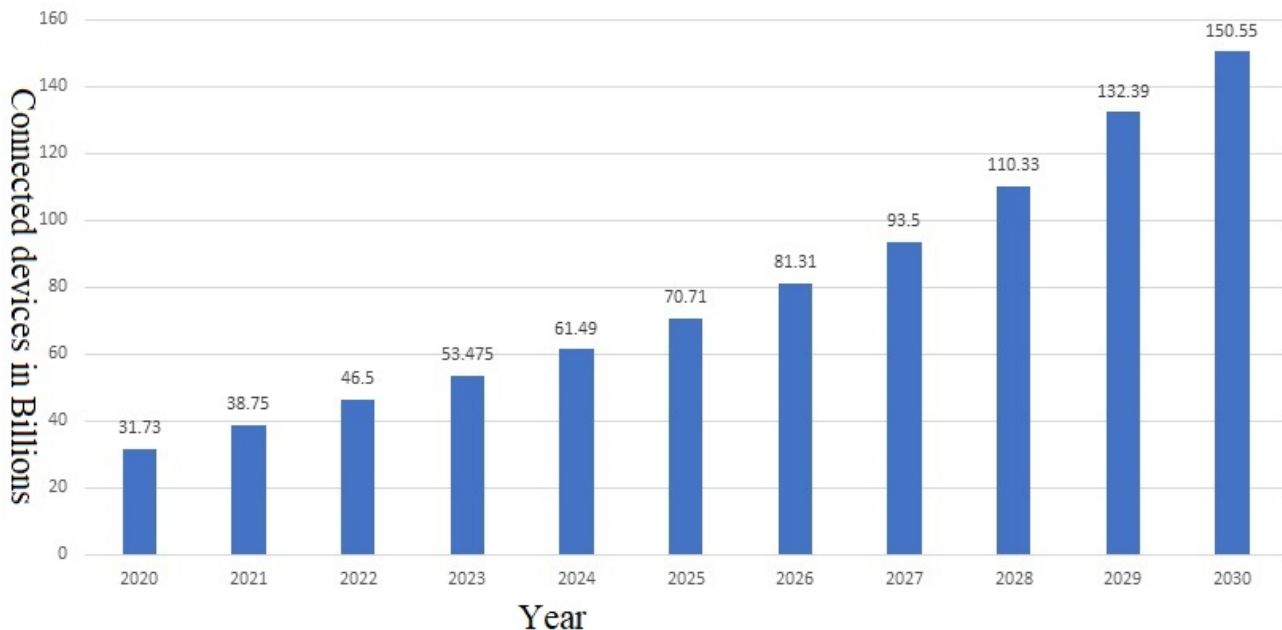


Figure 2.1: Growth of connected devices from year 2020-2030

These devices consume a large amount of energy, which needs to reduce to prevent any

failure in the system. IoT data is stored and processed in cloud data centers. But, cloud computing is only suitable for tasks requiring high latency and less service availability [41]. The growing IoT devices require a platform that reduces data transmission and energy consumption by cloud data centers. Fog computing was established to provide low latency and high availability services to IoT users. Fog computing is a highly virtualized technology that offers real-time interactions as it sits amongst the cloud data centers and IoT users [54] [55]. The fog layer is connected via WiFi, Bluetooth, and 4G LTE wireless internet access with nearby IoT devices [56].

With the expansion in IoT devices, the increased volume, velocity, and variety of data have led to growth in Fog computing [57]. IoT also requires a real-time interaction platform for the execution and storage of the data generated by these smart devices. Fog computing provides features like location awareness, edge data centers deployment, the geographical distribution of nodes. The fog layer contains small servers, smart routers, or other equipment that provide a data transmission band between different devices [58]. Fog computing was introduced in 2012, and now it has a vast scope in many applications like smart cities, smart healthcare, smart transportation. [48].

2.1.1 Motivation

This study finds the increasing demand to handle the workload on resources present at the fog layer and inquire about its precedent load balancing approaches. A comprehensive review has been conducted to evaluate load balancing approaches of fog computing, and these approaches were compared based on different metrics. The following facts particularly inspired this chapter:

- 1 An increased need to understand precedent load distribution techniques in fog computing. By balancing the load provided to the fog layer, resource utilization can be enhanced, which will help to reduce the wastage of resources at the fog layer.
- 2 The energy wastage by fog nodes is also increasing, which may cause failure in the fog network. So this environment needs optimal support to reduce the energy consumption in fog nodes.

2.2 Load balancing techniques

By the process of load balancing, performance improved by shifting workload among the processors. Workload means- the total time of processing required to execute the task assigned to a machine. Load balancing assigns load to underloaded VMs by taking it from overloaded virtual machines. Load balancing affects system performance while the application is executing. Load balancing aims to enhance the execution speed of applications on resources because their execution time is unpredictable as it varies at run time.

Load balancing in fog computing is applied to the physical nodes as well as the VMs. In load balancing, the load is distributed among all the processing nodes in equal proportion. Load balancing algorithms are divided into two main kinds, i.e., Initiation process-based and current state-based. The first type further divided into three types, i.e. sender initiation based [21], receiver initiation based [59], depends upon both sender and receiver [60] [61] . The second type is further categorized into two types static and dynamic, which can explain as follows:

a) Static load balancing: In static load balancing, the equivalent distribution of load among the servers gives prior information of the applications and statistical information. In static techniques of load balancing, those systems are considered in which load variation is very low. The load is distributed among all the servers. Prior knowledge of server resources is required to start implementation. The present system state is not considered in these algorithms. There are some load balancing algorithms available which are explained as follows:

In the round-robin method, the first node is selected randomly, and all other nodes are assigned the jobs in a Round Robin fashion in which time slots are allocated to each process. The tasks are assigned in a circular organization to the processor without giving priority to any particular task. [62].Min-Min load balancing technique is used for static load balancing [8] [62] [63]. It is suitable for small tasks only. Those tasks considered, which are less time-consuming. Less time-consuming tasks are allocated to the resources first. The processing of task will depend upon the execution time, i.e., the task having minimum execution time is allocated, whereas the tasks having maximum execution time will be on stand by till the processor becomes free [64]. Like

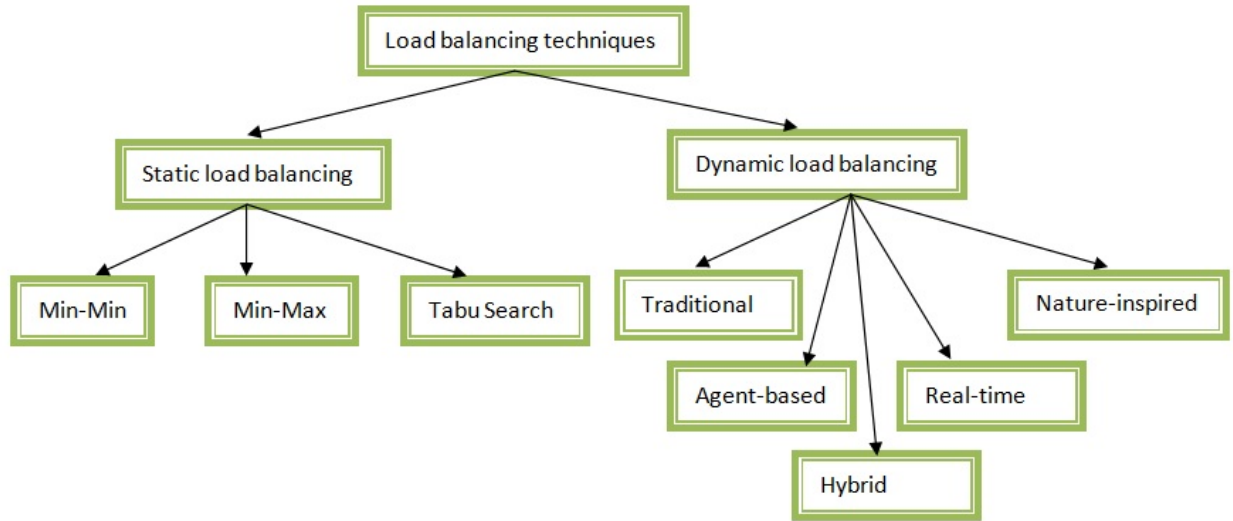


Figure 2.2: Load balancing techniques

the Min-Min algorithm, the Max-Min works to find out the minimum execution time, but the only exception was that it firstly deals with the tasks that took maximum time for execution. [62]. Once the task gets the resource, it is removed from the task list, and only pending tasks remain in the list, and their execution time is updated.

Tabu Search method is used for optimal load balancing between the fog and cloud layers, which helps avoid resource constraints, i.e., computational cost and memory usage. Finding the best task allocation at the fog layer is its main objective to optimize the time and cost by finding the best task allocation at the fog layer. Tabu search method ensures online computations in the fog layer to increase the immediate processing of tasks at the fog layer only. Tabu search method includes a bi-objective function to estimate the cost at fog layer as well as cloud layer [28]. Opportunistic Load Balancing Algorithm (OLB) tries to keep every node busy without considering the existing workload of each resource. OLB need not calculate the time of execution of tasks. OLB tries to balance the load between all the nodes to keep every node in working condition. Tasks are randomly assigned to available nodes so that present nodes can be fully utilized [65] [41].

b) Dynamic load balancing: The lighter server is searched in the dynamic load

balancing approach, and it is preferred for load balancing. The present system state controls the network load, and these algorithms use real-time network communication. The dynamic technique of load balancing finds out how to search for under-loaded servers and then nominate an appropriate load of work. The load is transferred from over-utilized virtual machines to under-utilized in real-time. In this, the workload is shared between the processors at run time. This technique is considered to have complicated algorithms, but their fault tolerance and overall performance are superior. Different researchers gave several load balancing algorithms. Here a review of existing dynamic algorithms has been provided. A few dynamic load balancing approaches studied categorized as follows:

- Traditional techniques
- Nature-inspired techniques
- Agent-based techniques
- Real-time techniques
- Hybrid techniques

i) Traditional techniques There are various existing algorithms of load balancing used as per user requirements. Breadth-First Search (BFS) algorithm is used for load balancing among fog nodes. The selected node is searched first, whether it is the starting node. It traverses the entire network layer-wise, thus exploring the neighboring nodes (nodes directly connected to the source node). Load balancing can start from the source node, and then the next-level neighbor nodes are searched [29]. With the help of graph partitioning of virtual machine nodes, load balancing of task allocation to various nodes was achieved. Graph partitioning theory can be used to enhance the load balancing in fog computing nodes [66] [67].

Dynamic resource allocation method (DRAM) is used as a resource in a fog environment. It is effective and dynamic that works as an aid to provide equilibrium among the load balance and situations like the slow running of systems to avoid them. The jobs reaching their deadlines should be immediately assigned to the processor to complete their execution within its deadline. So DRAM can help to balance the Load in Fog

computing [68] [16]. Real-Time Efficient Scheduling Algorithm (RTES) is the method with which the help of the available bandwidth can balance load and execution of the real tasks before their deadline. So that lesser time should be taken by intermediate fog layer to give a response to their clients. [54].

The throttled algorithm assigns one task to a virtual machine at a time, while the execution of tasks status of the virtual machine remains busy. VM can only perform a single task at a time, rather than executing multiple tasks [69]. In this algorithm, a list of virtual machines is prepared by the task manager. Tasks are assigned to the virtual machines depending upon their size and capability. Load balancer maintains its index table to keep the record of the status of virtual machines. When the client or server requests a virtual machine of the data center, the load balancer scans its index table and checks the availability of VMs. If VMs are available, free VMs are allocated to the requesting clients/servers [70].

X. Xu et al. [16], and H. Qiao et al. [71] suggested different methods for allocation of resources in a fog environment for load balancing. Fog computing has given a system scheme for the inquiry of diversified computer nodes and their load balance. A method is designed to balance the load in the fog environment that works through dynamic service migration is named resource allocation. The aim of this technique is load balancing in all computing layers: cloud and fog nodes. To boost up the load balancing during the execution of fog services is called a global resource allocation strategy.

Y. Yu et al. [72] presented a new design of a Scalable and Dynamic Load Balancer (SDLB) for accomplishing the requirements of fog and mobile edge computing. Mobile Edge Computing (MEC) come up with the migration of computing and stockpile services from the remote cloud to the edge of the network. The performance of SLDB is evaluated and compared with other adopted approaches for load balancing. The POG has a core algorithm named minimal perfect hashing, and on this ground, a new data structure was built named SDLB.

S. H. Abbasi et al. [73] proposed a four layers model for the fog with integration to smart grids. The main perspective of their advanced design is to govern the resources of a smart grid—the suggested design and work distributed like the six continents of the world. Four algorithms were used in this work, i.e., Round Robin (RR), Throttled, Particle Swarm Optimization (PSO), and Active VM Load Balancing Algorithm

(AVMLB) for load balancing and compared their results based on cost.

A. B. Manju et al. [43] proposed a four-layered architecture of fog computing environment in which fog nodes act as proxy servers between the cloud data centers and end-users. The fog layer reduces the workload on the cloud service providers [22]. Constraint-Based Min-Min Algorithm used for load balancing in fog environment. V. Velde et al. [41] designed a load-balancing algorithm using a round-robin in Virtual Machine (VM). Load balancing was found out to be a common and tough issue in cloud computing. The load is assigned to VM through the processor, and load on the cloud is stabilised using fuzzy logic. A digital system processor's speed and load are considered two contributory parameters to contribute to the fuzzifier in cloud load balancing. FRR load balancing algorithm carries a higher load in cloud computing than the conventional Round Robin algorithm [74] [19].

Table 2.1: Traditional techniques

Author	Algorithm proposed	Experimental Parameters	Approach used	Advantages	Disadvantages
Xu, X. <i>et al.</i> [16]	DRAM	Execution time, Cost	Resource migration	<ul style="list-style-type: none"> • Minimized load balance variance. • Improved resource utilization. 	Service migration cost not included.
Y. Yu <i>et al.</i> [72]	SDLB	Memory efficiency, update speed, and throughput	Minimal perfect hashing	<ul style="list-style-type: none"> • Less memory is used. • Efficient Resource utilization 	Implementation cost not considered.

continued on next page

continued from previous page

Author	Algorithm proposed	Experimental Parameters	Approach used	Advantages	Disadvantages
S. H. Abbasi <i>et al.</i> [73]	Active VM Load Balancing Algorithm (AVMLB)	Response time, Cost, Energy consumption	Demand side management	<ul style="list-style-type: none"> • Cost of energy consumption is reduced. • Response time is reduced. 	Security can also be considered.
D. Puthal <i>et al.</i> [29]	Adaptive behaviour based	Response time, security	EDC authentication Edge data centres are considered and BFS is used.	<ul style="list-style-type: none"> • Enhanced efficiency. • Increased Security. 	No real time scenario is considered for results.
Dou, W <i>et al.</i> [75]	Resource Co Allocation method	Start time, Resource utilization, variance	High performance computing applications	<ul style="list-style-type: none"> • Improved resource utilization. • Improved cost and performance. 	Energy consumption is not considered.

ii) Nature-inspired techniques

These kinds of the algorithm based upon habit of seeking food of animals or insects. Different researchers have provided many algorithms containing the constant experience of nature: Honey bee behavior-based load balancing algorithm based on the Honey bee behavior of collecting food from the beehives. The honey bees came back to their hives in a dancing mode called dancing after collecting food. This dance form of bees shows the quality and quantity of remaining food in the beehives. These bees inform

the other bees about the foods in the beehives by showing their joyful dangling dance form, which is like an analog waveform. For load balancing in VMs, the honey bee's foraging behavior is followed. The earlier removed task helps find the lightly loaded VM. In this way, the honey bee behavior is used to balance the load in the VMs [76]. In the Ant colony algorithm, the behavior of Ant while searching the food is considered. Ants follow the way which other ants follow. The ant colony algorithm reduces the makespan and also balances the load. If more ants choose a path, that means that path has high computation power. The tasks in the ant colony algorithm are independent of each other, computationally rigorous [77]. L. D. Dhinesh Babu [78] proposed an algorithm called honey bee behavior-based load balancing (HBB-LB). The fundamental objective of this algorithm is load distribution among all virtual machines. The average time of execution and declining waiting time of the task can be notably be affected by this. Other existing load balancing approaches are also discussed in this chapter. This algorithm acknowledges the task preferences assigned to VMs. The load balancing in VMs is dependent upon the honey bee's food collecting process and behavior while collecting food.

De Falco et al. [79] suggested a load balancing technique for the dynamic environment based upon Extremal Optimization (EO). EO algorithm is used for task migrations during the load balancing process. Some factors are considered to solve dynamic methods, i.e., target nodes and fitness function selection. The authors compared the proposed algorithm with the existing techniques which were proposed by them, i.e., sequential extremal optimization-based algorithms.

Mishra, R. et.al. [40] proposed Ant Colony Optimization(ACO) based load balancing technique. Authors mainly consider load balancing in a distributed environment which improves response time and utilization of resources. The proposed algorithms consider the behavior of ants for collecting food. The ants work mainly self-organized than learning; they mainly follow reinforcement learning. An optimal path was followed to find food. Babu and Samuel [80] in their recommended approach, i.e., QoS, considered response time, task migrations as their criterion. In their publication, they represented honey bees as task governors and VMs as their food resources. To diminish the load on overloaded VM, the task with the lowest preference has emigrated from one VM to the other VM. The recommended algorithm with PSO and ACO considered augmenting

its performance.

Table 2.2: Nature-inspired techniques

Author	Algorithm proposed	Parameters considered	Approach used	Advantages	Disadvantages
L. D. Dhinesh Babu <i>et al.</i> [78]	HBB-LB	Response time, makespan, priority	Task priority	<ul style="list-style-type: none"> • Improved execution time. • Reduced waiting time. 	Only independent tasks are considered for load balancing.
Hussein, M.K. <i>et al.</i> [81]	PSO, and ACO based Meta-heuristic algorithm	Response time, and communication cost	Foraging behaviour for finding food sources	<ul style="list-style-type: none"> • Improved response time. • Reduced computation cost. 	Power consumption of nodes is not considered.
De Falco <i>et al.</i> [79]	Extremal Optimization (EO)	Cost, efficiency, execution time	Task Migration	<ul style="list-style-type: none"> • Reduced execution time. • Task migration is reduced. • Increased resource utilization. 	Graph optimization is not considered.

continued on next page

continued from previous page

Author	Algorithm proposed	Parameters considered	Approach used	Advantages	Disadvantages
Babu and Samuel <i>et al.</i> [80]	Honey bees foraging behavior	Makespan, Response time	Honey bee technique is used to reduce the response time and improve resource utilization.	<ul style="list-style-type: none"> • Low response time. • High resource use. • Lower number of task migrations. 	Less scalability, More complexity.
Devi <i>et al.</i> [82]	Weighted Round-Robin(WRR)	Completion time, Task migration	Execution time is considered.	<ul style="list-style-type: none"> • Reduced response time 	Only homogeneous environment is considered for execution.
Mishra, R. <i>et al.</i> [40]	Heuristic algorithm has been considered based upon ACO to balance load	Makespan, energy consumption, throughput	Heuristics algorithms are analyzed	Minimized CPU load, memory consumption, load on network.	No clusters are constructed

iii) Agent-based techniques

The agent-based technique works upon real-time information. Each server has an agent in the server pool that keeps track of its current load and provides it to the load balancer. The load balancer decides according to this information while assigning the tasks to the servers. Singh, A *et al.* [83] proposed Agent-based Automated Service Composition (A2SC) for resource provisioning in the cloud environment. The authors mainly focus on the reduction of VM costs and equal distribution of resources. Java

has been used to get the experimental results. They considered four data centers having different platforms. The main aim is to provide efficient service allocation in the data centers.

Alam, M.G.R et.al [84] [85] proposed Multi-agent based offloading in mobile fog. Reinforcement learning-based techniques are used to reduce the latency of delivering services to mobile users. Mobile codes are deployed on mobile fogs that are geographically distributed. Agents act as the entity that has pre-knowledge about the environment. Agents learn from the environment. This method aims to reduce execution time and reduce the time to access services by mobile users. OmNet++ is considered to find the simulation results.

Chen, C et al. [86] developed an agent-based task assignment technique for load balancing in the cloud. They have implemented fair competitive and dynamic adjustment principles for task allocation. The primary purpose of this work was to improve resource allocation and resource utilization. Simulation results have been taken using CloudSim. With this technique, the processing time increased.

Table 2.3: Agent-based techniques

Author	Algorithm proposed	Experimental Parameters	Approach used	Advantages	Disadvantages
Singh, A <i>et al.</i> [83]	Agent based Automated Service Composition (A2SC)	Response time, resource utilization	Modularization of agents.	<ul style="list-style-type: none"> • VM cost is reduced. • Complexity is reduced. 	Increased cost for maintaining security.
Alam, M.G.R <i>et al.</i> [85]	Multi-agent based	Execution time, energy consumption, latency	Distributed reinforcement learning	<ul style="list-style-type: none"> •Reduced latency. •Reduced execution time. 	Privacy of mobile users is not considered.

continued on next page

continued from previous page

Author	Algorithm proposed	Experimental Parameters	Approach used	Advantages	Disadvantages
Chen, C <i>et al.</i> [86]	Agent Based Allocation Algorithm (ABAA)	Execution time, response time	Fair competition and dynamic adjustment principle.	<ul style="list-style-type: none"> • Improved resource allocation and utilization. 	Processing and transmission time is more.
Keshvadi <i>et al.</i> [87]	Multi-agent based load balancing technique	Makespan, degree of imbalance, response time	Multi agents helps to maximize the resource useage.	<ul style="list-style-type: none"> • Reduced response time. • Better resource use. • Improved make span. 	DcM agents do not have timers to self destroy themselves.

iv) Real-time based techniques

These kinds of algorithms exhibit real-time behavior. These algorithms try to improve latency and execution time. A scheduling table for load balancer is formed before the simulation, and this table is modified at run time according to the changes that occurred dynamically. M. A. Elsharkawey *et al.* [88] developed real time-efficient (RTES) algorithm for fog load balancing. According to their recommendations, fog computing architecture furnishes a setup approach for load balancing. The algorithm concentrates on real tasks to be achieved within a definite period and increases throughput and network utilization. The intention of RTES balances the load by utilizing bandwidth and responding to the clients in a limited period. CloudSim tool is pre-owned to resolve the load balancing algorithm in the fog computing environment.

According to B. Sotomayor *et al.* [89] and Dsouza *et al.* [62] round-robin algorithm is used for load balancing in a fixed environment. In this, the resources are merited on their time-sharing manner, i.e., based on the first-come-first-serve (FCFS). The node that will have the least number of connections is assigned the task.

Table 2.4: Real-time based techniques

Author	Algorithm proposed	Experimental Parameters	Approach used	Advantages	Disadvantages
M. A. Elsharkawey <i>et al.</i> [88]	RTES	Turnaround time, throughput	Real tasks completion within deadline	<ul style="list-style-type: none"> • Improved throughput. • Increased network utilization 	Less scalable and more complexity.
Verma, M. <i>et al.</i> [54]	Real time efficient scheduling algorithm	Network utilization, turnaround time, throughput	Real time streaming	<ul style="list-style-type: none"> • Less bandwidth utilization • Low fault tolerance. • Increased throughput. 	Low efficiency.
Wang, J. <i>et al.</i> [90]	Discrete-differential evolution approach has been used to design fog node deployment algorithm	Time, space, load	space time characteristic based strategy has been used	<ul style="list-style-type: none"> • Response time is reduced • Improved real-time performance. • Effective load balancing. 	Energy consumption in fog nodes is not considered.

v) Hybrid techniques

These kinds of algorithms are in a combination of one algorithm to the other. These approaches combine characteristics of two or more different techniques to make an efficient approach for load balancing. A centralized controller controls all the functionality

in a meta-heuristic approach, i.e., data interchange, mutation-based upon fitness function, to better map virtual machines hosts. In a meta-heuristic approach, a problem can be solved by following a few sets of operations. Mainly this algorithm is inspired by other existing load balancing algorithms, i.e., honey bee behavior-based load balancing, particle swarm optimization (PSO), ant colony optimization (ACO) [91] [92]. N. Song et al. [67] presented a scheme owned to transform physical nodes into virtual nodes by cloud automation technology in the fog computing environment. The graphic representation given in which fog nodes were expressed by vertex and task dependencies and used the edges expressed bandwidth. A layered framework was provided, and the graph was constructed using a Cloud automation system.

H. Menon et al. [66] provided graph repartitioning and mapping-based load balancing algorithms for parallel computing. They provided an automatic load balancing system along with an adaptive run time system. S. Verma et al. [93] put forward a productive algorithm for load balancing for a fog-cloud-based architecture. Data replication is used to minimize the complete dependency on big data centers and maintain fog networks. Data replication in datacentres is used to reduce the use of bandwidth [93]. The data replication technology emphasizes data maintenance and reduces overall dependency in fog networks on big data centers. They also compared the present cloud-based load balancing techniques with fog-cloud-based techniques. This algorithm aims to balance load through the fog network, which reduced the cloud dependency of the internet users by providing data closer to the users. They proposed a three-tier fog computing architecture comprised of the ground tier, edge tier, and core tier. CloudSim 3.0 is used to implement a load balancing mechanism in the Fog environment.

K. Dasgupta et al. [94] had given an algorithm for load balancing utilized to spot the globally optimal solution in complex or vast search space. This algorithm follows a manner that first initializes and then evaluates the fitness values. It gives an optimal solution to found out the chromosome with the lowest fitness twice and eliminates it with the chromosome of highest fitness. The primary function of this algorithm is to reduce the cost function.

Meftah, Ali et al. [95] modeled the behavior of "Facebook applications" and studied service broker policy and algorithms of load balancing to check the performance of datacentres and large internet applications. They measured the performance of Facebook

applications based on a few parameters. They applied service broker policies, i.e., closest datacentre and three algorithms(throttled, round-robin, and equally spread current execution) used for load balancing. Service broker policy considers the configuration of the datacentre and load balancing as prime factors. It helps route the traffic between datacentres and users with the various policies, i.e., minimal response time and nearby placement of data-centers policy.

Mohanty et al. [91] and Naqvi, Syed Aon Ali et al. [92] provided Meta-Heuristic approach-based load balancing. They used the PSO algorithm for equal distribution of tasks and to enhance resource utilization. CloudSim is used as a simulation environment. Mao, Yingchi et al. [96] proposed a load balancing approach based upon a load prediction model named Adaptive Load Balancing Algorithm(ALBA). The new resources are added if the workload is more than the maximum threshold, and if the load is less than the minimum threshold, then resources are removed from the cluster. The main motive is to balance the workload among all resources and to reduce the response time. The authors also introduced a load prediction model to increase load prediction accuracy. CloudSim is used as a simulation model to show the experimental results.

Few more papers have been studied from the current year. Various researchers are working in load balancing in fog computing, and they have provided many load balancing techniques. Some of the techniques have been discussed below:

Beraldi, R. *et al.* [97] proposed two load balancing approaches to resolving resource management problems in load balancing, i.e., adaptive and sequential forwarding algorithms. To evaluate their proposed algorithms, the authors considered a real-time scenario of a smart city. They combined theoretical models with a simulation approach. They tried to reduce the response time by 19%, and the loss rate has been reduced to 0.2%. To obtain simulation results, Omnet++ has been used.

Beraldi, R *et al.* [98] probe-based load balancing algorithm to resolve problem of load distribution in fog nodes. They mainly focus on selecting fog nodes to allocate incoming jobs. The proposed approach is based on mathematical and simulation models.

Rehman, A.U. [99] proposed a dynamic load balancing strategy based on energy-efficient resource allocation. The authors mainly focus on providing energy-aware

solutions of load balancing for fog and edge devices. They have used CloudSim to evaluate simulation results. The authors considered energy efficiency and cost parameters to assess the proposed approach, and energy has been reduced to 8.67%, and cost to 16.77% compared to the DRAM approach.

Singh, S.P. [100] studied various existing approaches of load balancing and compared them based on different parameters. The authors provided a taxonomy of different load balancers by comparing them and provided their applications also. The authors also focused on providing an energy-efficient load balancing approach.

Singh, S.P. [101] proposed a fuzzy-based load balancing algorithm. Authors have developed a fuzzy-based load balancer with different design levels of fuzzy control. The proposed load balancer has four layers, i.e., data center layer, core layer, fog access layer, and fog device layer. They mainly focus on traffic splitting in fog networks.

Table 2.5: Hybrid load balancing techniques

Author	Algorithm proposed	Experimental Parameters	Approach used	Advantages	Disadvantages
N. Song <i>et al.</i> [67]	Cloud atomization technology	Running time, Node migration	Graph re-partitioning	<ul style="list-style-type: none"> • Improved system performance. • Equilibrium time and cost improved. 	Resource utilization can be improved.
H. Menon <i>et al.</i> [66]	Graph re-partitioning and mapping based technique.	Execution time, object migration cost	Automatic load balancing as well as adaptive run time.	<ul style="list-style-type: none"> • Improved performance. • Increased execution time. 	Implementation cost not considered.

continued on next page

continued from previous page

Author	Algorithm proposed	Experimental Parameters	Approach used	Advantages	Disadvantages
S. Verma <i>et al.</i> [93]	Data replication technique	Response time, execution time	Maximal uniform distribution	<ul style="list-style-type: none"> • Reduced cloud dependency of users. • Improved response time Less reliability.	Less security.
K. Dasgupta <i>et al.</i> [94]	Genetic Algorithm (GA) based load balancing	Response time, cost	Natural selection and genetics.	Minimized make span.	Less efficiency
Meftah, Ali <i>et al.</i> [95]	Throttled, Round Robin, and Equally spread current execution	Response time, request processing time, operational cost	Service broker policy	<ul style="list-style-type: none"> • Improved performance of “facebook applications”. • System response time improved. 	Low security.
Mohanty <i>et al.</i> [91]	Meta-Heuristic approach-based load balancing	Makespan, Economic cost, resource utilization, waiting time, and turnaround time	Particle Swarn Optimization(PSO)	Increased resource utilization.	Less scalability

This section thoroughly studied different load balancing techniques and discussed their advantages and disadvantages. The following table 2.6 compares existing load balancing techniques in detail based upon the approaches used.

Table 2.6: Detailed approaches used in existing load balancing algorithms

Author,Journal(impact factor)	Algorithm	Approach	Simulator	Application
V. Velde <i>et al.</i> [41], IEEE conference	Round Robin(RR)	Designed load balancing algorithm basis of fuzzy technique.	Fuzzy Inference System (FIS)	Cloud systems, Information centers.
M. Verma [54], International Journal of Information Technology and Computer Science (0.765)	Real Time Efficient (RTES)	Proposed fog computing architecture providing scheduling policy for load balancing.	CloudSim	Real time streaming applications, IoTs, Sensor networks.
Q. Fan <i>et al.</i> [49],IEEE Transactions on Network Science and Engineering(3.894)	LoAd Balancing (LAB) scheme	Load balancing algorithm was proposed to reduce the latency of data flows of IoT devices	Base Stations(BS) are deployed for simulation results	Base stations, mobile cellular core

continued on next page

continued from previous page

Author,Journal(impact factor)	Algorithm	Approach	Simulator	Application
X. Xu <i>et al.</i> [16], Wireless Communications and Mobile Computing(2.336)	Dynamic Resource Allocation Method(DRAM)	Distinct forms of computing nodes have been given to investigated, for fog computing and load balance.	CloudSim	IoT applications
D. Puthal <i>et al.</i> [29],Journal of Parallel and Distributed Computing(3.734)	Adaptive EDC authentication technique	Intended peculiar load balancing way outs to verify the Edge data centres (EDCs) and detect fewer loaded EDCs for task allocation.	Scyther simulation environment	Edge data centres (EDC)
N. Téllez <i>et al.</i> [28], International Journal on Artificial Intelligence Tools(0.778)	Tabu Search Method	Proposed optimal load balancing method by using Integer linear Programming(ILP),otimal task allocation and to reduce memory consumption and cost of computation.	Synthetic task scenarios are used for experimental results, embedded devices are also used.	Smart grids, smart traffic

continued on next page

continued from previous page

Author,Journal(impact factor)	Algorithm	Approach	Simulator	Application
M. Zahid <i>et.al.</i> [102], Advances on P2P, Parallel, Grid, Cloud and Internet Computing	Hill Climbing Load Balancing	A three layered framework is provided to resolve electricity problems of the users. Service broken policy is used to implement the proposed algorithm to reduce the response time and processing time.	Cloud Analyst tool	Smart buildings, smart meters, smart grids
S. H. Abbasi <i>et al.</i> [73], International Conference on Broadband and Wireless Computing, Communication and Applications	Active VM Load Balancing Algorithm (AVMLB)	Proposed four layer based fog computing model to manage resources of smart grid.	Cloud Analyst	Smart Grid(SG) and Micro Grid(MG)
Y. Yu <i>et al.</i> [72], Proceedings of the Workshop on Mobile Edge Communications	Scalable and Dynamic Load Balancer (SDLB)	Mobile Edge Computing (MEC) has been proposed to move computing and storage services from cloud to edge.	POG data structure.	Mobile Edge computing(MEC)

continued on next page

continued from previous page

Author,Journal(impact factor)	Algorithm	Approach	Simulator	Application
L.D. Dhinesh Babu [76], Applied Soft Computing,6.725	Honey Bee Behavior based Load Balancing (HBB-LB)	Proposed load balancing algorithm to thoroughly balanced load in virtual machine.	CloudSim	Robotics
N. Song <i>et al.</i> [67], China Communications(2.688)	cloud automation technology	Presented a framework to transform the tangible nodes of fog computing into virtual nodes	Cloud automation system	Delay sensitive application, graph theory.
Mao, Yingchi <i>et.al.</i> [96], Proceedings of the Second International Conference on Innovative Computing and Cloud Computing	Adaptive Load Balancing Algorithm(ALBA).	Load prediction model (ALBA) based load balancing approach has been proposed, that focused upon optimal resource utilization and to reduce response time.	CloudSim	Web applications

continued on next page

continued from previous page

Author,Journal(impact factor)	Algorithm	Approach	Simulator	Application
S. Verma <i>et al.</i> [93], IEEE Conference, and A. Nahir [59], IEEE Transactions on Parallel and Distributed Systems(2.68)	Data replication technique	Proposed an effective load balancing algorithm for a fog-cloud architecture	CloudSim 3.0	Applied on heterogeneous platforms
M. J. Ali <i>et al.</i> [74], International Conference on Intelligent Networking and Collaborative Systems	State-Based Load Balancing (SBLB)	A four layer cloud-fog based architecture is proposed for effective allocation of tasks. Different load balancing algorithms are implemented, their results are compared.	Cloud Analyst tool	Tried to reduce electricity consumption
A. Chawla <i>et al.</i> [103], Big Data Analytics	Package-based load balancing algorithm	Load balancing is done by performing virtual machine replication and grouping the packages.	CloudSim	For proper utilization of resources, and to reduce the execution time.

continued on next page

continued from previous page

Author,Journal(impact factor)	Algorithm	Approach	Simulator	Application
N. Kumar [104], Information and Communication Technology for Sustainable Development	Fuzzy row penalty method	Fuzzy based load balancing technique is followed to reduce the uncertain time required to response in fuzzy cloud environment	CloudSim	Fuzzy cloud environment
J. Wan <i>et al.</i> [105], IEEE Transactions on Industrial Informatics(10.215)	Energy aware Load Balancing and Scheduling(ELBS)	Mainly focus to check the effects of service broker policy, and load balancing algorithms on data centres, and large-scale applications	Robots were setup which were attached with raspberry pie board	Packaging of candies in a Smart factory
Meftah, Ali <i>et al.</i> [95],International journal of advanced computer science and applications(1.3)	Service Broker Policies Based load balancing	Workload on the fog nodes is measured by establishing energy consumption model, and then an optimization function for load balancing was provided.	Cloud Analyst	Facebook applications

continued on next page

continued from previous page

Author,Journal(impact factor)	Algorithm	Approach	Simulator	Application
Naqvi, Syed Aon Ali <i>et al.</i> [92], International Journal of Knowledge-Based Organizations(0.498), and Mohanty <i>et.al.</i> [91]	Meta-Heuristic approach based load balancing	Particle Swarn Optimization(PSO) algorithm have been implemented for equal distribution of tasks, and to increase the resource utilization.	CloudSim	Web applications
Talaat, F.M. <i>et al.</i> [106], Journal of Ambient Intelligence and Humanized Computing(7.14)	Load balancing and optimization strategy (LBOS)	Modified weighted round robin technique has been proposed which is named as AWRR(Adaptive WRR). Proposed technique is applied in fog computing environment in order to enhance quality of service.	MATLAB	E-healthcare application

continued on next page

continued from previous page

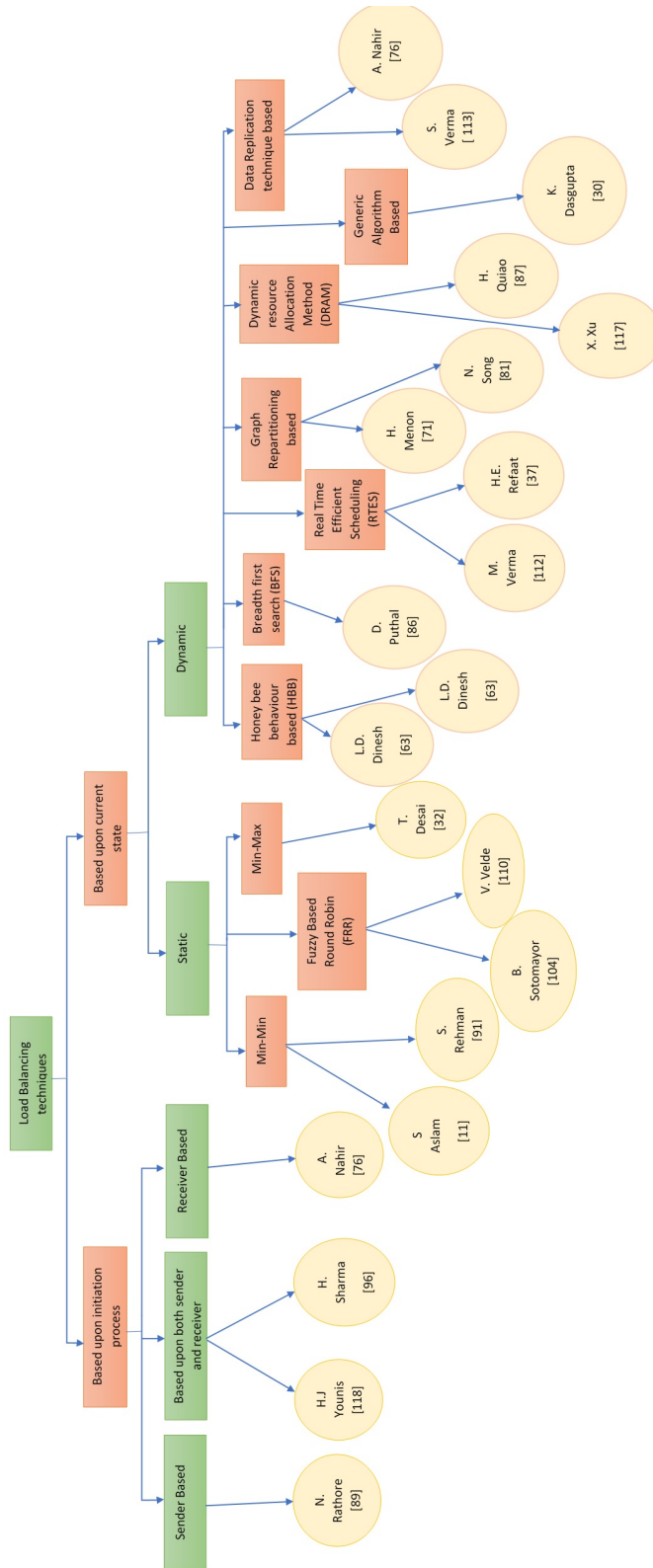
Author,Journal(impact factor)	Algorithm	Approach	Simulator	Application
Beraldi, R. <i>et al.</i> [97], Pervasive and Mobile Computing(3.453)	Two algorithms have been proposed i.e. Sequential and Adaptive Forwarding	The proposed algorithms aims to anticipate load balancing in the fog environment. They have used self tuning adaptation technique to enhance system performance.	Omnet++ framework	E-healthcare application
Shahid, M.H. <i>et al.</i> [107],Computer Communications(3.167)	Load balancing and content-filtration algorithms have been proposed	caching mechanism has been proposed to ensure data delivery in fog environment. Proposed two different algorithms to implement load balancing which helps to reduce energy consumption in fog computing.	Python for programming, and caching simulation environment is used to obtain results	Notice board case study has been considered

continued on next page

continued from previous page

Author,Journal(impact factor)	Algorithm	Approach	Simulator	Application
Bhatia, M. <i>et al.</i> [108], Computing(2.495)	Load scheduling algorithm (QCI) has been proposed that is inspired from quantum computing	A quantumized scheduling approach has been proposed for fog applications. A node based computing index is defined for computational capacity estimation.	iFogSim	Wireless applications were considered that were located at end users

This section has discussed various load balancing algorithms, which mainly depend upon their initiation process and current state. These approaches are applied to fog, cloud layers for load balancing, and better utilization of resources. The studied techniques are grouped into two different categories, and taxonomy is presented in detail. Figure 2.3 provides the taxonomy of load balancing techniques and provides the current research work in load balancing.



50
Figure 2.3: Taxonomy of existing load balancing techniques

2.2.1 Literature review

Many researchers have explored the area of fog computing. This section covers the review of different research works conducted in scheduling and load balancing in cloud and fog computing. A literature review of some research work done in fog computing, managing load balancing, and task scheduling in workflows has been conducted. Many types of research work provided different techniques for scheduling the workflows, but load balancing still needs more exploration. Load balancing techniques need to be provided for proper resource utilization in fog computing while processing scientific workflows. This section represents various approaches proposed by different researchers for load balancing in fog computing. Related work has been divided into three parts, i.e., cost-based, resource-utilization-based, and energy-aware load balancing approaches. Following is the review of some recent research works in fog computing:

2.2.2 Cost-based load balancing

Xie Y et al. [109] formalized scheduling problems in business workflow applications and proposed a novel PSO-based approach named DNCPSO to reduce the cost and makespan. The proposed algorithm is an improved PSO form that applies a directional search process to select data and mutation operations. In their experimental work, authors executed different workflows by considering various factors, cost, and time and compared them with the existing approaches to prove them better. Li, Chunlin, et al. [110] proposed a workflow scheduling algorithm for cloud resources based upon load balancing. They offered a model for workflow scheduling in the distributed cloud environment. This system model helps to reduce the system's response time while executing workflows. The authors proposed a workflow scheduling algorithm based upon the shortest path technique to reduce the execution time of tasks and energy consumption in cloud data centers. They developed social media applications and considered live video applications of workflows to implement their proposed scenario.

Rizvi et al. [111] proposed a workflow scheduling policy to reduce the computational cost and execution time; they named it a fair budget scheduling algorithm. They implemented different scientific workflows and compared their results with their pro-

posed technique. They verified the provided results through the ANOVA test to prove the effectiveness of their approach. De Maio V et al. [112] proposed a multi-objective workflow offloading approach called MOWO for task distribution in the fog environment. Their proposed approach's main objective is to reduce execution time, enhance reliability, and reduce financial costs. The authors considered real-world workflows for execution and obtained results, i.e., meteorological, biomedical, and astronomy workflows. The proposed approach is compared to the existing approach HEFT and reduced response time to 30%.

Tellez, N. et al. [28] proposed a tabu search algorithm to implement load balancing in a fog-cloud environment. They have considered two cost functions, one for computational cost in fog and the other for the cloud. The proposed approach tabu search ensures online calculations in the fog layer to ensure increased processing of tasks at the fog layer itself. Beraldi, R., et al. [97] proposed two load balancing algorithms to solve problems of resource management. The authors used fog node populations different in number, configuration, and processing power to evaluate their proposed approach. They combined both simulation and mathematical model-based approaches to verify their proposed approach's performance—the proposed approach reduced response time to 19% and a loss rate to 0.2%.

Ding, R. et al. [113] proposed a PSO-based approach for scientific workflow scheduling. They used the fitness function to keep track of the execution of workflows under deadlines. They developed an e-healthcare app for monitoring heart rate and tried to reduce the execution cost of workflows. They provided different workflow models for fog computing. Rehman, A. et al. [114] proposed an algorithm for workflow scheduling and named it a multi-objective genetic algorithm (MOGA). They mainly focus on reducing the makespan and on reducing energy consumption in cloud resources. They considered a few parameters to ensure proper resource utilization, i.e., budget, makespan, energy consumption, cost.

Xie, Y et al. [109] proposed a PSO-based workflow scheduling algorithm to reduce cost and make-span in the cloud-edge environment. WorkflowSim simulation is used to provide simulation results. They included Sipht, Epigenomics, Montage, CyberShake, and Inspiral workflows for experimental analysis. Zhou, X. et al. [115] proposed a fuzzy-based workflow scheduling algorithm for the cloud to reduce make-span and cost

optimization. They proposed different workflow models for the cloud environment. They implemented real-time workflow simulations on jMetal simulator. Serhani, M.A. et al. [116] proposed architecture for the cloud to execute IoT workflows. The proposed architecture keeps track of the current VM load and its capacity, state of applications, and if any recovery action is required, it acts accordingly. The QoS parameters, i.e., cost and time, Scalability, are considered to enhance proper resource utilization. Docker Swarm Cluster, along with PostgreSQL used for simulation results.

Bittencourt, L.F. et al. [117] proposed an HCOC scheduling approach for the cloud environment to reduce cost and execution time of tasks in cloud data centers. HCOC tried to execute all workflow tasks in private cloud resources based on pay per use. Xu, R. et al. [118] proposed an improved PSO-based scheduling algorithm for workflow application to reduce the execution time of workflows than existing PSO algorithms that will help to reduce the execution cost of workflows. They used MATLAB for experimental results.

Naik, K.J. et al. [119] proposed a scheduling model to balance the load in fog computing-based connected car applications. The tasks are scheduled at the server level rather than the device level. The proposed load optimization model examples are used to reduce runtime and deadline. iFogSim is used to show the results. Puthal, D. et al. [120] proposed a secure load balancing approach for edge data centers. The cloud data centers help authenticate the end-users to build a secure connection between end-users and edge computing devices. They proposed a secure and sustainable approach for load balancing in edge data centers.

2.2.3 Resource-utilization based load balancing approaches

Javadzadeh, Ghazaleh et al. [121] provided a systematic review of existing literature by studying existing approaches in fog computing. According to the authors, fog computing is the best solution for the limitations of cloud computing. Singh, Simar Preet [101] proposed a fuzzy-based load balancer to reduce the resource wastage in fog computing. They also provided a fuzzy-based three-tier model for the load balancer

based on the software-defined distribution of tasks. They have used both theoretical as well as empirical experiments. They tried to improve resource utilization and reduce costs.

Ding, Ruimiao, et al. [113] proposed a scheduling approach based upon Particle Swarm Optimization (PSO) and Min-Min strategy. They define different workflow models, i.e., time and cost models, based on resource cost and execution time in a fog computing environment. The fitness function is used to calculate the execution cost of workflow implementation. For the simulation results, java JDK 1.7 has been used. The work's main focus is to reduce the implementation cost by finishing the tasks before their deadline. Elsherbiny, Shaymaa, et al. [122] proposed Intelligent Water Drop (IWD) based algorithm for workflows scheduling in the cloud environment. They applied different workflows to the Workflow simulator, i.e., cybershake, sipht, epigenomics, to compare their make-span with the proposed algorithm. De Falco, et al. [79] proposed extremal optimization (EO) based load balancing approach. In the proposed EO approach, during the load balancing process, task migrations are done. The authors considered few factors in evaluating EO's performance, i.e., fitness function and target nodes. The authors compared their proposed approach to their previously proposed approach, i.e., sequential-extremal optimization algorithms.

Liao, S. et al. [123] developed a framework for optimizing training task distribution by considering communication cost and physical computing to reduce data transfer error at every device. They considered machine learning for their experimental evaluation and proved that their proposed network-aware approach reduces model training cost and enhances accuracy. The authors implement their experiments on synthetic and real-world data to confirm network resource utilization improvement by their proposed algorithm. M. Kaur et al. [124] developed fog computing architecture for load balancing in scientific workflow application that is named as FOCALB. The paper also proposed a hybrid load balancing approach i.e. Tabu-GWO-ACO. The proposed approach aimed at maximum resource utilization and reduce implementation cost and energy consumption in fog layer. The results have been obtained using iFogSim toolkit and the obtained results are compared with other existing approaches on the basis of considered parameters i.e. cost, energy, time.

Biswas, T et al. [125] proposed PSO based algorithm for scheduling workflows considering different parameters like resource utilization, makespan, load balancing, and speedup ratio to design the fitness function. They considered real-time workflows for simulation results, i.e., LIGO, SIPHT, Epigenomics, and Montage. The proposed algorithm tried to improve the speedup ratio and make-span. They also applied the ANOVA test to prove their provided result to be better than other existing approaches. Aron, R [126] proposed a resource provisioning model to analyze scientific workflows. The author aims to incorporate an efficient resource provisioning strategy to improve VM performance in the cloud environment. They proposed a hyper heuristic-based scheduling algorithm to decrease the makespan and cost also.

Singh, S.P. et al. [101] designed a fuzzy-based load balancer to maintain the payload and load balancing between the fog and cloud environments. The authors proposed an energy-efficient fuzzy-based three-level design for load balancing in fog networks. They also proposed a fuzzy-based load balancing approach to manage traffic between cloud and fog. C.-F. Lai et al. [127] proposed NAT based load balancing approach for resources in fog computing. The proposed load balancing approach is ICE based, and used three layer fog architecture containing TURN servers at the edge of network that are monitored by fog nodes controllers. The authors considered two parameters for analysing i.e. maximum load, and controller latency. It has been obtained that maximum load has been applied to the nodes and observed that the proposed method distributes cloud load and avoid transfer server load to the users. L. F. Bittencourt et al. [128] have proposed a health monitoring system for minimizing network usage and latency, and proposed model is based on fog architecture. Along with this, a load balancing approach has been proposed. The authors used iFogSim toolkit for validation of the effectiveness of their proposed approach.

2.2.4 Energy-aware load balancing approaches

Shahid, Muzammil Hussain, et al. [107] proposed load balancing and content filtering based energy-aware mechanisms. In their proposed approach, they applied load distribution among fog nodes in a random manner. Then content filtration is done on active nodes. The proposed load balancing algorithm helps to increase the efficiency

of the system. Kaur, Mandeep et al. [55] proposed a load balancing technique based on equal workload distribution. The provided approach is implemented using a cloud analyst tool, and results are compared with round-robin and throttled load balancing techniques. The chapter tries to reduce the implementation cost in the fog environment and improve resource utilization. D. Baldo et al. [129] has proposed a LoRaWAN architecture for Smart waste management. The proposed system is deployed at the edge of the network. Smart bins are designed that are deployed in the city containing capability of their own data collection. Video surveillance units are deployed to monitor the bins. The smart bins contains photo Voltaic panels for energy saving.

Saroa, Mandeep Kaur et al. [130] proposed architecture for innovative application in fog computing, i.e., intelligent waste management systems. The authors also discussed different fog computing applications. They also studied load balancing in a fog environment and compared the existing techniques. Wadhwa, Heena et al. [131] proposed additional resource provisioning and scheduling techniques in a fog environment. The authors studied fog computing-related to other models and proposed architecture of fog computing for e-healthcare. It also provides the challenges faced by IoT and their solutions through fog computing. Shahid, M.H. et al. [107] proposed an energy-aware mechanism in which they applied load balancing and content filtering on fog networks. Each active node is analyzed based on its energy level and several current neighbors. After analyzing nodes, the necessary content is cached in these nodes by applying the filtration process. The authors also proposed a load balancing algorithm to enhance system performance. The authors considered a notice board case study for the experimental results. The Fog-based caching simulation environment is used for simulation results, and Python is used for programming purposes.

Choudhary, Anita, et al. [132] has proposed a VM placement-based energy-aware load balancing algorithm. Their proposed approach uses a task clustering approach to reduce energy consumption in cloud data centers. The proposed approach is based on a min-min algorithm and combines smaller tasks into large tasks to reduce virtual machines' burden. Kaur M et al. [133] proposed an energy-aware load balancing technique for the fog computing environment. They considered scientific workflow applications to execute in fog computing using iFogSim. The proposed approach tries to enhance resource utilization by reducing latency and reduce energy consumption in fog nodes. R.

K. Naha [134] proposed an energy-aware resource allocation method based on multiple linear regression. The proposed approach tries to reduce failures that occurred in fog computing because of energy constraints. Along with this, an energy-aware framework has been proposed to execute different applications in fog. The proposed approach has been compared with other existing approaches, reducing execution and processing time.

Al-khafajiy et.al. [135] proposed COMMITMENT, a trust management-based approach for fog computing to secure fog nodes. To monitor fog resources, they proposed a load balancing algorithm that helps to monitor fog resources' performance and reduces congestion on fog with offloading. They considered CPU consumption, fog security while applying load balancing. De Maio et al. [112] proposed a multi-objective task offloading approach in fog computing for workflows. They considered biomedical, astronomy, and meteorological workflow examples to evaluate their proposed approach. They claimed that their proposed MOWO approach reduced the response time to 30% in small tasks. They modeled workflow problems in the fog-cloud environment while considering three different parameters cost, response time, and reliability.

Naqvi, S.A.A et.al. [92] proposed ACO based load balancing in fog-cloud computing. They also proposed three layer-based fog-cloud architecture for smart grids. They tried to prove ACO results to be better than a round-robin and throttled. They provided simulation results using NetBeans. They tried to reduce processing time and energy consumption by fog nodes. Talaat, F.M. et al. [106] proposed a load balancing algorithm (LBOS) for fog computing and implemented it in healthcare applications. They proposed a three-layer architecture for Fog-IoT. This algorithm aims to reduce response time and allocation cost. The proposed algorithm helps monitor network traffic, and it equally distributes the incoming user requests between the available servers. The authors ensure load balancing (85.71%) and resource utilization through their proposed approach in a fog environment.

Al-khafajiy et.al. [135] proposed COMMITMENT, a trust management-based approach for fog computing to secure fog nodes. To monitor fog resources, they proposed a load balancing algorithm that helps to monitor fog resources' performance and reduces congestion on fog with offloading. They considered CPU consumption, fog security while applying load balancing. De Maio et al. [112] proposed a multi-objective task

offloading approach in fog computing for workflows. They considered biomedical, astronomy, and meteorological workflow examples to evaluate their proposed approach. They claimed that their proposed MOWO approach reduced the response time to 30% in small tasks. They modeled workflow problems in the fog-cloud environment while considering three different parameters cost, response time, and reliability.

Naqvi, S.A.A et.al. [92] proposed ACO based load balancing in fog-cloud computing. They also proposed three layer-based fog-cloud architecture for smart grids. They tried to prove ACO results to be better than a round-robin and throttled. They provided simulation results using NetBeans. They tried to reduce processing time and energy consumption by fog nodes. Talaat, F.M. et al. [106] proposed a load balancing algorithm (LBOS) for fog computing and implemented it in healthcare applications. They proposed a three-layer architecture for Fog-IoT. This algorithm aims to reduce response time and allocation cost. The proposed algorithm helps monitor network traffic, and it equally distributes the incoming user requests between the available servers. The authors ensure load balancing (85.71%) and resource utilization through their proposed approach in a fog environment.

The following table 2.7 provides a review of existing load balancing and scheduling techniques in workflows. Existing approaches have been compared based on their execution environment, objectives, performance metrics considered for evaluation, and the experiment's application.

Table 2.7: Comparison of load balancing techniques considered in related work

Year	Author	Environment	Objective	Performance metrics	Applications
------	--------	-------------	-----------	---------------------	--------------

continued on next page

continued from previous page

Year	Author	Environment	Objective	Performance metrics	Applications
2019	Ding, Ruimiao, et al. [113]	Java with JDK 1.7	To provide cost-effective scheduling policy for multi-workflow	Execution cost and time	Heart rate monitoring
2019	Li, Chunlin, et al. [110]	WorkflowSim simulator	To provide workflow scheduling based on load balancing to utilize the resources of cloud efficiently	Execution Time, System Performance, Cost	Real live video
2020	Rizvi et al. [111]	CloudSim simulator	To schedule the task fairly and minimized the makespan to satisfy finance constraints	Computation cost and makespan	Amazon's EC2

continued on next page

continued from previous page

Year	Author	Environment	Objective	Performance metrics	Applications
2017	Choudhary, Anita, et al. [132]	WorkflowSim simulator	To provide an energy-aware load balancing technique for workflows in the cloud environment.	Energy consumption	Scientific applications
2017	Elsherbiny, Shaymaa, et al. [122]	WorkflowSim simulator	To develop the workflow scheduling algorithm for meta-heuristics	System performance and scheduling cost.	Common workflows, i.e. sipht, cyber share etc.
2020	Serhani, M. Adel, et al. [116]	Self-adapting cloud service orchestration	Proposed an architecture for end to end workflow management support	CPU utilization, storage	IoT workflows and e-health monitoring
2020	De Maio V et al. [112]	Multi-objective workflow offloading (MOWO)	To propose an efficient workflow offloading approach for fog computing	Response time, financial cost, and reliability	Real-world workflows

continued on next page

continued from previous page

Year	Author	Environment	Objective	Performance metrics	Applications
2020	Ying xie, et al. [109]	Docker Swarm Cluster along with PostgreSQL	To propose an architecture to support workflow management	QoS parameters i.e. cost and time, Scalability	Health monitoring
2021	Ijaz, S, et al. [136]	MATLAB	To propose a novel energy-aware workflow scheduling model to optimize makespan and energy-consumption in fog environment.	Makespan, and energy-consumption	Real world workflow applications
2021	D. Baldo et al. [129]	LoRaWAN gateways, Raspberry Pi, Mysql database	To propose LoRaWAN architecture for smart waste management system	Filling levels of bins, Absolute and relative errors	Smart waste management system

Many experiments have been conducted to offset the cloud storage burden as the load on the cloud grows exponentially, as discussed above. Since fog networks are hetero-

geneous and complex, they cannot use most cloud load balancing techniques directly. Some load balancing tasks are throughput maximization, response time minimization, and traffic management. Other load balancing techniques include server-side resource utilization management, request processing time reduction, and distributed environment scalability enhancement.

In today's fog computing environments, users need applications that react quickly anytime, and they try to access something and function quickly. Since load balancing is regarded as a fundamental problem, a practical load balancing approach can dramatically boost QoS factors in a fog network. According to Kashani et al. [137], objective function-based optimization in load balancing decision making should be extended to improve and optimize load balancing. In this analysis, Tabu scan, GWO, and ACO are combined to produce a converged objective function.

2.2.5 Year wise review of load balancing techniques

This section covers load balancing techniques, including QoS parameters and Focus of Study (FoS) in fog computing and cloud computing. Figure 2.4 describes the load balancing evolution of fog computing across many years.

In the year 2013, K. Dasgupta et al. [94] proposed a Genetic Algorithm (GA) based load balancing technique. GA mainly focuses on balancing workload among all cloud resources, which helps to reduce the makespan of tasks. A cloud analyst tool has been used to show the simulation results. The authors compared the proposed algorithms with the hill-climbing, round-robin, and first-come-first-serve algorithms and ensured their algorithm outperforms existing ones. Resource utilization increased by equal distribution of workload. L. D. Dhinesh Babu [78] proposed a honey bee behavior inspired load balancing algorithm which provides a balanced load among all virtual machines that improves throughput.

In the year 2014, H. Menon et al. [66] provided graph partitioning-based load balancing methods. To reduce service migration costs, the authors provided repartitioning methods. SCOTCH and CHARM++ have been used to implement the proposed algorithms. A greedy algorithm has been used to balance the workload. Different load balancers have been compared based on their performance. Singh, G.S. et.al. [138] HACOBE

load balancing algorithm for dynamic environment. They used ACO and Artificial Bee Colony techniques for equal distribution of load in all nodes. HACOBEES has been compared with the existing techniques, and it performs better than other techniques and reduces response time. Cloud analyst tool has been used for simulation results. In the year 2015, Ouies et al. [139] improves the quality of experience of users, as with the increase in traffic demands of users are increased. The authors mainly focused on load balancing. User requests are processed at fog layer cluster resources. Small cells are generated having less complexity, and a resource management-based algorithm has been proposed. In the year 2016, N. Song et al. [67] proposed a graph repartitioning-based load balancing technique that helps to improve network flexibility and resource utilization. Cloud automation technology is used to show the simulation results. Four layered fog computing architecture has been provided, i.e., Physical layer, cloud resource layer, service, and platform management layers. S. Verma et al. [93] proposed a data replication technique-based load balancing algorithm for the fog-cloud environment. The action aims to reduce cloud dependency, as data is processed at the fog layer only. The three-layered architecture proposed, which comprises ground, edge, and core layers. CloudSim 3.0 has been used to show simulation results. In year 2017, Beraldi et al. [140] proposed Cooperative load balancing (CooLoad) for fog computing environment. Requests have been observed; when requests came to a full processing node, they were forwarded to another node. Quality of service has improved with equal load distribution. Y. Yu et al. [72] proposed Scalable and Dynamic Load Balancer (SLDB) for mobile edge computing. Minimal perfect hashing has been used in the SLDB algorithm to enhance performance and reduces memory consumption. The Data Plane (DP) of SLDB has been considered, which fits in the cache and improves processing speed. In the year 2018, Rafique et al. [141] proposed Novel Bio-Inspired Hybrid Algorithm (NBIHA) for load balancing in a fog environment. Average response time and energy consumption have been reduced by applying efficient task scheduling. The proposed system architecture has three layers, i.e., client layer, fog layer having scheduler, and cloud layer containing datacentres. Metaheuristic Particle Swarn Optimization (MPSO) is used for scheduling tasks. iFogSim has been used to provide the simulation results. Arshad et al. [142] evaluate and analyses nature-based algorithms named Pегion In-

spired Optimization(PIO) and Binary Bat Algorithm(BBA). The energy consumption of Cloudlets has been measured by evaluating these algorithms. The three-layer architecture proposed containing smart homes in the first layer, cloudlets at the second layer and, cloud servers at the third layer. Smart meters at smart homes are used for bill estimation.

In the year 2019, A. Fahs et al. [143] proposed a routing algorithm for fog environment. This technique aimed to reduce latency and equal load distribution. With the equal distribution of load, sender to receiver service access time is reduced. Proposed proximity aware system based upon Kubernetes. N. Javaid et al. [144] proposed a nature-inspired Cuckoo search load balancing algorithm that combined with levy walk distribution and flower pollination. They optimized the response time and processing time of fog and cloud as well. Cost is also considered an important parameter, as they tried to reduce the cost of data transfer, microgrids, VMs, and the total cost. H.A. Khattak et al. [145] proposed fog-cloud server-based architecture, which aims in proper utilization of all resources. They implemented their proposed utilization-based load balancing approach in e-healthcare. In e-healthcare, data is very critical and can not tolerate fractional delay. They tried to distribute equal load among all servers by shifting load from overloaded servers to less load. They considered different parameters like latency, load balancing, QoS, bandwidth. They used iFogSim to obtain simulation results.

In the year 2020, Talaat, F.M. et al. [106] proposed a resource allocation-based load balancing approach that depends upon reinforcement learning. This approach keeps track of network traffic by measuring server loads that help it to handle incoming requests. This approach distributes the workload among all available resources for their appropriate utilization. A three-layer fog-cloud-based architecture has been proposed for healthcare. This approach helps to reduce response time. M. Kaur et al. [55] proposed a load balancing approach based upon the equal distribution of workload. They proposed three-tier architecture of fog-cloud. To reduce energy consumption, cost, and processing time in the fog-cloud environment is the main motive of this chapter. The results are obtained using a cloud analyst simulation tool by implementing proposed approaches. They also compared their approach with existing round-robin, throttled algorithms. M. Bhatia et al. [108] proposed a quantumized approach of task scheduling

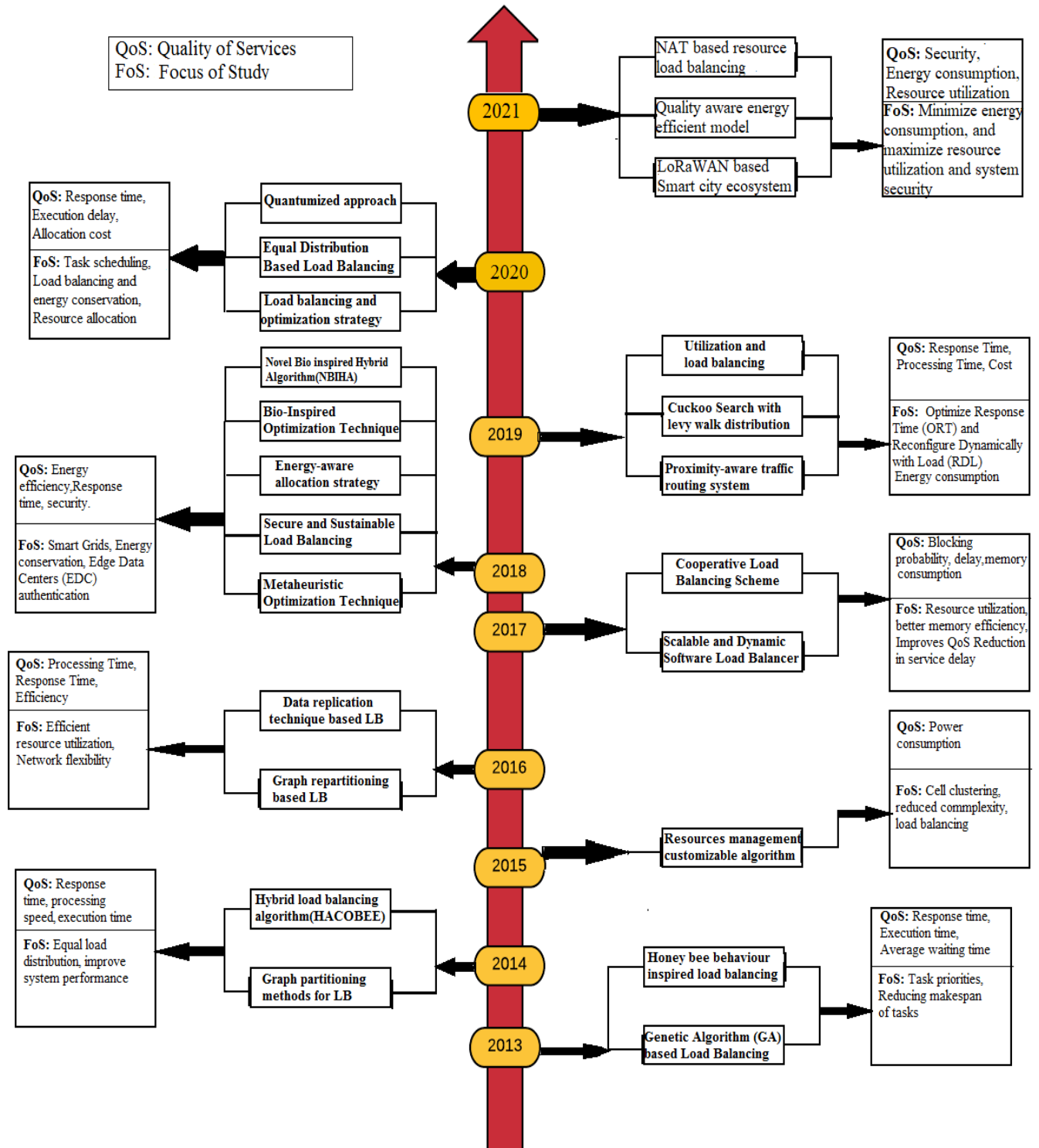


Figure 2.4: Evolution of load balancing

in a fog environment. The proposed approach tends to distribute workload among all fog nodes, so that system performance could be improved and execution delay can be reduced. iFogSim has been used to show simulation results.

2.2.6 Performance measurements that impact load balancing

Load balancing helps to enhance system stability. For the implementation of proper load balancing in a system, a good load balancer is required. N number of VM is required to execute n number of tasks. The workload of a finite number of users is equally divided into an equal number of virtual machines assigned to them. Each virtual machine remains in two states that are active state and ineffective state. The energy consumption of VM in the idle state is 60% of the VM inactive state [146]. Here a few performance measures are discussed which may affect load balancing in the fog computing environment:

- **Energy consumption:** Electric power consumed by the fog nodes can be considered as energy consumption in fog computing. There are various devices in the fog environment, i.e., servers, gateways, routers that consume energy while performing operations [147]. Load balancing is mainly done to reduce the overall energy consumption in the fog nodes.
- **Thrashing:** Thrashing is the state of the system in which all nodes spend their entire time forwarding jobs among themselves without executing these jobs [28]. A system may enter in thrashing state if memory or other resources are consumed by idle nodes only. When there is no idle VMs, the system will keep on finding idle VMs, and tasks are forwarded to the next node in the system. If still, the task does not get the resource for execution, it is again forwarded to the next node. Hence, until the tasks get the resources, all the nodes keep on forwarding them rather than executing. Proper load balancing is required to resist the system from entering into a thrashing state.
- **Reliability:** The stability of the system may be improved if the systems are reliable and it recovers from any failure. During the execution of the tasks, any system failure may occur; then the job is shifted to another machine to continue its execution [40].
- **Predictability:** Predictability is a degree of correct prediction about the state of a

system, i.e., allocating tasks, executing tasks, completing a task. The Predictability can be done based on the previous behavior of the tasks coming from the system. Proper load balancing can enhance the prediction value [40].

- **Fault tolerance:** Fault tolerance of any system is the availability of the system during any failure to continue the regular operation of the system. A load-balancing algorithm can be adopted to improve the fault tolerance of the system under partial failure [148].
- **Execution time:** Execution time of any system is the time from the assignment of any task till the execution completes. The system performs the task and returns the result to the requesting user. A simple load balancing algorithm can be adopted to reduce the execution time of the tasks [149].
- **Cost:** The cost of maintaining the resources is reduced by load balancing in the computing environment, which includes various factors: energy consumption, maintenance cost [150]. If only a few resources are utilized with more load, and others remain underutilized, they also require maintenance. Due to this, the maintenance cost will be high. So load balancing is needed to reduce implementation and maintenance costs.

—

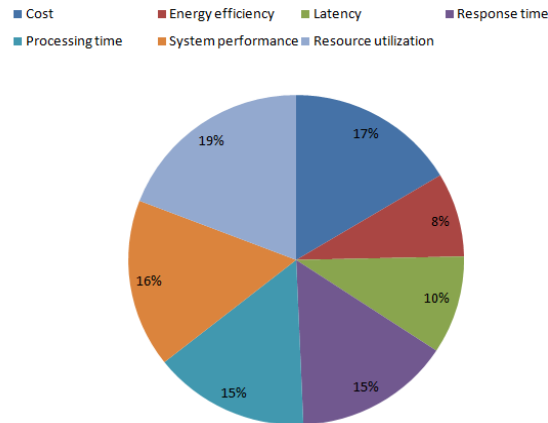


Figure 2.5: Percentage of load balancing metrics considered in reviewed papers

The table 2.8 provide a comparison of existing load balancing algorithms based on considered parameters.

Table 2.8: Comparison of different algorithms on the basis of considered parameters

Author & Year	Cost	Energy efficiency	Latency	Response time	Processing time	System performance	Resource utilization
[43]	×	×	✓	✓	✓	×	×
[150]	✓	✓	✓	×	✓	✓	✓
[28]	✓	×	✓	✓	×	✓	✓
[102]	✓	×	×	✓	✓	×	✓
[151]	✓	✓	✓	✓	✓	✓	✓
[73]	✓	✓	✓	✓	✓	×	✓
[72]	✓	×	×	×	×	✓	✓
[152]	✓	×	✓	×	✓	✓	✓
[93]	✓	×	×	✓	✓	✓	✓
[153]	×	×	×	×	✓	✓	✓
[74]	✓	✓	✓	✓	✓	✓	✓
[103]	✓	×	×	✓	✓	✓	✓
[104]	✓	×	×	✓	×	✓	✓
[105]	×	✓	×	✓	✓	✓	✓
[96]	✓	✓	×	✓	×	✓	✓

2.3 Open issues and research challenges

Fog computing provides a distributed computing environment. Fog computing expands the services of cloud computing to the edge of the network. Fog computing contains various computing devices called fog nodes, i.e., routers, switches, gateways deployed on geographically distributed areas. Fog nodes are connected through networking devices, i.e., WiFi, 3G, 4G VOLTE. Fog computing provides storage services at the network edges to process user requirements. Fog server offers services like data management and has to reduce maintenance costs. Some of the issues that are faced in fog computing are discussed below:

- **Security:** Fog computing has to deal with various kind of cyber-attacks which affects the security of fog nodes. Various researchers provide few authentication techniques, but there is still a need to secure edge data centers by providing the techniques which can help to execute lightweight processes in real-time. Location-aware data should also be secured because if the intruder knows the data location, they may try to temper data and nodes. The security threats should identified, so that fog nodes can be made more secure [29] [18].
- **Deployment of fog servers:** Fog servers need to be placed near to the IoT devices, and users should be aware of the locations of fog nodes. Fog servers need to be deployed in the way that they should provide maximum services. The maintenance costs of the fog servers are reduced if the work done by the fog nodes is continuously analyzed. Deployment of fog servers should be done expertly to utilize resources fully, and simulation results should match with the run-time environment [154] [155].
- **Privacy:** As fog computing supports location awareness, the privacy of fog servers becomes a challenging task. There should be the implementation of laws and privacy policies to provoke data transfer outside the network boundaries. While transferring data from the IoT layer to the fog and upper layers, leakage of sensitive data should be avoided. The IoT users who have to access the fog servers should be authentic and trustworthy so that they should not temper private information [45] [156].
- **Power consumption of nodes:** If the fog nodes are not properly deployed and no load balancing is applied, then few resources will become overloaded, and few will

remain under-loaded. The under-loaded resources will also consume the same amount of power for their work. So power consumption of fog nodes needs to be reduced by proper resource utilization. There is a need to design energy consumption models which can help to reduce the requirement of computation power, and also computational cost [157] [74] [55].

- **Computational delay:** Fog computing support time-sensitive applications which can not endure any computational delay. Real-time applications require an immediate response; whenever any request is received at fog resources, it should be immediately processed [18] [59].
- **Server Availability:** Network manager at the fog layer is responsible for the availability of resources in the fog computing environment. A sustainable and reliable fog architecture needs to design for resource availability. Application failure and security may affect the availability of fog resources.

2.4 Problem formulation

Based on the Literature review following problem has been formulated for this research work:

- While maintaining the workload in a fog-based cloud system, it becomes mandatory to keep the energy consumption of computing nodes at the fog layer.
- An efficient framework is required to reduce the energy consumption of fog-based cloud servers, which will help for the proper utilization of fog nodes.
- Due to overloaded requests from huge end-users, they may face problems while executing their requests.
- There is a need to design an efficient energy-aware load balancing algorithm to manage the workload on the fog nodes and enhance resource utilization in the fog computing environment.

These issues have been considered for further research work.

2.5 Research assumptions

- It has been assumed that in fog environment more time-critical tasks are required to be executed in limited time.
- It has been assumed that in case of more number of user requests in fog environment, there is load imbalance. There may be overloading and underloading of resources.
- In order to manage the workload on the fog nodes and enhance the resource utilization in the fog environment, there is a need to design an efficient energy aware load balancing algorithm.
- The proposed framework assumes that fog nodes can be clustered dynamically in order to process more number of requests.
- The proposed load balancing scheme assumes the presence of a centralized fog computing system that collects all the state information concerning the fog nodes, while our proposed framework is fully distributed.

2.6 Research objectives

The main objectives of research work are:

- To study and analyze existing load balancing techniques in fog computing.
- To design a resource utilization-based load balancing framework for fog computing.
- To design an efficient energy-aware load balancing algorithm for the fog computing environment.
- To implement the proposed algorithm in a fog computing environment and compare its results with the existing techniques.

CHAPTER 3

Proposed framework for load balancing (FOCALB)

The previous chapter provides a detailed description of load balancing in the fog computing environment. The study of related work depicted that only load scheduling has been addressed in fog computing. Still, load balancing has not been given much attention in fog computing for scientific workflows. To provide a solution to load balancing for scientific workflow applications in a fog environment, the fog computing architecture of load balancing for scientific workflow applications has been designed in this chapter.

The proposed framework considers three main parameters while implementing load balancing in a fog environment, i.e., Execution time, Computational cost, Energy consumption. A significant amount of data that needs immediate processing and storage also requires resource utilization in some scenarios. Many researchers focus on improving fog computing performance with critical concerns like scheduling, privacy, security, and system deployment. Fog computing still faces many obstacles in its growing way, i.e., Less storage space, Privacy issues due to location-awareness, Overloading of resources, More energy consumption, resource management.

This research mainly focuses on the overloaded resources issue, which means the resources need load balancing. Load balancing is an open issue that is reducing the performance of the fog computing environment. Load balancing equally distributes the workload among all the fog resources, considering system requirements. Efficient load balancing is required in fog computing to enhance the utilization of resources and to provide high-quality services to the users.

In this chapter we proposed a framework for maximum resource utilization in the fog computing environment and named it FOCALB. The proposed framework has been divided into three layers, i.e., end-users, fog layer, and cloud layer. FOCALB has

been divided into three modules in terms of operating techniques, i.e., pre-processing module, optimization module, and parameter analysis. Further, the chapter provides workflow models that are considered in this research. In addition to this, Hybridized load balancing algorithm for scientific workflows (Tabu-GWO-ACO) has been proposed to evaluate the working of FOCALB.

3.1 FOCALB: Fog computing architecture of load balancing for scientific workflow application

Workflows are used in scientific domains for performing different experiments. A large volume of data is exchanged during the communication process between other resources. Most workflow tasks are executed on local fog nodes in fog computing rather than sending them to the cloud. Nevertheless, with the increase of data exchange between different fog nodes, load balancing optimization becomes necessary so that neither of the fog nodes becomes overloaded with tasks nor remains underloaded. Hence, these resources consume more energy while processing the tasks, resulting in fog nodes' high hardware cost. So load balancing can help improve system performance and reduce workflow task's execution time and energy consumption.

This section proposed a fog computing architecture of load balancing (FOCALB) for scientific workflow applications to reduce the cost, execution time, and execution time energy consumption that will overcome overloaded resources in the scientific workflow application-based fog computing. The proposed FOCALB model is shown in the Figure 3.1.

Figure 3.1 illustrates the three-layered fog architecture for load balancing that includes the end-user layer, fog layer, and cloud layer. The proposed architecture has layers similar to the basic architecture of fog computing but with modifications in the fog layer. These layers are explained as follows:

End-user layer: End users are at the edge of the network that generates requests and submits to the fog layer. In scientific workflow applications, the number of requests leads to millions of tasks per second. These tasks are parsed and then sent to the fog layer for execution. The tasks with a minimum deadline are prioritized over the others.

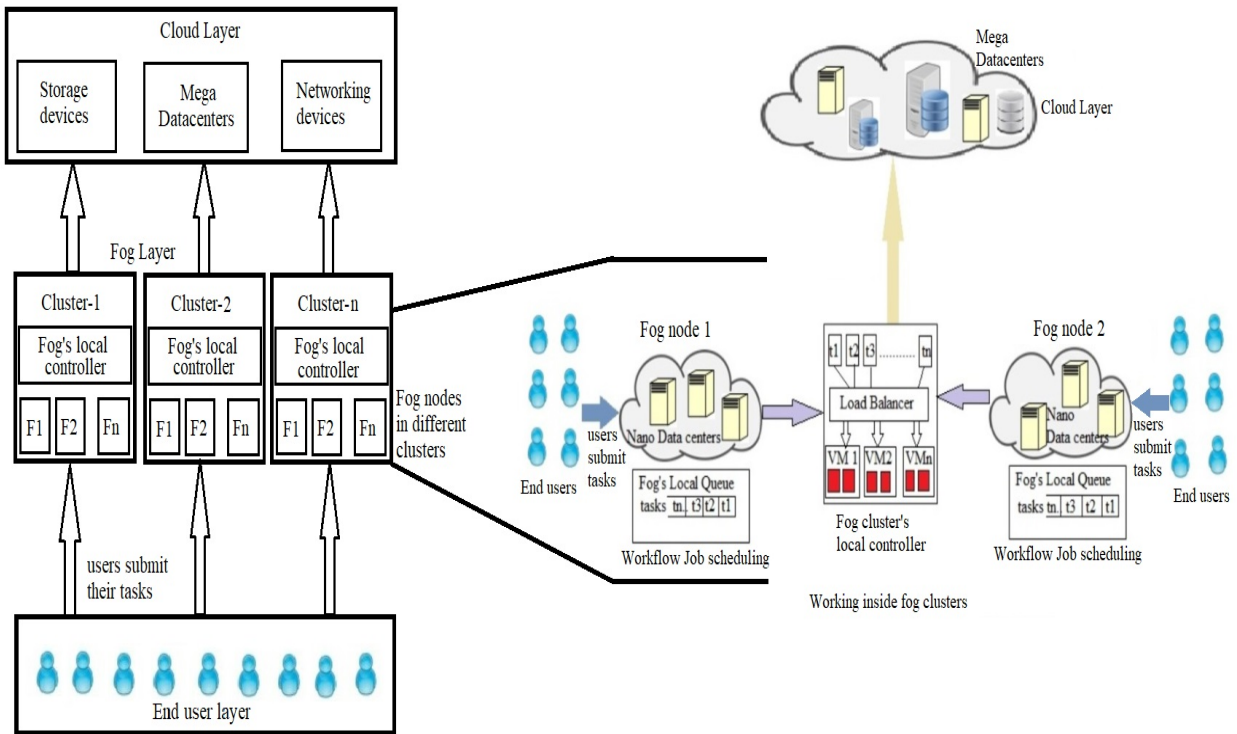


Figure 3.1: FOCALB

These tasks are executed in the fog layer itself, and others are sent to the cloud layer. **Fog layer:** The fog layer containing fog nodes has been divided into different clusters containing a few fog nodes in each. Each cluster has its local controller that keeps track of all fog nodes in the cluster and ensures full utilization of all the resources in the cluster. Users connected to the fog layer continuously transfer requests to fog nodes. There are many users, so the number of tasks generated is also significant in amount. This architecture aims to get networking services closer to the nodes producing data at the lowest awareness layer. Nodes, computers, physical and virtual sensors, vehicles, and other components make up the system. Many of these nodes are handled in compliance with the service's specifications and the node's characteristics. The fog layer contains nano data centers having cloud-like services but in a limited manner. These nano data centers have limited storage and processing capabilities. So, only those tasks that require an immediate response are executed, and others are forwarded to the cloud layer. Fog nodes have their local scheduler assigning tasks to the cluster's

local controller based on their priority. Load balancing in a fog environment allows for an even allocation of workload across infrastructure, intending to continue to deliver services even though a portion of service fails. That is accomplished by provisioning and de-provisioning instances of applications, as well as adequate resource management. Since data centers procure differences between hosts and display unique traffic characteristics, fog computing requires an effective load balancing system to improve device efficiency and network usage.

Cloud layer: The cloud layer is attached to the Fog layer for data transmission and storage in the future. Cloud layers have Large data centers that have ample storage, computing, and networking capacities. This data center provides storage support to the fog nano data centers for their less prioritized tasks for future use and storage.

3.1.1 Operating modules of FOCALB

The proposed model is divided into three modules in terms of operating techniques pre-processing module, optimization module, and parameter analysis module as given in Figure 3.2. The process adopted in each module is given below:

Preliminary processing: Workflows are first taken as input and then parsed into a group of tasks using WFMS (Workflow management system), which allows for automated and smooth workflow execution. It lets users identify and model workflows, set deadlines and budget constraints, and the environments in which they choose to work. The WFMS then evaluates and executes these inputs within the given constraints. The task dispatcher then examines the dependencies and sends the completed assignments to the scheduler.

Optimization module: Using this approach to conduct multiple assignments enables the user to get a more detailed view of the service received. Task scheduling in the cloud has recently become a research subject, and current research demonstrates its relevance and various types of solutions. Tsai and Rodrigues [158] work demonstrated the importance of balancing intensification-based and diversification-based solution search algorithms to produce the best scheduling performance. The number of tasks mapped onto fog nodes is executed if all the nodes get the resources. However, if a few tasks did not get resources and nodes in the fog layer remain underloaded, it needs optimization

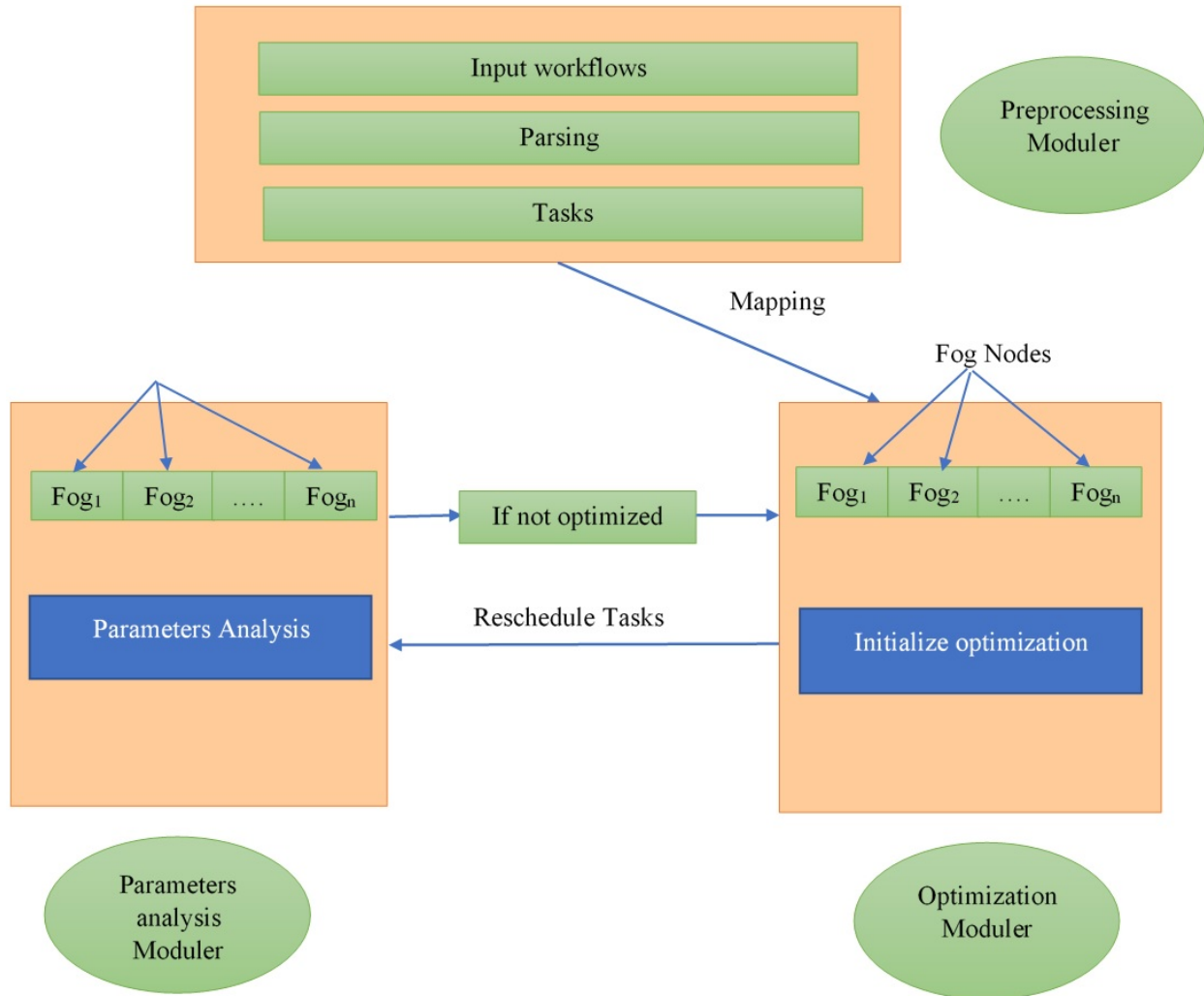


Figure 3.2: Working of FOCALB

of resources. Hence the load balancing technique is required for the optimization of resources. The proposed algorithm Tabu-GWO-ACO is applied for the optimization of resources.

Parameter analysis module: When optimization of resources is done, then considered parameters are analyzed, i.e., energy consumption, cost, and execution time. If, while analyzing parameters, it is found that optimization is still required, tasks are sent back to optimization modular, and tasks' rescheduling is done.

3.1.2 Workflow task assignment

Figure 3.3 shows the task assignment process in fog nodes. The workflow scheduler

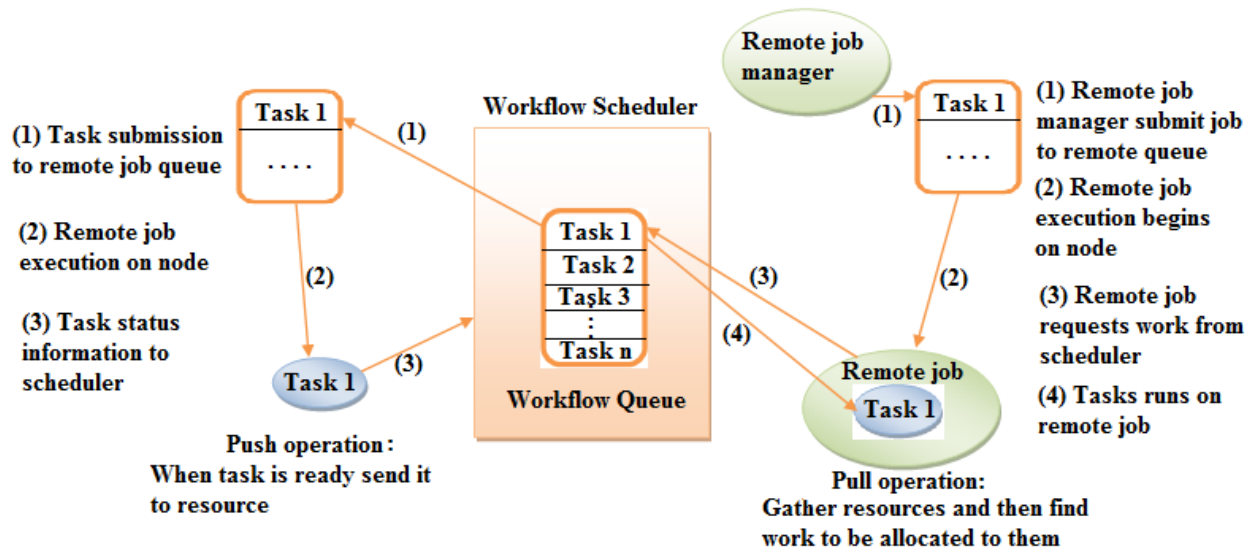


Figure 3.3: Workflow task assignment

collects tasks from different users, pushes them into the queue, and tasks remain in the queue until they get processing resources. Remote users submit their jobs for processing at fog nodes. The workflow scheduler assigns these jobs to fog's local controller according to their priority. When resources are available, tasks are sent for execution. After the execution of tasks, the job scheduler is informed about the status of tasks. In this way, the fog layer load can be reduced, as the jobs are assigned when resources are available. Fog cluster's local controller keeps an eye on load on virtual machines. If VMs are overloaded, tasks are taken from the corresponding VM and assigned to other idle VM. Load balancer balances the load on fog nodes.

3.1.3 Workflow models for load balancing in fog computing

Workflows in fog computing are considered NP-complete problems and can be defined as Directed Acyclic Graph (DAG) that can be denoted by a set of vertices $(V_1, V_2, V_3, \dots, V_n)$ and edges $(E_1, E_2, E_3, \dots, E_n)$. Here vertices denote the set of tasks mapped onto the set of VMs $(VM_1, VM_2, VM_3, \dots, VM_n)$, and edges represent the communication between tasks T $(T_1, T_2, T_3, \dots, T_n)$. The edges in workflows are given weights by providing communication and computation time of each job. These tasks are assigned to resources in fog and cloud layer R $(R_1, R_2, R_3, \dots, R_n)$. Here are some models represented in this section. They are time, cost, energy models, and makespan and objective functions for workflows in fog computing.

Time model

The numerical solution can be used to measure the available execution time when the workflow is being performed to track the workflow execution phase or reschedule the remaining tasks. According to Chirkin et al. [159], when calculating the workflow execution period, task dependencies should be considered along with task heterogeneity and computational capital. Another critical aspect of scientific workflows is that certain runtime elements are stochastic in design from the standpoint of estimation. The execution time in workflows is calculated as the total time taken by any process from its submission to completion. This time also considers when any process remains in the queue, i.e., the time to wait for resources or the time to wait for another task to complete.

$$T_t = \sum_{x=1}^{VM_x} T_R + \sum_{x=1}^{VM_x} T_P + \sum_{x=1}^{VM_x} T_W \quad (3.1)$$

T_t = Total time

T_R = Receiving or passing the time of task

T_P = Processing Time of task

T_W = Waiting Time of task

Cost model

Cost in scientific workflow execution is considered the total Movement Factor (MF) and the Cost Factor (CF). MF can be calculated as the ratio of cost taken during the execution of a task considering migration cost to the VM cost. CF can be calculated as the ratio of total process cost multiplied by memory taken by the task to the VM and data centre cost.

$$T_c(\text{TotalCost}) = (MF + CF)/2 \quad (3.2)$$

Where MF= Movement Factor

CF= Cost Factor

$$MF = \frac{1}{\text{Number of host in data center}} \sum_{x=1}^{VM_x} \frac{\text{Number of migration used VM}}{\text{used VM}} \quad (3.3)$$

$$CF = \sum_{x=1}^{VM_x} \frac{\text{Cost to process} * \text{Memory of task}}{VM * \text{data center}} \quad (3.4)$$

Where, VM_x defines the total number of VM in the system. The actual cost can be calculated as the sum of under deadline task's cost and deadline crossed task's cost.

$$\text{ActualCost} = \text{Underdeadline task's cost} + \text{Deadline crossed task's cost} \quad (3.5)$$

Energy model

Energy is the sum of total time, movement factor, and cost factor of the total number of instances. The following equation shows the energy consumption by the fog environment while executing workflows.

$$\text{Energy} = \sum_{x=1}^{VM_x} (T_t + MF + CF) * \text{number of instances} \quad (3.6)$$

T_t represents total time, MF is the movement factor, and CF is the cost factor.

Makespan

Expected time to compute matrix $ETC(T_j, R_j)$ where T_j is the sequence of tasks, and R_j is a sequence of resources, that is the precondition of tasks. The postcondition shows that the sequence in which the tasks are executed in order to minimize the makespan. In order to reduce the makespan, the load is distributed among available resources. The makespan (MS) can be computed as follows:

$$MS = maximum(C(T_j, RR_n)) \quad (3.7)$$

Where C is the task completion time and is computed by $C=RR_n+ER_n$

Here RR_n represents ready time of resource n.

ER_n represents the execution time of task j for resource n.

Objective function

On the basis of total makespan, energy model, cost model, and time model, that are determined above, the objective function of this paper can be defined as follows.

$$f(p) = \alpha * (T_t + T_c + E + MS) \quad (3.8)$$

Here, $f(p)$ is considered the model's objective function, which should be as minimum as possible for the best solution. When optimization is achieved in the algorithm it means fitness value is achieved. T_t, T_c, E, MS represents total time, total cost, energy, and makespan, respectively.

3.2 Hybridized load balancing algorithm for scientific workflows (Tabu-GWO-ACO)

Load distribution in fog computing needs an equal proportion distribution of tasks among all fog nodes. VM load balancing requires detecting all the nodes' workload within some time to get resources within time. This section describes different optimization techniques used for this study and proposed a hybridized load balancing

algorithm for scientific workflows that are based on the combination of Tabu-GWO-ACO.

3.2.1 Optimization methods used

The key aim of using load balancing optimization techniques is to minimize the cost function. This research has used the tabu search, Grey Wolf Optimization(GWO), and Ant Colony Optimization(ACO), among other approaches. Tabu search improves local search efficiency by taking into account load balancing [28]. The key reason for using tabu search is that online computations are needed in specific layers, and tasks should be processed when they arrive. Natesan and Chokkalingam [160] have shown that grey wolf works actively to reconfigure and balance loads. In assigning the individual tasks allocated for the virtual machine implementation, GWO used ACO to improve performance.

Tabu search

Tabu search helps to find a feasible solution by reducing circuit simulations. Tabu search technique has online computations that help for processing of task as it arrives. The tabu search method uses a flexible memory that makes the search more versatile. Tabu search method is used to search overloaded and underloaded nodes so that proper load can be distributed among all the nodes [114] [161].

The tabu search's fundamental concept is to penalize actions that take the solutions into search areas previously explored. On the other hand, tabu search accepts non-improving options deterministically to avoid being trapped in local minimums. The research work undergoes an initial solution and generates a series of adjacent alternatives to the current solution, s' , which is marked as S_n . The tabu list solutions are omitted from this set of solutions, excluding those following the Aspiration Criterion. Then by removing all obsolete options from the list to update it.

$$s' \in N(s) = \{N(s) - T(s)\} + A(s) \quad (3.9)$$

Choose from $N(s)$ the best solution s' . Update the best solution if the solution is better than the current best one. Here upgrade s to be s' , regardless of whether s' is more substantial than s . Here $T(S)$ represents tabu search list that contains the list of all moves in tabu search. This list is modified by removing the expired moves and added new move s' in it. Furthermore, $A(S)$ is the aspiration criteria that is considered to update set of solutions in tabu search.

Ant colony optimization

ACO technique solves computational problems with probabilistic methods, finding proper routes to distribute the load among all nodes. It works based on the behavior of real ants for collecting food. This algorithm is used to find the optimum path for those tasks which are discarded by overloaded nodes, and the path helps them to reach the underloaded nodes [81] [162].

Every ant produces a solution in the first step of solving a problem. The second phase contrasts the paths of numerous ants. Moreover, pathways or pheromones are modified in the third step. ACO optimizes a challenge by upgrading the pheromone trail in the search area and shifting these ants according to basic mathematical formulas about the likelihood of transformation and complete pheromones. ACO produces and measures the fitness of world ants at each iteration. Finally, Update the vulnerable regions' pheromone and edge.

When fitness is increased, local ants are moved to better areas. Otherwise, a random search path is selected. The ACO is focused both on local and international searches. Local ants can switch into the latent zone with the best option for a region x .

$$P_x(i) = \frac{Tph_x(i)}{\sum_{j=1}^n i_j(i)} \quad (3.10)$$

Where $Tph_x(i)$ is the complete pheromone in area, i is the total number of tasks. Update the pheromone by $i_t(i+1) = (1 + e_r)i_t(i)$. ACO is simple to integrate with other approaches, and it performs well when solving complex optimization problems.

Grey wolf optimization

GWO is a meta-heuristic optimization approach. GWO mainly depends on the principle of the grey wolf's nature to hunt cooperatively. The model structure of GWO makes it different from others. It is a large-scale search method centered on three optimal samples. The hunting behavior of grey wolves inspires grey wolf optimization. The wolves always live in groups. The hierarchy of grey wolves is divided into four primary levels, i.e., alpha(α), beta(β), delta(δ), omega(ω). Here α acts as the leader of the group that takes all hunting decisions. β are the subordinates of α , which help them to take decisions. δ acts as scouts that have to report α and β , but they dominate ω . ω are the last wolves in the group allowed to eat in the last [163].

When developing GWO, the fittest approach was used to mathematically model wolves' social hierarchy alpha(α). Consequently, beta(β) and delta(δ) are the second and third-best alternatives, respectively. Taking X and Y as coefficient vectors to describe the wolfs' encirclement of prey,

$$\vec{X} = 2\vec{x} \cdot \vec{v}_1 - \vec{x} \quad (3.11)$$

$$\vec{Y} = 2 \cdot \vec{v}_2 \quad (3.12)$$

When iterating, the components of x decrease to zero, and v1 and v2 are random vectors. To mathematically model grey wolf's hunting behaviour, the first three best solutions obtained so far will be saved, requiring the other search agents to change their positions following the best search agents' position.

$$\vec{D}_a = |\vec{Y}_1 \cdot \vec{P}_a - \vec{p}|, \vec{D}_b = |\vec{Y}_2 \cdot \vec{P}_b - \vec{p}|, \vec{D}_c = |\vec{Y}_3 \cdot \vec{P}_d - \vec{p}|, \quad (3.13)$$

$$\vec{P}_1 = |\vec{P}_a - \vec{X}_1 \vec{D}_a|, \vec{P}_2 = |\vec{P}_b - \vec{X}_2 \vec{D}_b|, \vec{P}_3 = |\vec{P}_c - \vec{X}_3 \vec{D}_c|, \quad (3.14)$$

$$\vec{P}(t+1) = \frac{\vec{P}_1 + \vec{P}_2 + \vec{P}_3}{3} \quad (3.15)$$

Here, grey wolves position vector is denoted by \vec{P} , and $\vec{P}_1, \vec{P}_2, \vec{P}_3$ are the position vectors of the α, β, δ respectively.

3.2.2 Proposed hybridized algorithm Tabu-GWO-ACO

All defined algorithms are hybridized to implement load balancing in the scientific workflow application. Our proposed algorithm is a combined form of tabu search method, GWO algorithm, and ACO algorithm. All these algorithms are combined in one algorithm to achieve good results in fog load balancing. Tabu search method works well to search over-utilized and underutilized fog nodes; ACO and GWO are both excellent for optimizing the fog environment. When workflow tasks are generated, they are parsed and mapped on fog nodes for processing. Firstly, the tabu search method is applied to check the utilization of fog nodes. Sometimes tasks are allocated to few fog nodes only, and others remain underloaded so that the tabu search method will recognize the over-utilized and underutilized fog nodes. On underutilized fog nodes, ACO will be applied for optimization. On over-utilized fog nodes, GWO will work for optimization. ACO and GWO work better than the tabu search method to obtain better load balancing. The proposed algorithm is implemented in four parts. Following are some steps that thoroughly explains the proposed algorithm:

step I: Firstly, the number of fog nodes are initialized as N , and workflows as W , respectively. Make parsing trees of fog nodes. Then tasks are extracted from workflows and mapped onto fog nodes.

Step II: When all tasks are mapped onto fog nodes, fog nodes are searched to utilize available resources. Fog nodes can be over-utilized and under-utilized. So, the tabu search method is applied to check the utilization of fog nodes. This algorithm will continue searching nodes until it finds over-utilized and under-utilized nodes. All neighboring nodes are checked for their utilization, and when over-utilized or under-utilized nodes are found tabu list is updated.

Algorithm 2: Hybridized load balancing algorithm for scientific workflows (Tabu-GWO-ACO)

Input: Workflow number and fog nodes

Output: Optimize cost and delay

1 **A. Initialization Algorithm (W,T)**

2 Initialize the number of fog nodes and workflows $N \leftarrow$ fog nodes, $W \leftarrow$ Number of workflows, and Parse workflows

3 **while** W **do**

4 └ Parse W_i , Extract task $T_i \leftarrow W_i$

5

$$X \leftarrow \sum_{i=1}^n [T_i + W_i] \quad (3.16)$$

Map Task on fog nodes $N \leftarrow X$

6 **B. TABU search (N,W,X)**

7 Set $T = T_0$ and $Iteration = N$

8 Repeat, and set $m = 0$, $worst = 0$, $it = it+1$ // Here "it" is iteration

9 Repeat, and set set $m = m+1$

10 Execute searching neighbour (T, T_m)

11 Execute searching node (N, N_m)

12 **if** $D(N_m) > D(N)$ **then**

13 └ until $D(N_m)$

14 └ update Tabu list (T, T_m)

15 └ $N = N_m$

16 └ **if** $DN_m \leftarrow Tabulist(T, T_m)$ **then**

17 └ until iteration = max_{it}

18 └ Selected nodes

19 **C. Grey Wolves (N)**

20 Initialization Population $(S(N))$

21 Define all combination $X \leftarrow 2^N$, $Wolves \leftarrow X$, $X_\alpha \leftarrow G_\alpha$, $X_\beta \leftarrow G_\beta$, $X_\delta \leftarrow G_\delta$

$$X_{update} = (G_\alpha + G_\beta + G_\delta)/3 \quad (3.17)$$

22 **D. ACO** (X_{update})

23 $ApplyAnts \leftarrow X_{update}$

24 update pheromones, optimize fitness value $f(p)$

25 **if** $optimization\ exist$ **then**

26 go to the step 29

27 **else**

28 go to step 2 and repeat again

29 $Threshold \leftarrow optimizevalue$

30 Migrate task according to the threshold and Analyze time, cost, and energy.

Step III: After finding under-utilized and over-utilized nodes, GWO and ACO methods are applied for optimization.

In the proposed solution, three approaches are hybridized: GWO, ACO, and Tabu. Here, hybridized does not imply contemporary work but instead works on the same objective function. In this case, hybridized usage is concerned with obtaining a quick and natural convergence rather than taking time and enforcing convergence. The first input to GWO, which operates for all variations in a given number of iterations (500), monitors the time and then reduces the area of variation and is assigned to ACO. All parameters are not simplified, and no multi-objective have been used in this work, which has the inherited problem of implementing convergence. This research has chosen these optimization methods because they need global optimization, which GWO and ACO also provide. Because of the speed at which ACO searches, it achieves high convergence compared to all other optimizations. GWO is experimenting with semantic optimization that provides an association between input space and quick convergence in a larger population.

3.2.3 Flow of execution of Tabu-GWO-ACO

This section contains a framework explaining the flow of execution of the Tabu-GWO-ACO algorithm. FOCALB has three layers similar to fog's basic architecture. The implementation of scientific workflow tasks is presented through the framework in this section. End users generate workflow tasks, which are mapped to the fog nodes near them. These fog nodes are further connected to cloud data centers. After initializing tasks to fog nodes, these nodes are searched for over-utilization and under-utilization. Tabu search method is applied on over-utilized nodes for searching, which checks the nodes with time delay. Load balancing is applied to these nodes to avoid over-utilization of nodes. Nodes are reviewed based on time delay for task execution; if some nodes are free, tasks from overloaded nodes migrate to free nodes. Load balancing is done using GWO. Here, three parameters have been considered, i.e., α , β , γ , as best searching agents in GWO, which search for free resources in fog nodes. If the convergence point is reached, then the threshold is sent to the fog node, and tasks are migrated to the cloud. If no converge point is reached, tasks are initialized, and again GWO is applied.

Time delay, cost of execution, and energy consumption in fog nodes are analyzed. This process works in the case of over-utilized fog nodes.

On the other hand, when fog nodes are underutilized, ACO balances the load on fog nodes. ACO works on the behavior of ants in search of food. Pheromones are updated after all ants completed their tour, and after convergence, the task from overloaded fog nodes is taken and transferred to under-utilized fog nodes. Figure 3.4 shows the proposed methodology followed to achieve load balancing in the fog computing environment, and the algorithm is given in Algorithm 2.

3.3 Verification and validation of proposed framework-FOCALB

This section mainly focuses on the verification and validation of the proposed framework. Experimental requirements have been discussed, and performance evaluation criteria have been described. The proposed load balancing framework and proposed Tabu-GWO-ACO load balancing algorithm have been implemented in the iFogSim toolkit. This work aims to provide a load balancing strategy for workflow-based applications of fog computing. The proposed technique tries to reduce execution time, implementation cost, and energy consumption in fog nodes. Three types of experiments have been percolated to analyze the validity of the proposed work. Three test cases have been provided to present the experimental results. In the first test case, the cost of implementation has been analyzed. In the second test case, the execution time of workflows applications has been represented. In the third test case, energy consumption by different resources has been described. Approximately 2 to 200 fog nodes have been taken to calculate the results. An average of forty runs has been done to ensure the statistical correctness of experiments.

This section contains the simulation results obtained using iFogSim, a simulator used for edge computing, IoT and fog environment for IoT service management, and modeling and simulating networks and different applications. iFogSim works with the CloudSim in an associated form where CloudSim has a vast library containing cloud environment simulation and resource management. CloudSim handles the events be-

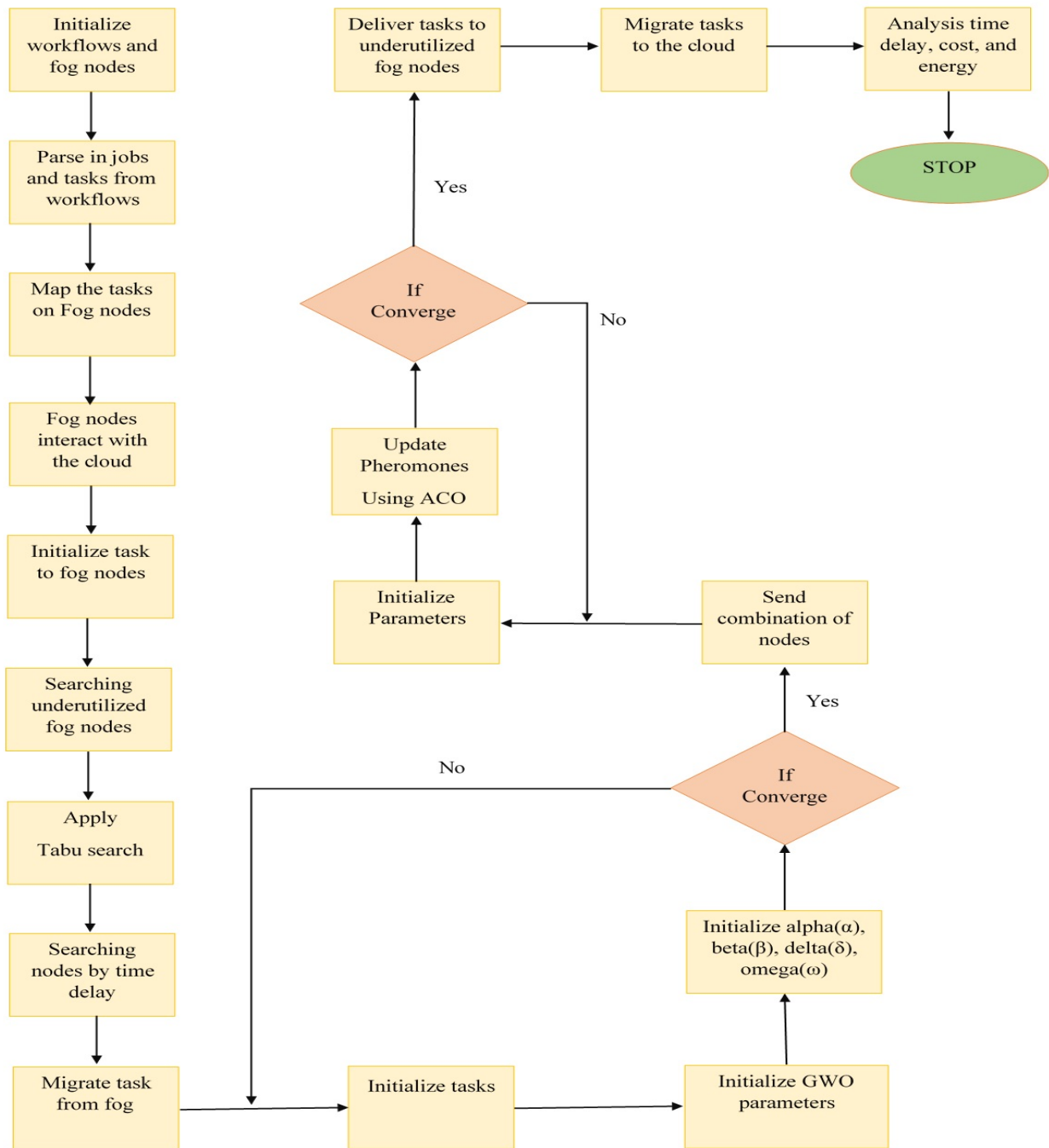


Figure 3.4: Tabu-GWO-ACO methodology

tween fog components.

3.3.1 Experimental setup

Few experimental requirements have been considered to evaluate the proposed approach. This work used 64-bit Windows 7 operating system. iFogSim, the most powerful simulation tool, has been used to represent simulation results. The fog computing layer has been divided into the form of fog clusters containing several fog nodes. The rest of the requirements are explained in the form of a table.

Table 3.1: Required parameters

Parameter	Value
Operating system	Windows7 64 Bit
Simulator	iFogSim
MIPS	2000
No.of fog nodes	2 to 200
No.of Hosts	1 to 2
RAM	200MB
Number of tasks	100-1000
Number of workflows	10 to 12
Bandwidth	up to 60 Mbps

Table 3.1 describes the requirements to achieve simulation results.

iFogSim is an open-source, high-performance toolkit used for fog computing, IoT, and edge computing environments. It is used to simulate fog computing and IoT networks. iFogSim works with CloudSim in collaboration. iFogSim has three main components, i.e., physical components that include physical fog nodes, logical components that contain different application modules and application edges, and lastly, management components that include module mapping objects and fog controller [164].

Why iFogSim for simulation results?

iFogSim is an open-source, high-performance toolkit used for fog computing, IoT, and edge computing environments. It is used to simulate fog computing and IoT networks. iFogSim works with CloudSim in collaboration. iFogSim has three main components, i.e., physical components that include physical fog nodes, logical parts that contain different application modules and application edges, and lastly, management components that include module mapping objects and fog controller [164].

Due to its simple interface and low complexity, iFogSim is used in this work. The iFogSim simulation toolkit is based on the simple CloudSim platform. CloudSim is one of the widely accepted Cloud computing simulators. Extending the abstraction from basic CloudSim classes, iFogSim provides space for simulating custom fog computation with several fog nodes and IoT computers (e.g., sensors, actuators). However, the groups are annotated in iFogSim so that users with no previous knowledge of CloudSim can conveniently identify the fog computing infrastructure, service placing, and resource allocation policies. iFogSim uses sense-process-actuate and the distributed data flux paradigm when simulating every fog computing environment scenario. It makes it easier to assess end-to-end latency, network congestion, electricity use, operating costs, and quotas [165].

3.3.2 Results and discussion

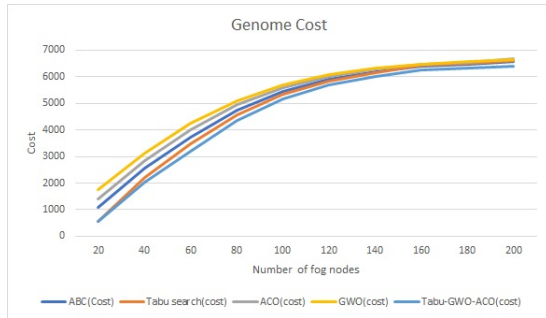
Scientific workflows are the representation of tasks in the form of Directed Acyclic Graphs (DAG). Different sensors generate these tasks, actuators in different applications like astronomy, e-healthcare, intelligent traffic management, and many other application scenarios [112] [166]. There are various scientific workflows, i.e., Cybershake, Genome, SIPHT, LIGO, Epodonomic [167]. In DAG, tasks are represented in connected nodes and edges in which nodes represent the tasks and edges represent communication between them. The research work considered LIGO, Cybershake, SIPHT, and Genome workflow samples for our experimental results. Cybershake is utilized to characterize the earthquake hazards by Southern California earthquake center [109]. Cybershake can also classify as a data-intensive workflow having significant Cpu and

memory requirements. LIGO workflow stands for laser interferometer gravitational observatory used in the area of physics. It is used to detect the earth's gravity. LIGO have More memory and CPU requirement as it contains tasks of large size [122]. Most of the tasks of LIGO need memory optimization-based VMs [168]. SIPHT is used at Harvard University in bioinformatics projects to detect bacterial replicons. It is used to search Bacterial small RNAs (sRNA) that regulate the secretion process in bacteria. National center uses SiphT workflow to automate search for encoding genes in sRNA [168] [115]. Hans Winkler created GENOME in 1920. It is used in the field of genetics and biology to collect an organism's genetic material. It contains RNA or DNA of viruses. It may include non-coded DNA or coded genes. Genomics is the study of genomes [169].

This section will provide the results obtained after implementing the proposed algorithm. It is not easy to execute scientific workflow data sets in a real-time environment; hence they are executed in the simulation environment. Scientific workflow data sets are run on the iFogSim simulator using Eclipse to minimize execution time, cost, and energy consumption.

Different scientific workflow data sets (i.e., LIGO, Cybershake, Genome, and SiphT) are considered for experimental results. Two algorithms run on iFogSim, i.e., traditional tabu search method and our proposed method tabu-GWO-ACO method. It has been obtained from the experimental results that the combined form of tabu-GWO and ACO methods works better than the tabu search method for achieving efficient load balancing in the fog computing environment.

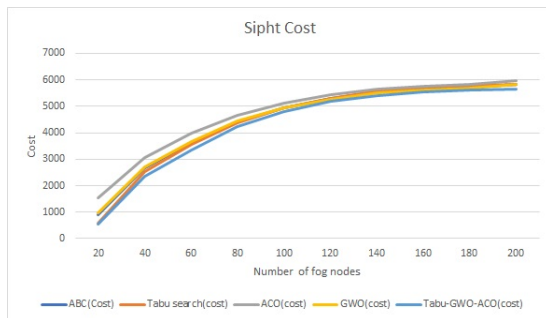
For simulation results, iFogSim has been used to check the performance of the Tabu-GWO-ACO load balancing technique. Proposed technique results are compared with existing approaches to verify that performance of Tabu-GWO-ACO is better than ABC, ACO, Tabu search, GWO. Various applications have been investigated to analyze the performance of our proposed technique



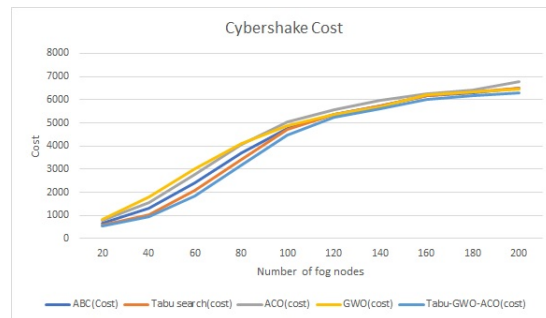
(a) Genome



(b) LIGO



(c) Sipht



(d) Cybershake

Figure 3.5: Cost Analysis of Tabu-GWO-ACO approach

Test case I: Cost analysis

Different type of workflow tasks are assigned to fog nodes, and their performance is analyzed. If more number of fog nodes are used in the fog layer, then their cost consumption will be more. This research have considered Genome, Cybershake, Sipht, LIGO workflows for execution and compared different existing approaches with the proposed one.

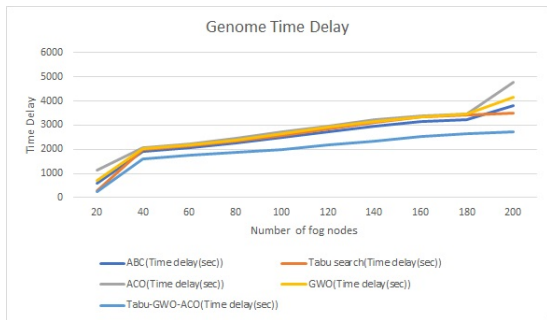
In Figure 3.5, four subFigures show different workflow execution results. In Figure 3.5(a), execution of Genome scientific workflow execution result is displayed. The graph shows the number of fog nodes on the x-axis and the cost on the y-axis. With the increase of fog nodes, implementation cost is also increased. This research proposed a hybrid approach containing tabu search, grey wolf optimization, and ant colony optimization algorithms in hybrid form. With the Tabu-GWO-ACO approach, the

cost of implementation has been by applying load balancing. The Figure shows that the proposed method reduces the cost compared to other techniques, i.e., ABC, ACO, Tabu search, GWO. Similarly, other workflows, i.e., LIGO, Sipht, and Cybershake tasks, have been provided to fog computing and stored their results. These results are shown in Figure 3.5(a),(b),(c) and (d), respectively. All the results obtained by implementing the different workflow tasks are assigned to fog nodes, and their performance is analyzed. If more fog nodes are used in the fog layer, then their cost will be more. This research work considered four different scientific workflows, i.e., Genome, Cybershake, Sipht, LIGO, for execution and compared existing approaches with the proposed one. In Genome and LIGO, implementation cost is reduced by 3% using Tabu-GWO-ACO than other existing techniques. On the other hand, for Sipht and cybershake workflows, 4% is reduced with Tabu-GWO-ACO than in different current approaches.

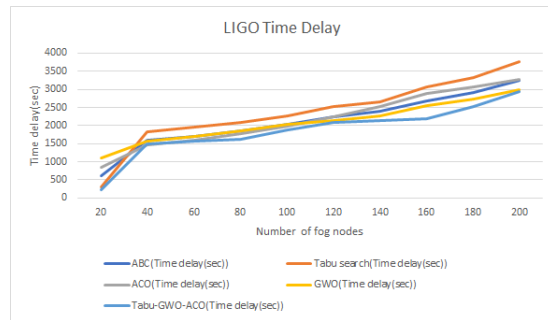
Test case II: Execution time analysis

Scientific workflows, i.e. Genome, LIGO, Sipht, and cybershake, is having large datasets from which tasks are parsed to fog nodes. The fog layer needs more number of nodes to execute these large tasks. Execution time increases as the number of tasks are increased. This work have implemented the proposed technique and analyzed the execution time of tasks in the fog layer.

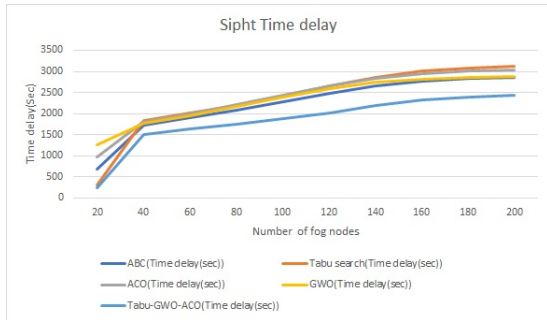
Figure 4.6 shows the analysis done for the execution time of different workflow tasks. The execution time of tasks is calculated by the time taken from the submission of tasks till the execution completes. Time taken in the queue is also calculated. Figure 4.6 have four parts 4.6(a),(b),(c),(d) representing execution time of each scientific workflows Genome,LIGO,Sipht,Cybershake respectively.The x-axis shows the number of fog nodes in the graphs, and the y-axis represents execution time. It can be seen from graphs that with the increase of fog nodes, execution time also increases. This research works to reduce the execution time of tasks with the help of the proposed hybridized load balancing algorithm for scientific workflows (Tabu-GWO-ACO). In Genome and LIGO, execution time is reduced by 25% and 12%, respectively, using



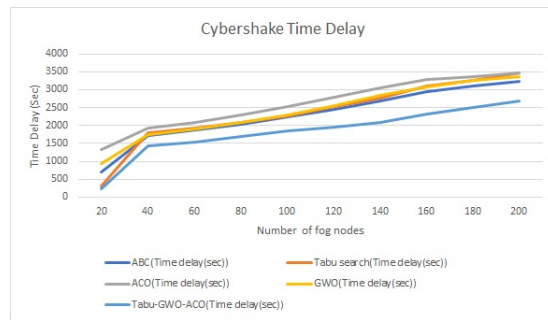
(a) Genome



(b) LIGO



(c) Sipht



(d) Cybershake

Figure 3.6: Execution time Analysis of Tabu-GWO-ACO approach

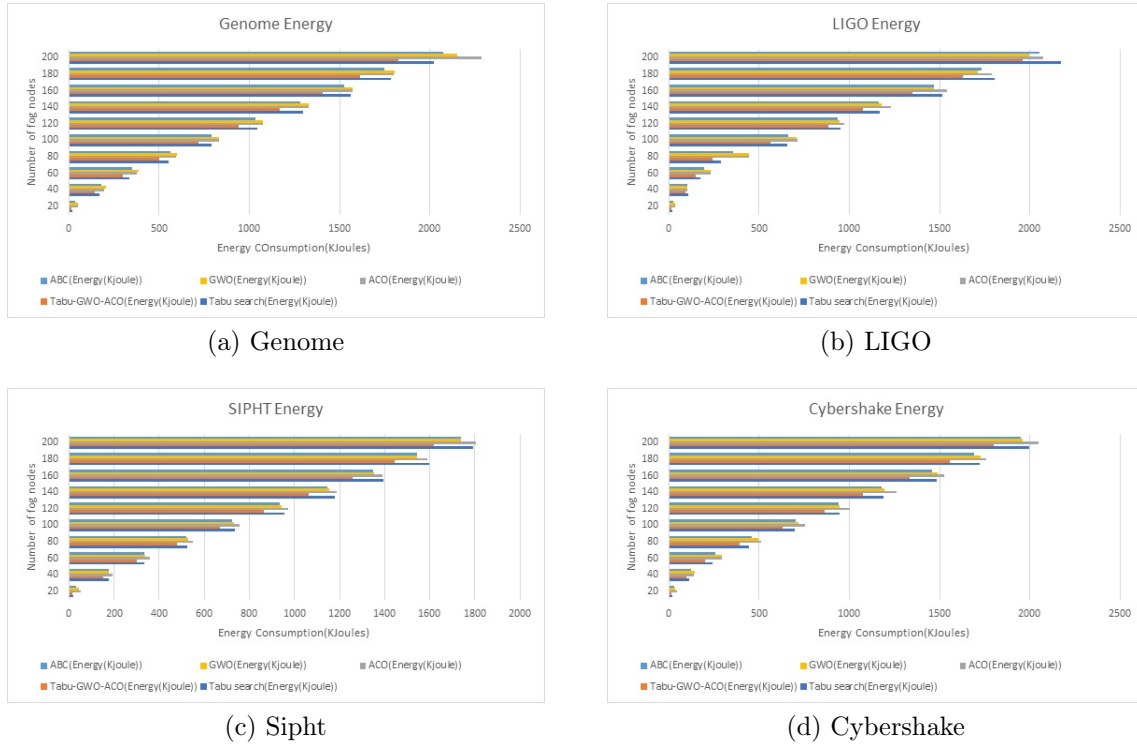


Figure 3.7: Energy Analysis Tabu-GWO-ACO approach

Tabu-GWO-ACO than other existing approaches. On the other hand, Tabu-GWO-ACO tries to minimize the execution time of Sipt and cybershake workflows by 18% and 20%, respectively.

Test case III: Energy consumption analysis

In this case, energy consumption has been analyzed by different fog nodes in the fog layer. In the case of a large number of tasks, more resources are required. When the number of resources is more, more energy is consumed. Four different scientific workflows, i.e., Genome, Cybershake, Sipt, LIGO, have been considered in this work for execution and compared different existing approaches with the proposed one.

Figure 3.7 is used to depict the energy consumption in fog nodes using the Tabu-GWO-ACO approach. The Figures 3.7(a),(b),(c),(d) represents energy consumption in the

fog layer in which the number of fog nodes is represented on the y-axis and energy consumption on the x-axis. In more tasks, the requirement for the number of nodes is increased, due to which energy consumption is also more. With our proposed approach Tabu-GWO-ACO this work tries to reduce energy consumption in fog nodes by load balancing in the fog layer. The Figures show that Tabu-GWO-ACO outperforms all other techniques with which it is compared. For example, in Genome and LIGO, Tabu-GWO-ACO tries to reduce energy consumption in fog nodes by 14% and 6%, respectively. On the other side, in Sipt and cybershake, energy consumption is reduced by 8.40% and 9.55%, respectively.

3.3.3 Applications of proposed architecture and approach in real time environment

With the growth of the internet of things, fog computing has also grown and has a broad scope in many areas nowadays. Fog computing is becoming an alternative to cloud computing for some applications with a distributed environment. Due to more number of users, fog computing faces the problem of load imbalance. So by implementing proposed load balancing architecture in real-time applications, fog computing can work more efficiently. Some applications of proposed architecture FOCALB have been defined as follows:

- **E-HealthCare:** In e-healthcare systems even a fractional delay is unbearable. In case of more IoT devices used for e-health systems, by implementing load balancing in the fog layer, all requests generated by IoT sensors can be equally assigned to all resources for fast processing. Due to its less latency feature, fog can help to build efficient e-healthcare systems that can help to store and process patient’s data in fractional seconds [106].
- **FoAgro:** Fog computing has a broad scope in agriculture also. Different types of sensors can be installed to monitor the crops. Hence, fog nodes can monitor all the activities with the help of data collected from these sensors. In the case of more number of sensors, data requests will be more. Load balancing can enhance the working of fog nodes by speeding up the processing in the fog layer [170].

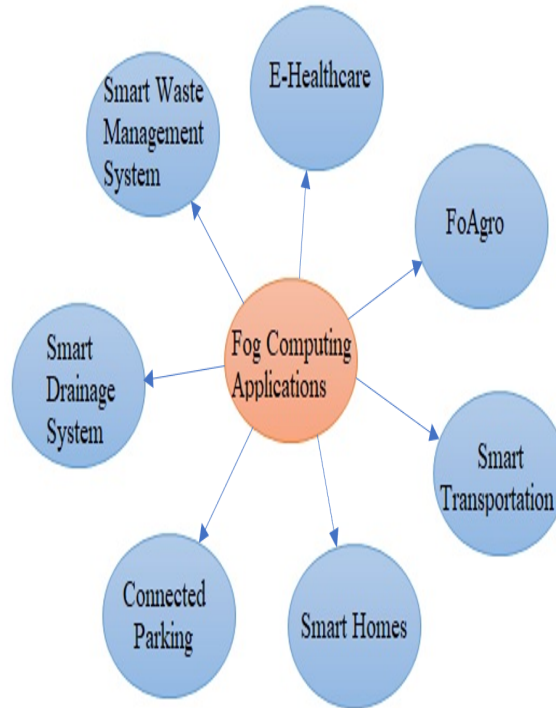


Figure 3.8: Applications of FOCALB

- **Smart transportation** Smart vehicles can be managed with the help of fog computing. Fog nodes can be deployed near roadside units that may collect and store data from smart vehicles and processes faster than the cloud. In large cities, traffic is more, so more data is generated. Sometimes, due to more traffic jams, data can be generated in terabytes. Due to which there may be an overloading of fog resources. So, load balancing can be applied to maintain road traffic smoothly. [48].
- **Smart homes:** Fog can monitor all the smart devices in a home, i.e., smart clock, bulb, coffee machine, and all other necessary devices in our daily life. Fog computing can immediately process the requests generated by these smart devices and help them to immediately respond to the user. In daily routine, data generated by the smart devices can be more, so load scheduling can be applied along with load balancing in fog layer in the computing environment [130].
- **Connected parking System:** Fog computing help to find a free parking lot for the

users of their cars. Fog nodes installed at parking areas collect data continuously from nearby sensors in the parking area. According to this data, fog nodes update the available parking lots online to easily find parking spaces for them. Load balancing can be applied in fog computing to handle a massive number of requests generated by a large number of vehicles and parking sensors [170].

- **Smart waste management:** Nowadays, with the growth of population, garbage is also growing. A waste management system can be a solution for the betterment of the environment today. In large cities, garbage collection is in tons, and waste bins are filled faster. Hence more data is generated by sensors installed in containers. Here load balancers can be installed in fog nodes to handle this data efficiently. So, fog computing can help to generate such kind of system that can help to collect and manage the waste more efficiently [130].

The proposed load balancing framework FOCALB can be used to implement real time application i.e. smart waste management systems to build smart city homes. This chapter proposes a prototype of waste management system implementing load balancing at fog layer. The proposed prototype model contains three layers: Smart sensor devices, fog nodes/ servers, Cloud datacenters. Smart bins will be deployed which will contain the sensors. These Smart bins will be further tied up to the fog layer which filters the data to be passed to the cloud. Load balancer will balance the load on the fog nodes. It will divide the load equally on all the nodes. Fog nodes will generate alert messages to notify the garbage carriers to collect the waste. Security sensors will also be installed in the bins which will inform about the safety of the bins. If anyone will try to temper the bins then these sensors will start playing loud noise signals which can save the stealing of bins. The Smart bins will be connected through Wi-Fi, 3G/4G, LTE to the fog layer. An application will be generated to operate the whole scenario. Fog layer will be further connected to the cloud. CloudSim will be used to manage the cloud and the fog nodes.

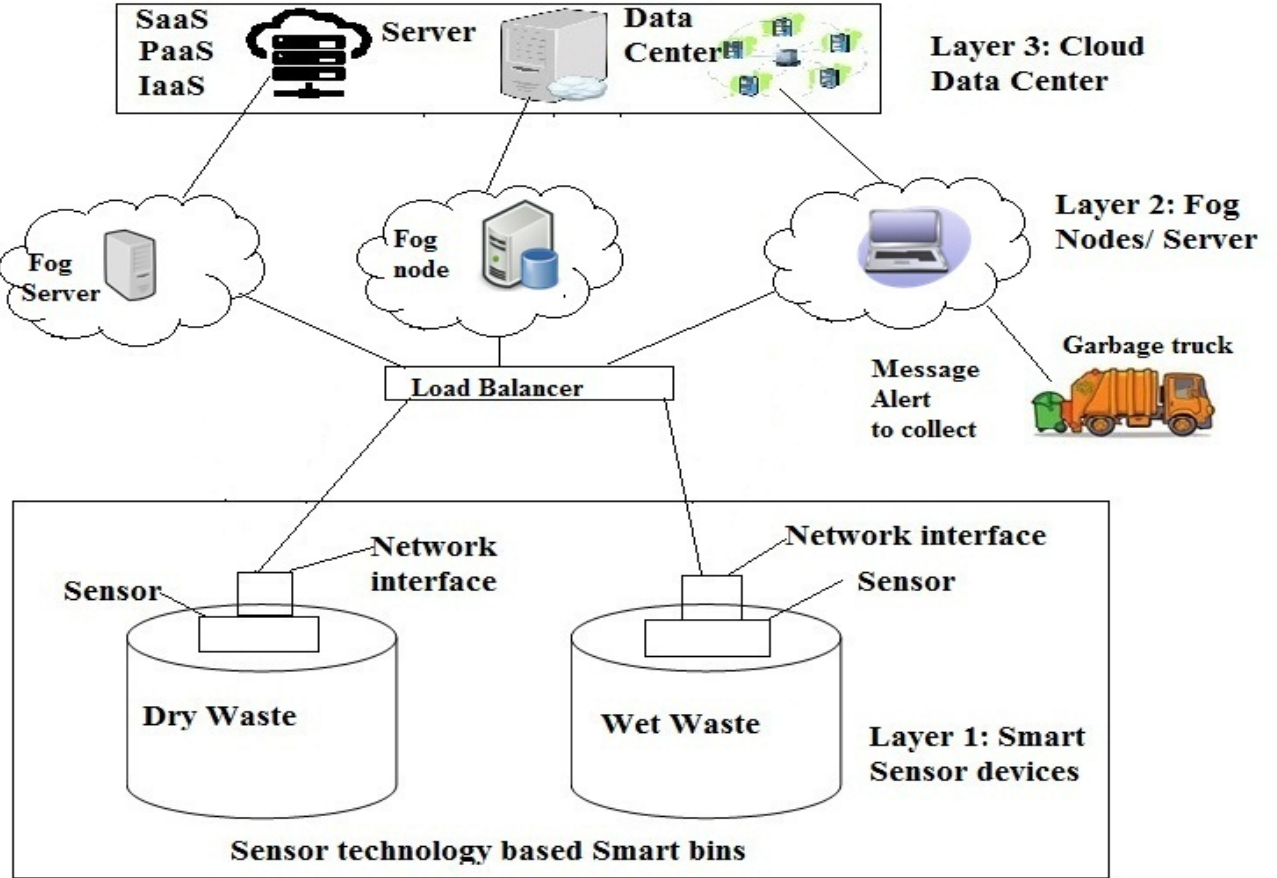


Figure 3.9: Prototype of Waste management system

3.4 Conclusion

This chapter described the proposed Resource-utilization based load balancing framework for the fog environment. Different workflow models have been described that are considered to analyze the performance of the proposed framework. A load balancing model has been proposed for scientific workflow applications in the fog computing environment. Scientific workflow applications have been considered to evaluate the proposed technique. The proposed algorithm Tabu-GWO-ACO is a hybrid form containing three different algorithms, i.e., Tabu search, GWO, and ACO. Tabu search algorithm is used to find out the underloaded and overloaded fog nodes, and then GWO and ACO are used to optimize fog nodes. The proposed approach mainly tries

to enhance resource-utilization by proper load balancing in the fog computing environment. Further, the proposed FOCALB model has been implemented in simulation environment implementing proposed load balancing approach Tabu-GWO-ACO and analyzed that obtained results outperform the other existing approaches. Furthermore, real time applications of FOCALB have been described and one prototype of smart waste management has been proposed to be implemented in real time environment in future works. Smart waste management system can help to develop a clean and green city.

CHAPTER 4

An energy-efficient load balancing approach for scientific workflows in fog computing

The previous chapter discussed the proposed resource utilization-based load balancing framework for scientific workflows. In this chapter, an energy-aware load balancing algorithm has been designed considering scientific workflow applications. Fog computing has to face the load balancing problem having all valuable features and limited storage capacity. Many internet users using smart devices keep on sending data simultaneously, which causes a shortage of resources. Sometimes only a few resources are utilized, and the others in the fog layer remain unused, hence wastage of resources and wastage of power to keep them on. Load balancing becomes a challenging task in the fog computing layer to reduce the cost and energy usage. With the imbalance in load in the fog layer, bandwidth is also wasted, providing less throughput, and response time to the user increases. All this happens due to a highly restricted environment and limited resources availability [55].

This chapter firstly introduces scientific workflows. A further resource-utilization-based workflow execution model for fog computing has been proposed. Furthermore, the meta-heuristic techniques-based energy-aware load-balancing algorithm is named as PSW-Fog Clustering-based Load balancing algorithm. This algorithm is hybridized from PGO, WCO, and SAA algorithms. PWS considers three main parameters to analyze the results, i.e., time delay, computational cost, and energy consumption. This approach aims to reduce the energy consumption in fog computing by maximizing resource utilization. Load balancing is applied to optimizing the use of resources in the fog layer. Fog nodes in the fog layer are combined in few clusters containing few fog nodes each. Scientific workflow applications are considered to evaluate the proposed approach.

4.1 Scientific workflows

Fog computing is the most trending technology in the Internet of Things (IoT) nowadays. The fog has removed the barriers of computing and storing IoT data at cloud datacentres by providing local storage and processing services. Fog computing brings computing and storage services local to the end-users and enhances the popularity of IoT. The system has to decide where the applications have to execute, i.e., in the fog layer or the cloud, to fulfill the quality of the service requirements. A cloud-fog scheduler should be installed to make the system's execution decision to avoid any delay in task processing. Load balancing plays a vital role in enhancing the performance of a fog computing system. Due to the distributed nature of the fog environment, load balancing becomes a very challenging task. The load distribution mechanism becomes difficult due to more users' presence, which leads to load fluctuation in the fog environment. For maximum utilization of resources in a fog environment, the load should be distributed among all available VMs to avoid overloading and underloading resources in the fog computing layer.

Scientific workflows are data-intensive applications representing distributed data sources and complex computations in various domains, i.e., astronomy, engineering sciences, and bioinformatics. In distributed environments like fog computing, multiple sensors and experimental processes generate a large volume of data that needs collection and processing within specific time constraints. Geographically distributed fog resources can be used to collect and process this data. Although fog computing has numerous advantages over cloud computing, it also faces many challenges [112]. One challenge is load balancing in scientific workflow task execution in the complex environment of resources. Scientific workflow tasks require real-time implementation, but fog computing resources can be overloaded due to the large volume of data. Hence there is a need to balance the data among existing resources in equal proportion to be processed in real-time. Even distribution of tasks among all the resources can result in proper utilization of resources, hence save energy and execution time also [109].

Workflows in fog computing are considered NP-complete problems and can be defined as Directed Acyclic Graph (DAG) that can be denoted by a set of vertices $\{V_1, V_2, V_3, \dots, V_n\}$ and edges $\{E_1, E_2, E_3, \dots, E_n\}$. Here vertices denote the set of tasks

mapped onto the set of VMs, i.e., $\{VM_1, VM_2, VM_3, \dots, VM_n\}$, and edges represent the communication between tasks T, i.e., $\{T_1, T_2, T_3, \dots, T_n\}$. The edges in workflows are given weights by providing communication and computation time of each job. These tasks are assigned to resources in fog and cloud layer R, i.e., $\{R_1, R_2, R_3, \dots, R_n\}$. This section represents the time, cost, energy models, and makespan and objective function for workflows in fog computing. Figure 4.1 below shows the example of simple workflows: This section defined different type of common workflows such as Montage [171],

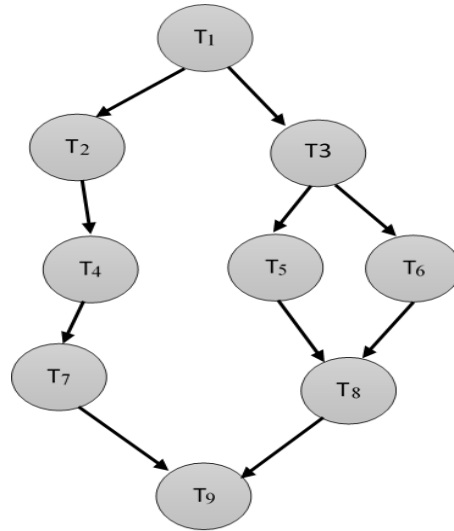


Figure 4.1: Example of workflow

Sipht, CyberShake [172], Ligo and Epigenomics [122]. These different workflows are explained as below:

A montage workflow application is used for astronomic applications, which builds huge picture mosaics of the sky. Montage tasks can be identified into input and output, which do not use more CPU processing capability [171]. CyberShake workflow is used to identify natural disasters, i.e., earthquakes. It is classified as a data concentrated workflow which consumes a large amount of memory CPU capability [172]. Sipht is used in the national center to detect replicates of all bacteria used to collect biotechnology information. These types of workflows need more CPU utilization but less input-output utilization. Laser Interferometer Gravitational-Wave Observatory (LIGO) is the workflow application used for earth's gravity detection. Ligo workflows contain

those kinds of tasks which take more memory for execution in CPU [122]. Epigenomics workflows are used to detect the production of DNA. They are data intensive hence more CPU utilization. They are being used for genome sequencing operations in epigenome center [122].

4.2 Resource-utilization based Workflow execution model for fog computing

This section proposes a workflow execution model for a fog computing environment that is based on resource utilization. While executing larger computational tasks, fog computing faces specific problems like load scheduling, load balancing. Our proposed solution will help to enhance resource utilization and reduce energy consumption in fog nodes. Figure 4.2 shows the proposed workflow execution model for the fog computing environment. The proposed architecture has three layers as that of traditional fog architecture. As shown in the Figure 4.2, five steps describe the working of this proposed model. These steps are described below in the layer-wise format:

End-user layer: The very first layer is the end-user layer in which end users generate a large number of workflow tasks. Workflow container stores these tasks for some time and then assign these tasks to the workflow scheduler. The working of these steps is as follows:

Step 1. Workflow container submits workflow tasks to the workflow scheduler. These tasks are submitted in the manner they arrived in the workflow container.

Step 2. Workflow scheduler contains a queue where tasks wait for resources. The workflow tasks entered the queue from the queue's rear end when they arrived and were removed from the front end.

Step 3. As the resources become available, and these tasks are removed from the task queue and assigned to the fog layer's central controller layer.

Fog layer: This is the second layer of the proposed workflow execution model. This layer contains different fog clusters that contain various fog nodes. This layer also contains a central controller that controls the working of these fog clusters. The central controller checks the availability of fog codes in each cluster and assigns the workflow

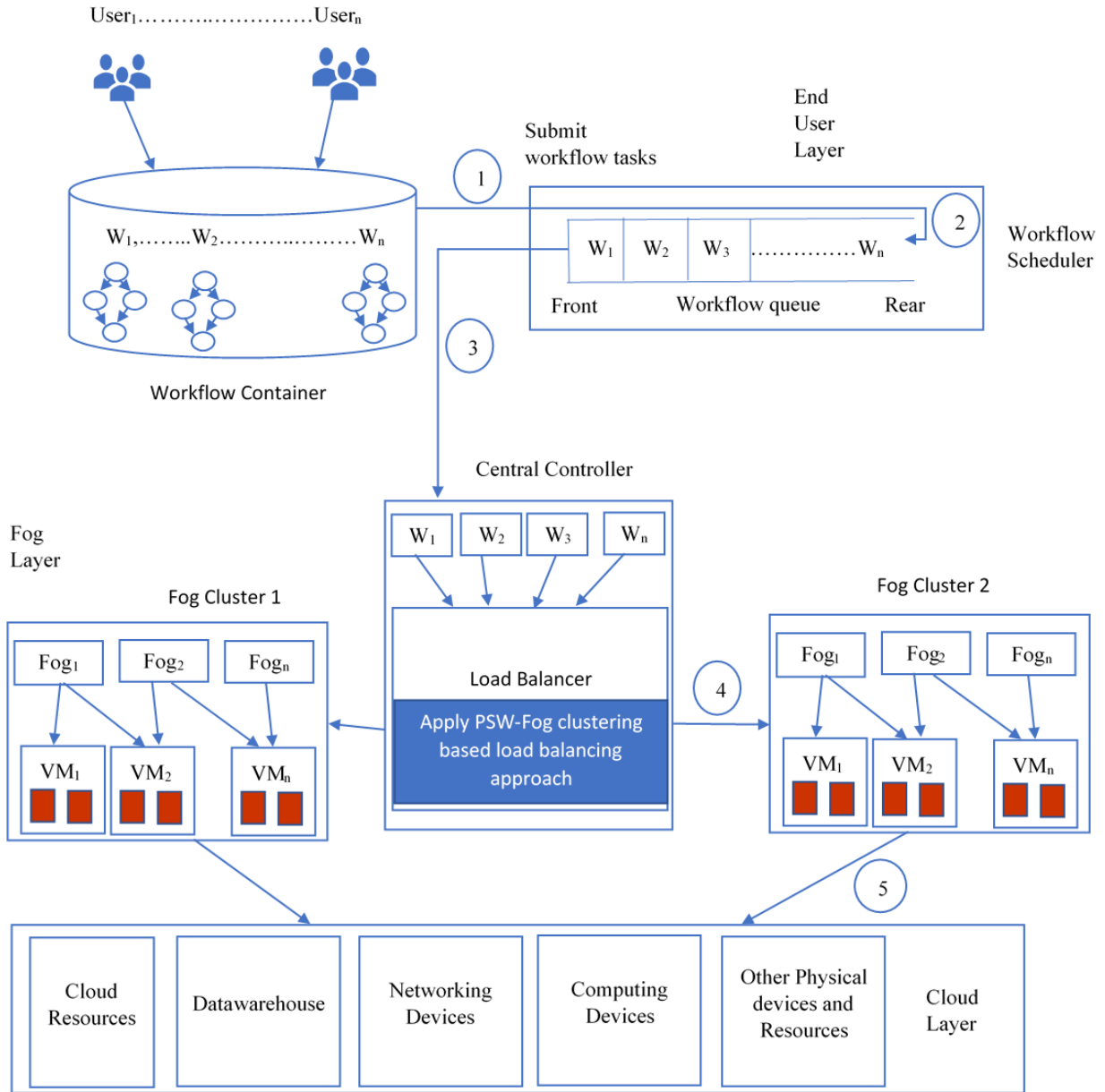


Figure 4.2: Resource-utilization based workflow execution model for fog computing

tasks to the available nodes. Working of this layer is explained in the following step:
Step 4: The central controller receives the workflow tasks from the workflow scheduler. It contains the load balancer that continuously keeps a watch on all fog nodes in all fog clusters. Here PSW-Fog clustering-based load balancing approach has been

applied to distribute the tasks among all available nodes in equal proportion. In each fog cluster, various fog nodes contain VM that execute these workflow tasks. The tasks with the highest priority are executed first. The other tasks with low priority can wait for the processor and assign them to the cloud layer for processing and further storage. The fog layer executes real-time tasks that are having limited time and require an immediate processor. Load balancer assigns such tasks to the available VMs, and users respond after processing tasks.

Cloud layer: This is the third layer of the proposed workflow execution model. This layer contains large data centers that can store and process a large amount of data. Further, this layer includes various computing, networking, and storage resources.

Step 5: After processing at fog layer, workflow tasks processing results are informed to users, and these tasks are further sent to the cloud layer for storage and more processing if required.

The proposed workflow execution model tries to reduce the execution time of resources. With the reduction of execution time, the number of resources needed will also reduce energy consumption. So, our proposed solution can enhance maximum resource utilization and minimize energy consumption.

4.3 PSW-Fog clustering-based load balancing algorithm

In this section, a Fog-Clustering-based load balancing algorithm has been proposed for executing workflow datasets. While executing workflow datasets, a load balancing problem is raised due to the fog computing layer's lesser storage and computing capacity. Due to this, a few VM in the fog layer becomes overloaded with the tasks, and others still wait for the tasks. Along with this, VMs consume more energy even if they are free. An efficient load balancing approach is required to reduce the overloading of VMs. This section contains the methodology and algorithm proposed in this work. This section includes three subsections, i.e., proposed methodology, proposed algorithm, and performance metrics.

4.3.1 Optimization approaches used in our proposed hybrid algorithm

Various meta-heuristic optimization approaches are available to find near-optimal solutions to solve complex computational problems that can not be solved using a single method. Using a single approach may not find the required optimum solution within the defined problem's time constraints. There are many natural phenomena available that different researchers use to solve many various optimization problems. This research work tried solving the load balancing problem faced in fog computing during the execution of sizeable scientific workflow computations. So, only one single optimization approach may not be able to find one optimum solution. Hence, this work combined different natural phenomenon-based optimization approaches, i.e., plant growth optimization, simulated annealing algorithm, and water cycle optimization. All these approaches are explained in this section. Here table 4.1 represents different notations used in this chapter.

Table 4.1: Notations

Notations	Description
M_i	Optimized threshold at i th iteration
$f(x_i)$	Distance function
P	Prediction of task mapping
e	Current resource allocation
C_i	Population of clusters
NS_n	Optimize cluster mapping by Ncluster
C_n	Fog nodes in cluster
C_i	Average value of fog nodes in cluster

continued on next page

continued from previous page

Notations	Description
F_n	Fog nodes
$N_{clusters}$	Random cluster of fog nodes
S_n	Simulated annealing
N_{pop}	N population
T_i	Execution time
P_i	Computation time
E_{En}^{Fog}	Energy consumption
C_{Cn}^{Fog}	Forwarding tasks
C_{Pc}^{Fog}	Computational Tasks
∂	Clustering threshold

Plant Growth Optimization(PGO)

This algorithm is proposed to simulate the real way of a plant's growth by considering its branching, leaves growth, and phototropism. PGO varies from the natural plant-growth process to consider two kinds of behaviors to find the optimal solution. Firstly it produces new branching points to find the optimal solution. Secondly, it considers the new growing leaves around branches to find accurate solution [173] [174]. This research considered the PGO approach in the proposed hybrid approach to select cluster heads in different fog-nodes clusters. PGO is considered because of the need for less variance because cluster size variance increases by default that causes load imbalance in nodes. The PGO is based upon a real tree's growth in which the trunk grows from roots and branches grow from the branches. This process goes on, and some new more branches grow from the nodes of branches. The same process continues till the tree is formed. The plant growth process has been followed to develop an optimization approach in which optimization starts from the plant's root and keeps growing till branches until

the best solution is found [175] [176]. The following equation can be used to find an optimum solution using PGO.

$$M_i = \left\{ 1 - \frac{f(x_i) - f_{\min}}{\sum_{j=1}^N [f(x_j) - f_{\min}]} \right\} + \left\{ \alpha_{i(t)} * \left(P_i - \sum Q^i \right) * \beta_i^{cl} \right\} \quad (4.1)$$

In eq (1) M_i optimize threshold at i^{th} iteration its depend on fog nodes distance function $f(x_i)$ its Euclidian distance. With minimum value of threshold maximum chance on same cluster.

Simulated Annealing Algorithm(SAA)

The SAA was introduced by [177] in 1983 based on metal annealing's chemical process. The process of SAA starts with a random answer, and then the neighbor solution is found. SAA is generally used for single solution problems [178]. Many researchers have used SAA to solve different NP-hard problems and solved large computational tasks [178–180] In our proposed approach, fog nodes have been divided into clusters. After selecting the cluster head using the PGO algorithm, workflow tasks are mapped onto clusters. So, this work used SAA for intracluster mapping. SAA is used to analyze all cluster resources and their energy consumption. The mapping of tasks can be predicted using the current allocation of resources. SAA helps to find a global solution for the NP-hard problem by having a large error margin. The margin error can be defined as the acceptance probability that can be considered to predict the task mapping on cluster resources. The following formula has been used to predict task mapping in clusters.

$$P = \left\{ \frac{1}{1 + e^{-\frac{\Delta(Cur_i)}{I}}} \right\} \quad (4.2)$$

In eq(2) P is the mapping task prediction of task mapping on fog cluster that is calculated by $e^{-\frac{\Delta(Cur_i)}{I}}$ current allocation resources, resources utilization reduce then task mapping will increase because of prediction value increase.

Water Cycle Optimization(WCO)

The main idea behind WCO is taken by the observation of the natural water cycle process that describes how all water from streams and rivers flow into the sea [181]. The water cycle describes how rivers and streams are formed by receiving water from the melting of glaciers or heavy rains, and then this water goes into the sea. This basic concept of the natural water cycle has been used by different researchers to implement optimization to solve various computations [182]. When SAA could not find the optimized solution, WCO is used to maximize the energy and cost of resources between inter-cluster resource mapping. Firstly, the population in clusters is defined on which tasks to be mapped. The following equation describes the population of clusters.

$$C_i = f(x_1^i, x_2^i, x_3^i, \dots, x_n^i) \text{ Where } i = 1, 2, 3, \dots, N_{pop} \quad (4.3)$$

In eq(3), C_i shows the population of clusters on which task is mapped. Here N is number of design variables. A matrix shows the initial population of size N_{pop}

$$NS_n = \text{round} \left\{ \left| \frac{C_n}{\sum_{i=1}^n C_i} \right|_{cluster} n = 1, 2, 3, \dots, N_{cluster} \right. \quad (4.4)$$

In eq (4), NS_n optimize the task mapping on fog clusters by WCA objective function which optimizes cluster mapping by number of cluster resources $N_{cluster}$. C_n is number of fog nodes in a cluster and $\sum_{i=1}^n C_i$ is average value of normalize cluster fog nodes.

4.3.2 PSW-Fog clustering algorithm

This section proposes a PSW-fog clustering-based load balancing algorithm for scientific workflow applications. The heuristic optimization algorithms explained in section 4.1 have been used in a hybrid form to create a PSW-fog clustering-based load balancing algorithm. These approaches are combined to find the best optimal solution without wasting execution time and energy of resources. In the algorithm, 3 number of fog nodes and cloud resources are considered input. The desired output of the algorithm is to optimize load balancing on fog nodes. Available fog nodes are combined

in the form of fog clusters. The PGO algorithm is applied to find the cluster head for each cluster. Here population in each cluster is defined, and the fitness function takes two different parameters, i.e., energy and cost. The convergence is applied according to both these parameters. If converged, then select cluster head. Now, after the selection of each cluster head, tasks are mapped onto resources by using SAA. If optimized, then computing parameters are analyzed. If not optimized, task migration is applied to balance each resource and WCO using the same computing parameters. Now here population for each cluster is defined, and an optimized solution for cluster mapping is found. If optimized, then computing parameters are analyzed; otherwise, again, WCO is applied to optimize. These approaches can not provide an optimal solution for load balancing in fog computing if used alone. This work has done hybridization of all the explained techniques to enhance resource utilization in a fog environment. When all the resources are utilized reasonably, it will reduce the time delay processing the tasks. Energy consumption in these nodes will automatically be reduced, which will reduce the computational cost.

4.3.3 Flow of execution of proposed algorithm

This section explains the proposed algorithm in the form of a flow chart that explains the proposed algorithm's working. The methodology used in this algorithm is divided into three parts, i.e., making fog nodes clusters, initial task mapping, and optimize mapping among clusters. All these steps are explained as follows:

- 1. Making fog nodes cluster:** In this step, firstly, fog nodes are created into a group of fog nodes, and then a cluster head is selected. The plant growth (PGO) approach takes this decision. This approach uses two parameters for optimization, i.e., energy and resource cost. If both parameters are optimized, make a cluster and select cluster head. Plant growth optimization is used because of the need for less variance of the group because cluster size variance increases by default that causes load imbalance.
- 2. Task initial mapping:** After selecting optimize clusters and cluster head, parse the workflow and map task on fog groups. Here the different combination of fog nodes has been made on which different tasks are mapped. Here simulated annealing (SAA) has been initiated according to fog clusters that take intra-cluster mapping decisions.

Algorithm 3: PSW-Fog Clustering based load balancing algorithm

Input: Number of fog nodes and resources of cloud

Output: Optimize Load balancing on fog nodes

```

1  $F_n \leftarrow$  Fog nodes
2  $N_{clusters} \leftarrow$  Random Cluster of Fog nodes
3  $PG =$  plant growth( $N_{cluster}, F_n$ )
4 for 1 to  $N_{cluster}$  do
5 Apply  $M_i = \begin{cases} 1 - \frac{f(x_i) - f_{\min}}{\sum_{j=1}^N [f(x_j) - f_{\min}]}, & f(x_i) > f_{\min} \\ 1, & f(x_i) = f_{\min} \end{cases} \quad i = 1, 2, 3, \dots, N$  and
   Calculate  $M_i(\text{ClusteringThreshold})$ 
6 if  $Cluster_{th} > M_i$  then
7 Begin
8  $Cluster \leftarrow F_n$ 
9 else
10  $Cluster_{i+1} \leftarrow F_n$ 
11 EndIf
12 Stop
13  $S_n =$  simulated annealing ( $Cluster_i, F_n$ )
14 for 1 to  $Cluster_i$  do
15 Start
16 Map task by  $P = \frac{1}{\left(1 + e^{-\frac{\Delta(Cur_i)}{T}}\right)}$ 
17 Stop
18 if (optimize) then
19 Begin
20 Analysis of Task computing Parameters
21 else
22 Apply WCO (tasks,  $cluster_i, F_n$ )
23 For every cluster define population by  $C_i = f(x_1^i, x_2^i, x_3^i, \dots, x_n^i) \quad i=1, 2, 3, \dots, N_{pop}$ 
24 Define Cost by  $NS_n = \text{round} \left\{ \left| \frac{C_n}{\sum_{i=1}^n C_i} \right|_{cluster} \quad n = 1, 2, 3, \dots, N_{cluster} \right.$ 
25 if  $\lfloor NS \rfloor_n < \min(NS_n)$  then
26 Begin
27 Computation of tasks on VM
28 Analysis of Task computing Parameters
29 else
30 Go to step 22
31 End
32 End

```

After simulated annealing analysis of all cluster node resources, map tasks on nodes and analyze performance parameters, i.e., energy, cost, and time delay. If performance parameters are optimized, run the computation and again analyze parameters. Otherwise, go to the next step.

3. Optimize mapping among clusters: This step comes into the picture when the previous step does not optimize. If the previous step fails to optimize all considered parameters, there is still load imbalance in the fog layer, hence needing load balancing. So task migration is applied to the clusters, and tasks from heavily loaded clusters are taken and transferred to lightly loaded clusters. In this step, water cycle optimization has been used by considering all performance metrics. The WCO reduces the energy consumption and cost of resources between inter-cluster. If all parameters are optimized, then all the parameters can be analyzed to check the performance of the proposed approach; otherwise, the loop continues with task migration on clusters. Figure 4.3 represents the flow of execution in the PSW-Fog clustering-based load balancing algorithm. The Figure 4.3 is divided into three steps explained above in this section.

4.3.4 Performance metrics

The proposed algorithm tries to reduce the execution time of tasks in fog clusters to reduce time delay. Along with this computational cost, and energy consumption in fog nodes has been reduced. All the considered computational parameters have been described as follows.

Time delay:

Time delay in fog environment can be considered as the time interval between the submission of the task to the response after processing of that task. The time delay depends on computational time; if the computation time is less, then the time delay will be lower. The fog layer mainly tries to reduce the time delay for processing tasks near end users. The following equation describes the time delay calculated in the proposed

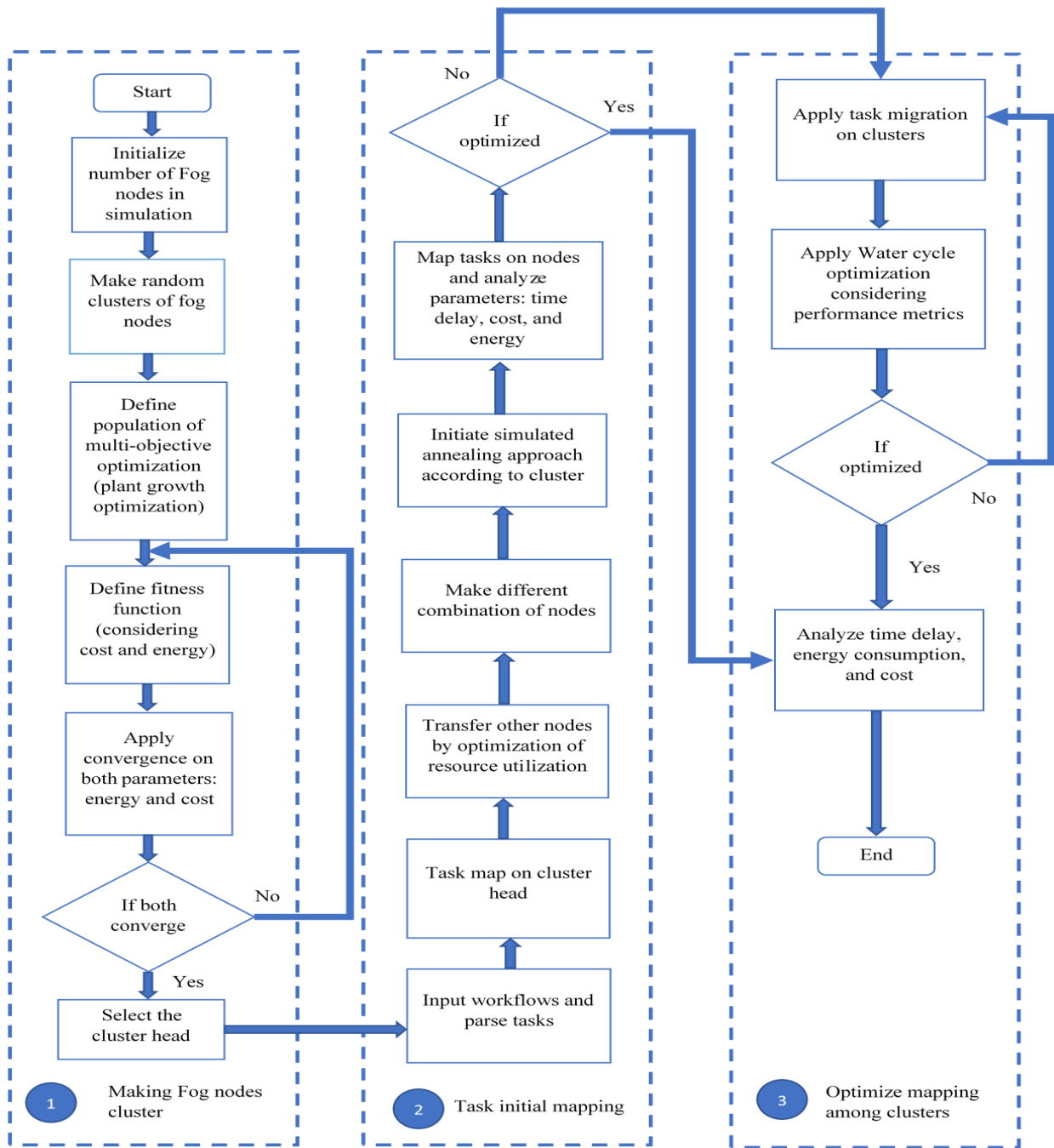


Figure 4.3: Flow of execution of PSW-Fog clustering based load balancing algorithm

algorithm.

$$T_i = \left\{ 1 - \frac{f(x_i) - f_{\min}}{\sum_{j=1}^N [f(x_j) - f_{\min}]} + \left\{ \alpha_{i(t)} * \left(P_i - \sum Q^i \right) * \beta_i^{cl} \right. \right. \quad (4.5)$$

In eq (4.6) First part of latency depend on objective function of plant growth optimization and second show the latency calculation by fog nodes and cloud respectively but fog nodes cluster and temporary storage improve the computation time P_i and $\sum Q_i$ show the resources and these are dependent on two hyper parameters β_i^{cl} and $\alpha_{i(t)}$.

Cost

The computational cost in a fog environment can be considered in the form of the maintenance cost and energy consumption in fog nodes [150]. If only a few resources are utilized, and others remain underutilized, they must be adequately maintained. Along with fog resources, cloud resources also compute tasks that also have some computational cost. The following equation is used to calculate the computational cost in a fog-cloud environment.

$$C_{Pc}^{cost} = \left\{ Cluster_i . NS_n (1 - \beta) * (\alpha_{i(t)} - \alpha_{i(t-1)}) + C_{Cn}^{cloud} \right. \quad (4.6)$$

In eq (5) C_{Pc}^{cost} represents the analysis of computation task in Cloud. Computation depends on different parameters where $NS_n (1 - \beta)$ N_V is virtual machine mapping with task migration of cloud fog nodes, and $(\alpha_{i(t)} - \alpha_{i(t-1)})$ is number of computation resources available. If this quantity increases then computation cost also increases. C_{Cn}^{cloud} also depend on resources of cloud.

Energy

The fog layer contains various devices such as gateways, routers, servers that consume much energy while executing large computational tasks. While migrating tasks between clusters, energy consumption in nodes is also increased. Hence maximum load

balancing in fog nodes can help to reduce energy consumption in the fog environment. The following equation represents the energy consumption in a fog environment.

$$E_{En}^{Fog} = \left\{ \partial * \sum_{i=1}^N C_{Cn}^{Fog} + (1 - \partial) * \sum_{i=1}^N C_{Pc}^{Fog} + C_{Cn}^{cloud} \right. \quad (4.7)$$

Equation (7) shows the E_{En}^{Fog} Energy consumption which depends on two factors: first is C_{Cn}^{Fog} forwarding tasks, and second is C_{Pc}^{Fog} Computation tasks. Here ∂ is clustering threshold.

4.4 Analysis of PSW-Fog clustering-based load balancing approach

This section evaluates the proposed PSW-Fog clustering approach by executing scientific workflow applications. For simulation purposes, iFogSim collaborates with CloudSim because of its huge library of resource management and cloud environment simulation. CloudSim handles all the events occurring between different fog components. In the experimental requirement table 4.2, it has been explained that 20 to 200 fog nodes have been considered, which makes fog nodes clusters. Approximately 20 fog clusters have been considered that contain 10 to 20 VM per cluster. Fog nodes have less computing capacity, so the fog layer is further connected to the cloud layer having a large data center. The fog layer works in collaboration with the cloud layer. The fog layer processes the tasks having fewer time requirements, and other tasks with low priority are sent to the cloud layer. This research is carried out experiment by considering benchmark workflows such as GENOME, Cybershake [172], SIPHT [122], and LIGO [122]. This research compared its results with other existing approaches, i.e., Artificial Bee Colony (ABC), Tabu search [114] [161], Ant Colony Optimization(ACO) [81] [162], Grey Wolf Optimization(GWO) [163], and Tabu-GWO-ACO approaches, and it has been represented through graphs that proposed approach outperforms than all other optimization approaches.

4.4.1 Experimental Requirement

This research work has considered few experimental requirements to obtain simulation results. The fog layer has been divided into clusters, each containing a maximum of 20 nodes. Table 4.2 represents the experimental requirements used for the evaluation of the proposed PSW-Fog clustering-based load balancing in fog environment.

iFogSim has been considered because of its open-source availability. It is a high-performance toolkit used for fog and edge computing environments. It is used to simulate fog computing and IoT networks. iFogSim works with CloudSim in collaboration. iFogSim contains three components, i.e., physical components containing physical fog nodes, logical elements containing different application modules and application edges, and lastly, management components that include module mapping objects and fog controller [164].

Table 4.2: Experimental Requirement

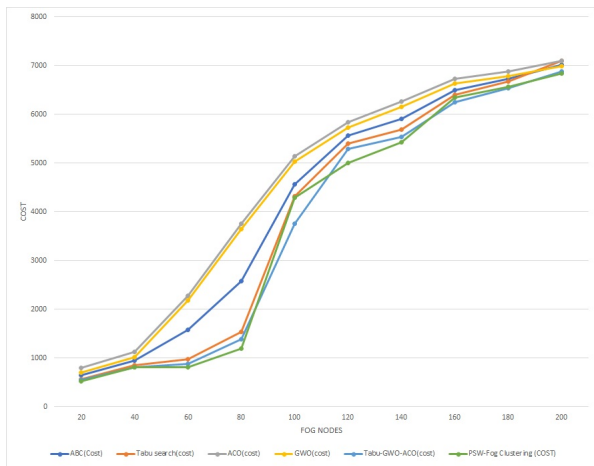
Experiments Parameters	Value/Name
Simulator	CloudSim and iFogSim
Dataset	Workflows (LIGO, SIPHT, GENOME, Cybershake)
Datacenter	One
VM	10 to 20
Fog Nodes	20 to 200
Optimization algorithm	simulated annealing, plant growth, WCA
Processor Per VM	0.5MIPS
Memory per VM	200MB
Max Clusters	20
Max fog nodes in Cluster	20

4.4.2 Result analysis

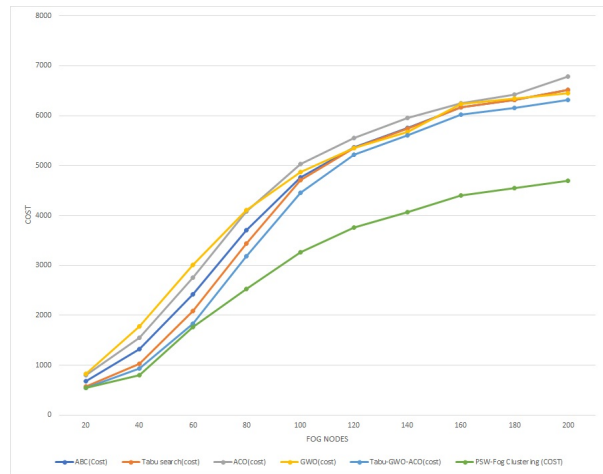
This section covers the simulation results obtained from iFogSim after evaluating the proposed approach PSW-Fog clustering-based load balancing. These results have been compared based on time delay, cost, and energy consumption in a fog environment. This work has considered datasets of LIGO, Genome, SIPHT, and cybershake. The obtained results are shown in graphs that compare the proposed approach with other existing techniques.

Cost analysis

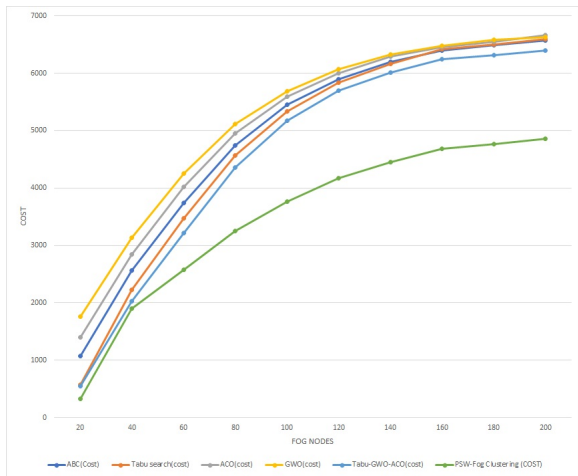
The considered workflows are executed using the proposed approach, and their computational cost has been analyzed. The workflows are taken in size of 20 to 100 tasks, and they are executed on fog nodes considering 20 to 200 fog nodes as represented in graphs in Figure 4.4. The graphs represent variations in cost during execution on a varying number of fog nodes. It can be seen in graphs that with an increase in the number of fog nodes, computational cost also increased in every experiment. With our proposed multi-objective optimization approach, This research has improved cost reduction compared to other existing heuristic approaches. Figure 4.4(a) shows the computational cost while evaluating LIGO workflow. It can be seen from the graph that the proposed approach significantly reduces the computational cost in fog nodes during execution on the different number of fog nodes from 20-200. Our proposed approach overlaps the cost values of other existing approaches in some experiments, but its average value improves the cost reduction. The improvement in cost reduction in the proposed approach is due to the clustering of fog nodes. Workflows are executed in distributed resources that reduce cost and other parameters. It has been obtained that after comparing with other approaches, PWS-fog clustering tries to reduce cost by 25% in the case of LIGO. In Figure 4.4(b), variation in cost can be seen during the evaluation of cybershake workflow, which clearly shows the reduction in cost by the proposed approach. The mean value of cost reduction in each experiment has been improved by PSW-Fog clustering. This reduction in cost in the case of cybershake is its lesser complexity compared to LIGO. In the experimental analysis, it has been obtained that PSW-Fog based approach tries to reduce 45% as compared to the average



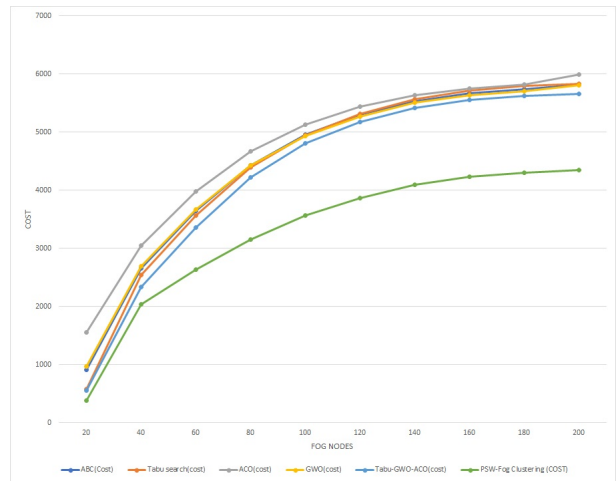
(a) LIGO



(b) Cybershake



(c) Genome



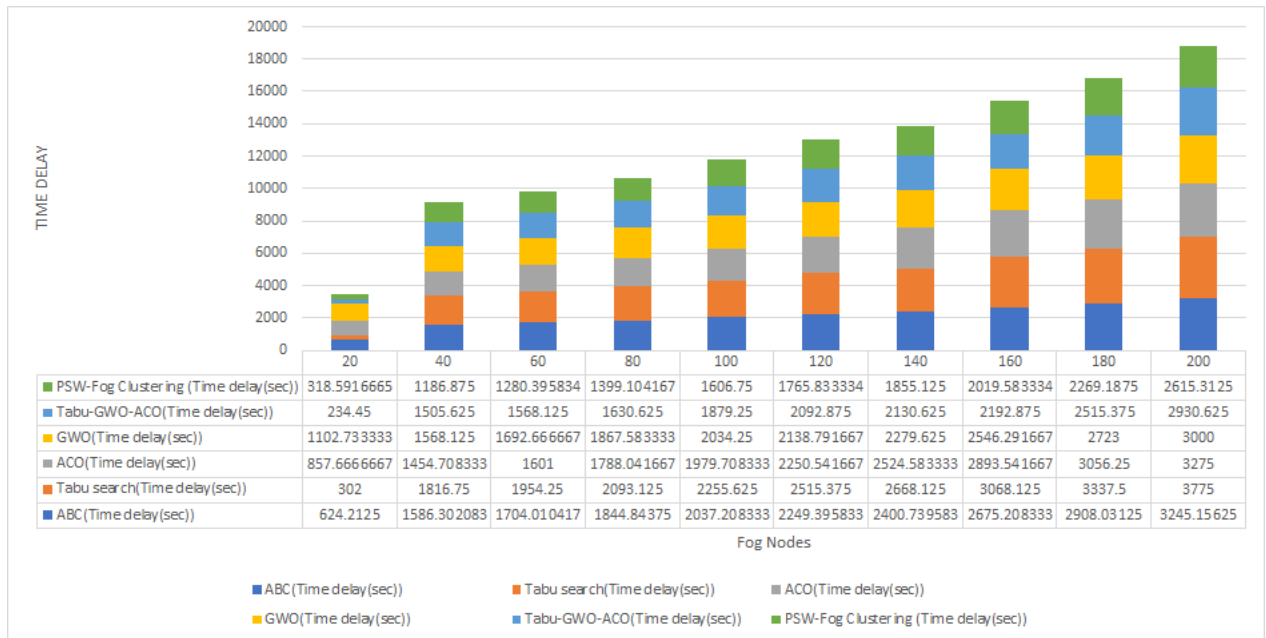
(d) SIPHT

Figure 4.4: Cost Analysis of different workflows(PSW-Fog clustering Approach)

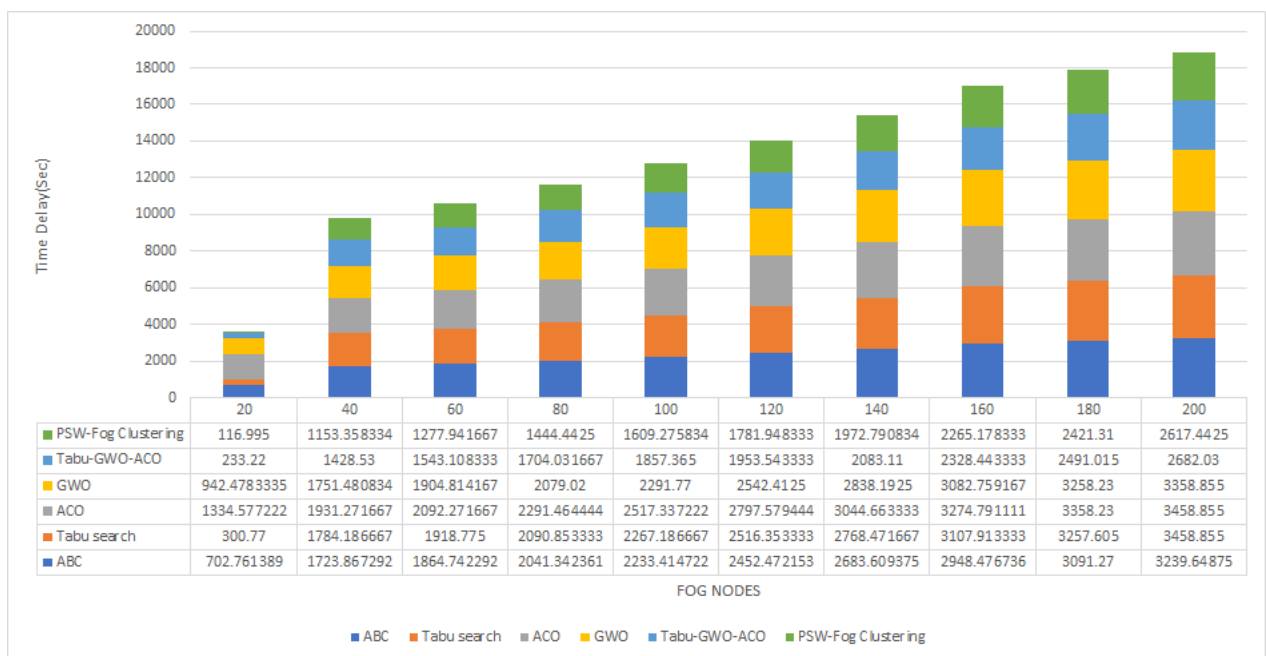
of all other considered approaches. Same as LIGO and cybershake, other workflows GENOME and SIPHT are also executed. Figure 4.4(c) and Figure 4.4(d) shows the improvement in cost reduction to 35%, 40% in the case of GENOME and SIPHT, respectively. Both the graphs show a significant reduction in computational cost in all experiments. This cost reduction in GENOME and SIPHT is due to optimization in the clustering of fog nodes and the migration of tasks between the optimized cluster.

Time delay analysis

Time delay has been evaluated and represented in the form of bar graphs. Figure 4.5 has been divided into two parts, out of which Figure 4.5(a) represents the time delay in evaluating LIGO workflows. Our proposed approach reduces time delay in executing workflow tasks on fog nodes compared to other existing techniques. As shown in the LIGO graph, when 20 fog nodes are considered, the time delay overlaps the different existing approaches. However, the average time delay has been reduced in other experiments by considering 40-200 fog nodes. Fog nodes have been grouped into clusters that help in reducing the execution time of tasks. Due to distributed nature of fog computing, all the resources are distributed in available fog nodes clusters that help execute large workflow tasks, i.e., 20-100 tasks per cluster. Time delay is measured in seconds. The X-axis of graphs represents the number of fog nodes considered for experiments, and the y-axis represents the time delay in seconds.



(a) LIGO

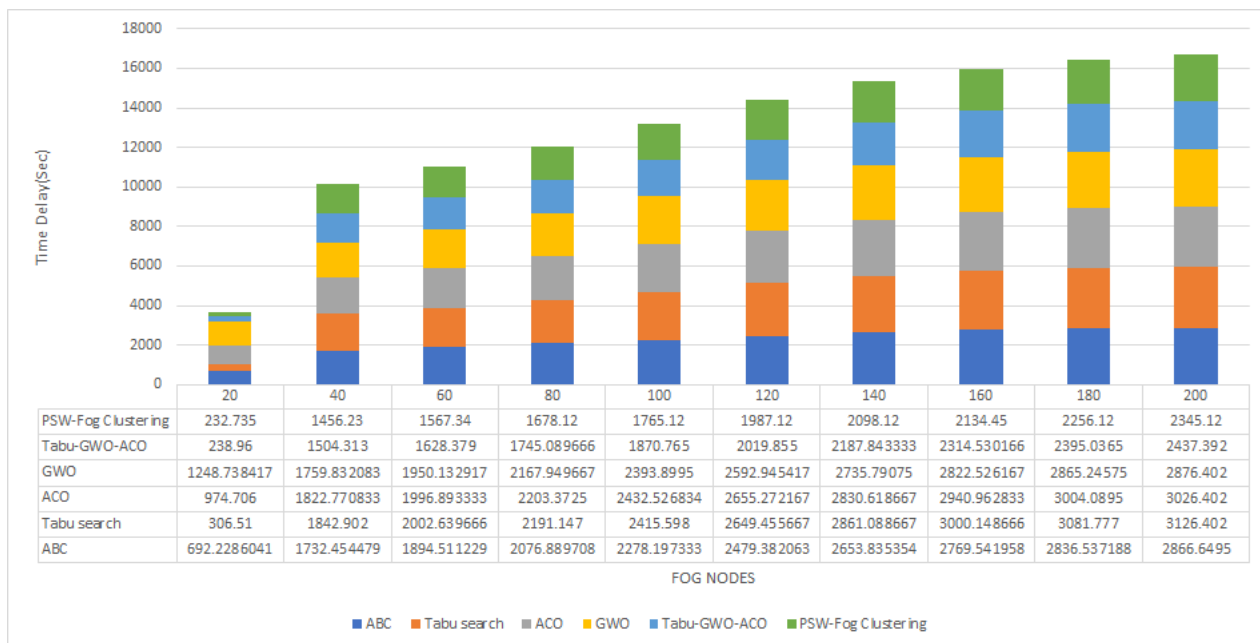


(b) Cybershake

Figure 4.5: Time delay analysis of LIGO and Cybershake workflows (PSW-Fog clustering)



(a) Genome



(b) SIPHT

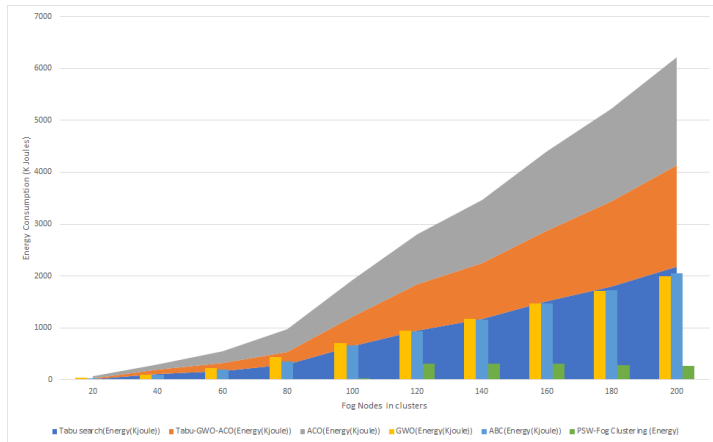
Figure 4.6: Time delay analysis of Genome and SIPHT workflows(PSW-Fog clustering Approach)

Figure 4.5(b) shows time delay calculated while executing cybershake workflow that indicates that the proposed approach reduces time delay as compared to other considered approaches. Due to the low complexity of cybershake, the mean value of time delay has been reduced in every experiment.

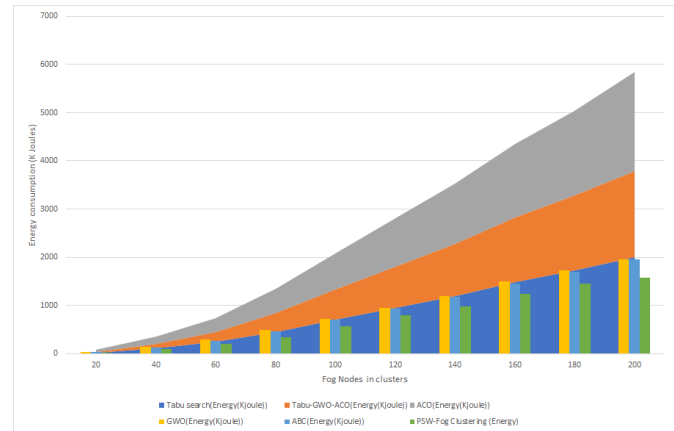
Similar improvement trends have been observed in the case of GENOME and SIPHT that are represented in Figure 4.6(a) and 4.6(b), respectively. Figures represent execution time of GENOME and SIPHT that shows significant improvement in all experiments. It has been obtained that while considering 20 to 200 fog nodes in experiments, time delay has been reduced to 30%, 50%, 40%, and 45% in case of LIGO, cybershake, Genome, and SIPHT, respectively.

Energy-consumption analysis

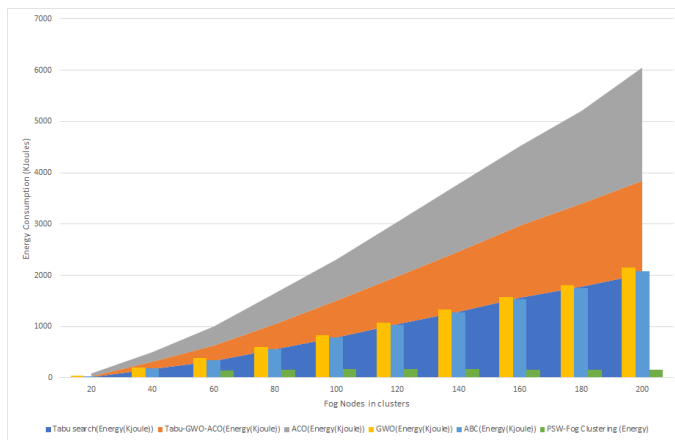
A similar kind of experiment has been conducted to analyze energy consumption in fog nodes. This research has considered the same number of fog nodes like cost and time analysis, i.e., 20 to 200, for conducting experiments. This research has conducted approximately ten experiments considering 20,40,60,.....,200 fog nodes in every experiment. Figure 4.7(a) shows the energy consumption in the case of LIGO. It can be seen from Figure 4.7(a) that PSW-Fog based load balancing approach reduces energy consumption to 60% compared to other existing approaches. Clustering fog nodes reduces the intra-cluster task migration, reducing energy consumption during the migration of tasks between nodes. Figure 4.7(b) represents energy consumption in cybershake, which shows a 70% reduction in energy consumption with the proposed approach. Similarly Figure 4.7(c) and 4.7(d) shows energy consumption in GENOME and SIPHT respectively. Both the Figures 4.7(c) and 4.7(d) represents the reduction in energy consumption to 50%, and 45% in case of PSW-fog clustering-based load balancing approach in case of all the experiments. This improvement is due to optimization in the clustering of fog nodes. The proposed model and approach can be implement in real time applications that are further described in the further section. Energy consumption in real applications is the main issue now days that is further required to be worked upon. In future works, real time application will be implemented and energy consumption in real time application will be tried to be reduced.



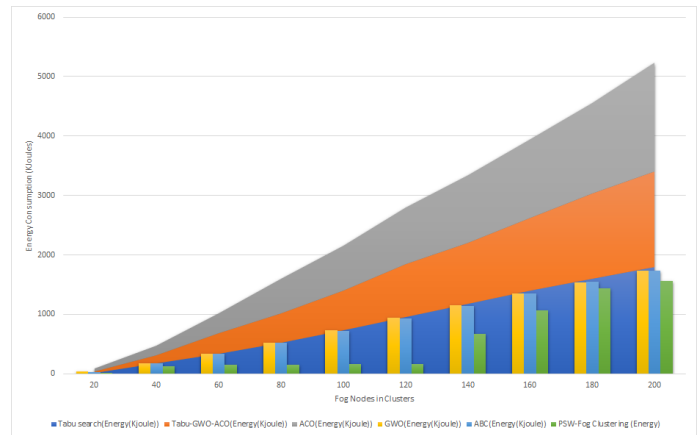
(a) LIGO



(b) Cybershake



(c) Genome



(d) SIPHT

Figure 4.7: Energy Analysis PSW-Fog clustering Approach

4.4.3 Applications of proposed energy efficient load balancing algorithm

- **Home energy management** Traditionally energy was generated by burning fossil fuels which increases human life threats, increased carbon emission and global warming as well. The other source of energy production is solar system and windmills. In this modern computing era, smart systems exist like smart grids which help in green power distribution, power usage control, improves the load. These day smart meters are installed in smart grids which help to keep record of power consumption and also

control the power consumption. These smart meters generate a large volume of data which is stored and processed by cloud datacenters. But due to high latency problem with cloud, data is being processed by fog computing layer in between the cloud and smart grids (IoT layer). Fog nodes can be deployed in the homes which keep record of on/off electric devices. If the load on smart grids is more, then fog nodes automatically switch off the unused appliances. This helps to save energy consumption in smart homes.

- **Smart traffic management system** With the growth of IoT and fog computing Internet of Vehicles (IoV) came into existence. Due to driving mistakes of humans, there are chances of accidents. IoV can help to reduce accidents and reduce congestion in the city. IoV has transformed the moving vehicles into intelligent electric vehicles, which helps to reduce latency as well as energy consumption. Due to real time interconnections between vehicle to vehicle, vehicle to sensors and vehicle to computing framework, fog nodes need to process immediately to avoid any congestion [183]. In the IoV, a large volume of data is generated intelligent electrical vehicles containing sensors, which need to be distributed among all nearby fog nodes for immediate decision making as well as network management. The smart vehicles can be informed by smart traffic lights about any congestion or road-holes in their path and these vehicles can change their routes and will further send signals to the vehicles coming behind them. In this way traffic is re-routed to avoid any congestion or accident. Along with this demand for reducing the energy consumption by Road Side Units(RSU) and electrical vehicles is also increasing. As RSUs are having installed batteries which consumes a large amount of energy while interacting with electric vehicles and other RSUs. But there is less work provided in energy conservation in moving vehicles, which is compulsory to conserve natural energy resources and improve the computing process in IoV. There is a need to provide energy aware computing in fog environment for smart traffic management system [184].

These applications can be further implemented using proposed approach to reduce energy consumption in real time environment. As, the power consumption has become a big issue now days that need to be further worked upon.

4.5 Conclusion

Load balancing in scientific workflows is necessary to utilize the resources at the fog layer fully. This chapter provides the architecture for fog computing implementing load balancing in a scientific workflow. Furthermore, this chapter reviews the existing load balancing and scheduling techniques in workflows and briefly reviews them. Furthermore, the PSW-fog clustering-based load balancing algorithm has been proposed by amalgamating different existing heuristic approaches, i.e., plant growth, simulated annealing, and water cycle algorithms. Different types of existing scientific workflow examples have been described to implement in a fog environment using the proposed approach. This research work has considered three different computing parameters used to check the performance of the proposed approach, i.e., time delay, cost, and energy consumption. In order to evaluate the proposed approach, iFogSim has been used to find the simulation results. The results obtained by executing PSW-fog clustering are compared with other existing approaches. It can be seen from graphs that our proposed solution tries to improve than other approaches. PSW-fog clustering tries to reduce time delay, computational cost, and energy consumption in fog nodes.

CHAPTER 5

CONCLUSION AND FUTURE WORK

The main contribution of the thesis is concluded in this chapter. A thorough study is done in fog computing, and its architecture, applications, open issues, and research challenges have been discussed. Fog mainly provides services of cloud computing near the edge of the network. The main issue of latency faced by cloud computing has been resolved by fog computing by deploying its nodes near to network edge. Due to large and complex computational tasks, fog computing faces the problem of overloaded resources. Different issues caused by overloaded resources have been identified. Various existing approaches have been studied, and a thorough literature review has been provided. This research proposed fog computing architecture of load balancing for scientific workflow application to overcome overloaded resources in the fog computing environment and named it FOCALB.

Further, to evaluate and analyze the proposed framework, a hybrid load balancing algorithm has been proposed (Tab-GWO-ACO). Further, a resource-utilization-based Workflow execution model for fog computing has been proposed to reduce energy consumption in a fog environment. PWS-Fog based load balancing approach has been proposed to analyze the proposed model. The proposed approaches have been analyzed, their experimental requirements have been explained, and the obtained simulation results are compared with existing approaches. The results obtained from the simulation show that the proposed techniques outperform the existing methods.

This chapter concludes the research work done in this thesis by highlighting its main contributions. The chapter begins with the conclusion of the research work conducted in this thesis by briefing each and every chapter. Further, it discussed load balancing and energy consumption in fog computing. The chapter provides the future scope of the

work and provides the research directions for future researchers.

5.1 Conclusion

The thesis, "Energy aware load balancing in fog computing," addresses resource utilization, load balancing, and energy consumption challenges in the fog computing environment.

Chapter 1 studied fog computing, its definitions provided by different researchers, and its key areas. Further, the need for load balancing at the fog layer is discussed. Parameters that affect load balancing are discussed, and a taxonomy is provided describing existing load balancing techniques. Also, Open issues and challenges faced in fog computing environments discussed can be considered for future research. Scientific workflow applications have been described that are used to evaluate the proposed approaches in this research work.

As provided in chapter 2, a thorough literature review has been conducted in fog computing considering its challenges, i.e., load balancing, resource utilization, energy consumption. Various load balancing techniques provided by different researchers have been studied, and their comparison in the form of tables is provided. Year-wise reviews of existing approaches have been conducted, and a taxonomy showing year-wise studies has been provided. Further, the chapter concluded by formulating the problem and providing the research objectives.

The next chapter 3 proposed a load balancing architecture for the fog computing environment. The proposed framework has been divided into three layers, i.e., end-user layer, fog layer, cloud layer. Here, the fog layer has been divided into various clusters containing few fog nodes each. Fog clusters have their local controllers that keep track of each fog node in the cluster. The load balancing process is applied if there is overloading or underloading of resources. Fog's local controller transfers tasks from overloaded resources to underloaded resources. The proposed load balancing approach Tabu-GWO-ACO has been applied to avoid overloading of resources and for full utilization of resources in a fog environment. FOCALB has been implemented in simulation environment. Load balancing approach TABU-GWO-ACO has been

evaluated by using scientific workflow applications. Different application scenarios of FOCALB have been discussed at the end of the chapter. The chapter also proposed a prototype model for implementing FOCALB in real time environment i.e. Smart waste management application.

Further chapter 4 proposed a resource-utilization-based model for energy-aware load balancing in a fog environment. This model mainly works for minimizing the energy consumption in a fog environment. An energy-aware load balancing approach, i.e., PSW-Fog clustering, has been proposed for maximum resource utilization. Energy consumption in fog resources can be reduced with maximum resource utilization. All the results have been obtained by evaluating the proposed approaches in the iFogSim simulation environment using LIGO, Sipht, Cybershake, Genome workflow applications. An energy-aware load balancing approach (PWS-Fog clustering) has been executed in the proposed model by using scientific workflow applications. It has been obtained from experimental results that the proposed approaches outperform the existing load balancing approaches. At the end, the chapter also describes the real time application scenarios of proposed energy-aware load balancing approach.

The main contributions of this research work are summarised as follows:

- A thorough literature review has been conducted in load balancing in the fog computing environment.
- Resource utilization based load balancing framework has been proposed that ensures maximum utilization of available resources in fog computing
- A load balancing approach has been proposed to optimize the resources in the fog layer.
- Three main parameters have been considered to analyze the proposed resource utilization-based load balancing framework and energy-aware load balancing algorithm, i.e., Execution time, Computational cost, Energy consumption.
- This thesis presents the development and implementation of the proposed framework (FOCALB) and loads balancing approaches, i.e., Tabu-GWO-ACO and PWS-Fog clustering approaches.
- The proposed framework provides load balancing and maximizes resource utilization in the fog layer, minimizing energy consumption.

- The proposed solution for load balancing helps for efficient resource utilization.

5.2 Future enhancement

There are some open issues in fog computing that can show future directions to the researchers to explore this area more. Moreover, many open areas need to be explored more, i.e., security, resource provisioning, and energy consumption. The section provides the research directions for future researchers, and also describes how limitations of this work can be overcome in future.

Few open challenges and future directions are provided below:

- The security and privacy of fog nodes have been a big challenge nowadays. So this area can be considered for future research.
- In the case of extensive data, energy consumption in fog computing increases. Hence there is a need to evaluate the energy-aware load balancing approach in real-time fog computing applications, i.e., e-healthcare.
- There is a need for implementing load balancing and load scheduling in a real-time environment.
- All existing load balancing approaches have been executed in the simulation environment, so experimentation is needed in the real world.
- There is a need to improve the multi-objective load scheduling problem in a fog-cloud environment.
- It can not be said which load balancing approach is best to reduce cost and energy consumption in a fog environment.

5.2.1 How limitations can be overcome in future

The challenges described above can guide future researchers to explore and enhance the area of fog computing more. In the future, this research work will be extended towards other issues faced by the fog environment. All these issues can be considered in the future to explore the fog computing area. Further, the proposed approaches will

be implemented in real time environment by implementing Smart waste management system that will help to enhance cities. In the future, more load balancing approaches will be explored in fog computing and try to enhance the fog-cloud environment's performance in real-time systems. Further, case studies will be conducted in nearby city areas for implementation of smart waste management systems and overcome the problem of waste management.

References

- [1] F. Bonomi, R. Milito, P. Natarajan, and J. Zhu, “Fog computing: A platform for internet of things and analytics,” in *Big data and internet of things: A roadmap for smart environments*, pp. 169–186, Springer, 2014.
- [2] M. Peng, S. Yan, K. Zhang, and C. Wang, “Fog computing based radio access networks: Issues and challenges,” *arXiv preprint arXiv:1506.04233*, 2015.
- [3] T. H. Luan, L. Gao, Z. Li, Y. Xiang, G. Wei, and L. Sun, “Fog computing: Focusing on mobile users at the edge,” *arXiv preprint arXiv:1502.01815*, 2015.
- [4] S. Shahzadi, M. Iqbal, T. Dagiuklas, and Z. U. Qayyum, “Multi-access edge computing: open issues, challenges and future perspectives,” *Journal of Cloud Computing*, vol. 6, no. 1, pp. 1–13, 2017.
- [5] P. Varshney and Y. Simmhan, “Demystifying fog computing: Characterizing architectures, applications and abstractions,” in *2017 IEEE 1st International Conference on Fog and Edge Computing (ICFEC)*, pp. 115–124, IEEE, 2017.
- [6] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies, “The case for vm-based cloudlets in mobile computing,” *IEEE pervasive Computing*, no. 4, pp. 14–23, 2009.
- [7] F. Bonomi, R. Milito, J. Zhu, and S. A. Computing, “its role in the internet of things||,” in *Proceedings First Ed. MCC Workshop Mob. Cloud Comput., New York, NY, USA: ACM*, pp. 13–16.
- [8] S. Khan, S. Parkinson, and Y. Qin, “Fog computing security: a review of current applications and security solutions,” *Journal of Cloud Computing*, vol. 6, no. 1, p. 19, 2017.

- [9] T. Shuminoski, S. Kitanov, and T. Janevski, “Advanced qos provisioning and mobile fog computing for 5g,” *Wireless Communications and Mobile Computing*, vol. 2018, 2018.
- [10] M. R. Anawar, S. Wang, M. Azam Zia, A. K. Jadoon, U. Akram, and S. Raza, “Fog computing: an overview of big iot data analytics,” *Wireless Communications and Mobile Computing*, vol. 2018, 2018.
- [11] F. Hosseinpour, J. Plosila, and H. Tenhunen, “An approach for smart management of big data in the fog computing context,” in *2016 IEEE International Conference on Cloud Computing Technology and Science (CloudCom)*, pp. 468–471, IEEE, 2016.
- [12] B. Farahani, F. Firouzi, V. Chang, M. Badaroglu, N. Constant, and K. Mankodiya, “Towards fog-driven iot ehealth: Promises and challenges of iot in medicine and health-care,” *Future Generation Computer Systems*, vol. 78, pp. 659–676, 2018.
- [13] K. Kai, W. Cong, and L. Tao, “Fog computing for vehicular ad-hoc networks: paradigms, scenarios, and issues,” *the journal of China Universities of Posts and Telecommunications*, vol. 23, no. 2, pp. 56–96, 2016.
- [14] A. Giordano, G. Spezzano, and A. Vinci, “Smart agents and fog computing for smart city applications,” in *International Conference on Smart Cities*, pp. 137–146, Springer, 2016.
- [15] P. G. V. Naranjo, Z. Pooranian, M. Shojafar, M. Conti, and R. Buyya, “Focan: A fog-supported smart city network architecture for management of applications in the internet of everything environments,” *Journal of Parallel and Distributed Computing*, 2018.
- [16] X. Xu, S. Fu, Q. Cai, W. Tian, W. Liu, W. Dou, X. Sun, and A. X. Liu, “Dynamic resource allocation for load balancing in fog environment,” *Wireless Communications and Mobile Computing*, vol. 2018, 2018.
- [17] M. M. Mahmoud, J. J. Rodrigues, K. Saleem, J. Al-Muhtadi, N. Kumar, and V. Korotaev, “Towards energy-aware fog-enabled cloud of things for healthcare,” *Computers & Electrical Engineering*, vol. 67, pp. 58–69, 2018.

- [18] M. Aazam and E.-N. Huh, "Fog computing and smart gateway based communication for cloud of things," in *2014 International Conference on Future Internet of Things and Cloud*, pp. 464–470, IEEE, 2014.
- [19] T. Desai and J. Prajapati, "A survey of various load balancing techniques and challenges in cloud computing," *International Journal of Scientific & Technology Research*, vol. 2, no. 11, pp. 158–161, 2013.
- [20] M. M. Rathore, A. Paul, W.-H. Hong, H. Seo, I. Awan, and S. Saeed, "Exploiting iot and big data analytics: Defining smart digital city using real-time urban data," *Sustainable cities and society*, vol. 40, pp. 600–610, 2018.
- [21] N. Rathore and I. Chana, "Load balancing and job migration techniques in grid: a survey of recent trends," *Wireless personal communications*, vol. 79, no. 3, pp. 2089–2125, 2014.
- [22] J. Baliga, R. W. Ayre, K. Hinton, and R. S. Tucker, "Green cloud computing: Balancing energy in processing, storage, and transport," *Proceedings of the IEEE*, vol. 99, no. 1, pp. 149–167, 2011.
- [23] H. Atlam, R. Walters, and G. Wills, "Fog computing and the internet of things: a review," *Big Data and Cognitive Computing*, vol. 2, no. 2, p. 10, 2018.
- [24] M. Ghobaei-Arani, A. Souri, and A. A. Rahmanian, "Resource management approaches in fog computing: a comprehensive review," *Journal of Grid Computing*, pp. 1–42, 2019.
- [25] T. Huang, B. Xu, H. Cai, J. Du, K.-M. Chao, and C. Huang, "A fog computing based concept drift adaptive process mining framework for mobile apps," *Future Generation Computer Systems*, vol. 89, pp. 670–684, 2018.
- [26] B. Bhavani and H. Guruprasad, "Resource provisioning techniques in cloud computing environment: a survey," *International Journal of Research in Computer and Communication Technology*, vol. 3, no. 3, pp. 395–401, 2014.
- [27] S. S. Gill and R. Buyya, "Resource provisioning based scheduling framework for execution of heterogeneous and clustered workloads in clouds: from fundamental to automatic offering," *Journal of Grid Computing*, vol. 17, no. 3, pp. 385–417, 2019.

- [28] N. Téllez, M. Jimeno, A. Salazar, and E. Nino-Ruiz, “A tabu search method for load balancing in fog computing,” *Int. J. Artif. Intell.*, vol. 16, no. 2, 2018.
- [29] D. Puthal, R. Ranjan, A. Nanda, P. Nanda, P. P. Jayaraman, and A. Y. Zomaya, “Secure authentication and load balancing of distributed edge datacenters,” *Journal of Parallel and Distributed Computing*, vol. 124, pp. 60–69, 2019.
- [30] S. Saeedi, R. Khorsand, S. G. Bidgoli, and M. Ramezanpour, “Improved many-objective particle swarm optimization algorithm for scientific workflow scheduling in cloud computing,” *Computers & Industrial Engineering*, vol. 147, p. 106649, 2020.
- [31] P. Maechling, E. Deelman, L. Zhao, R. Graves, G. Mehta, N. Gupta, J. Mehringer, C. Kesselman, S. Callaghan, D. Okaya, *et al.*, “Scec cybershake workflows—automating probabilistic seismic hazard analysis calculations,” in *Workflows for e-Science*, pp. 143–163, Springer, 2007.
- [32] N. A. Niemi, M. Oskin, and T. K. Rockwell, “Southern california earthquake center geologic vertical motion database,” *Geochemistry, Geophysics, Geosystems*, vol. 9, no. 7, 2008.
- [33] S. Callaghan, P. Maechling, E. Deelman, K. Vahi, G. Mehta, G. Juve, K. Milner, R. Graves, E. Field, D. Okaya, *et al.*, “Reducing time-to-solution using distributed high-throughput mega-workflows-experiences from scec cybershake,” in *2008 IEEE Fourth International Conference on eScience*, pp. 151–158, IEEE, 2008.
- [34] E. Deelman, S. Callaghan, E. Field, H. Francoeur, R. Graves, N. Gupta, V. Gupta, T. H. Jordan, C. Kesselman, P. Maechling, *et al.*, “Managing large-scale workflow execution from resource provisioning to provenance tracking: The cybershake example,” in *2006 Second IEEE International Conference on e-Science and Grid Computing (e-Science’06)*, pp. 14–14, IEEE, 2006.
- [35] H. Li, J. Ruan, and R. Durbin, “Maq: Mapping and assembly with qualities,” *Version 0.6*, vol. 3, 2008.
- [36] J. Livny, H. Teonadi, M. Livny, and M. K. Waldor, “High-throughput, kingdom-wide prediction and annotation of bacterial non-coding rnas,” *PloS one*, vol. 3, no. 9, p. e3197, 2008.

- [37] C. Team, “Dagman (directed acyclic graph manager),” *See website at <http://www.cs.wisc.edu/condor/dagman>*, 2005.
- [38] A. Kaur, P. Gupta, and M. Singh, “Hybrid balanced task clustering algorithm for scientific workflows in cloud computing,” *Scalable Computing: Practice and Experience*, vol. 20, no. 2, pp. 237–258, 2019.
- [39] D. A. Brown, P. R. Brady, A. Dietz, J. Cao, B. Johnson, and J. McNabb, “A case study on the use of workflow technologies for scientific analysis: Gravitational wave data analysis,” in *Workflows for e-Science*, pp. 39–59, Springer, 2007.
- [40] S. K. Mishra, B. Sahoo, and P. P. Parida, “Load balancing in cloud computing: A big picture,” *Journal of King Saud University-Computer and Information Sciences*, 2018.
- [41] V. Velde and B. Rama, “An advanced algorithm for load balancing in cloud computing using fuzzy technique,” in *2017 International Conference on Intelligent Computing and Control Systems (ICICCS)*, pp. 1042–1047, IEEE, 2017.
- [42] J. Rufino, M. Alam, J. Ferreira, A. Rehman, and K. F. Tsang, “Orchestration of containerized microservices for iiot using docker,” in *2017 IEEE International Conference on Industrial Technology (ICIT)*, pp. 1532–1536, IEEE, 2017.
- [43] A. Manju and S. Sumathy, “Efficient load balancing algorithm for task preprocessing in fog computing environment,” in *Smart Intelligent Computing and Applications*, pp. 291–298, Springer, 2019.
- [44] R. Buyya and S. Venugopal, “A gentle introduction to grid computing and technologies,” *database*, vol. 2, p. R3, 2005.
- [45] Y. Simmhan, “Big data and fog computing,” *arXiv preprint arXiv:1712.09552*, 2017.
- [46] S. E. Bibri, “The iot for smart sustainable cities of the future: An analytical framework for sensor-based big data applications for environmental sustainability,” *Sustainable Cities and Society*, vol. 38, pp. 230–253, 2018.
- [47] M. Chiang and T. Zhang, “Fog and iot: An overview of research opportunities,” *IEEE Internet of Things Journal*, vol. 3, no. 6, pp. 854–864, 2016.
- [48] A. Amin, S. Riyaz, A. Ali, and Z. Paul, “Review of iot data analytics using big data, fog computing and data mining,” *International Journal of Computer Science and Mobile Computing*, vol. 6, pp. 33–39, 2017.

- [49] Q. Fan and N. Ansari, "Towards workload balancing in fog computing empowered iot," *IEEE Transactions on Network Science and Engineering*, 2018.
- [50] O. Consortium *et al.*, "Openfog reference architecture for fog computing," *Architecture Working Group*, 2017.
- [51] L. M. Vaquero and L. Rodero-Merino, "Finding your way in the fog: Towards a comprehensive definition of fog computing," *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 5, pp. 27–32, 2014.
- [52] J. Letić, "Internet of things statistics for 2020 – taking things apart," 2019.
- [53] N. G., "How many iot devices are there in 2020? [all you need to know]," 2020.
- [54] M. Verma, N. Bhardwaj, and A. K. Yadav, "Real time efficient scheduling algorithm for load balancing in fog computing environment," *Int. J. Inf. Technol. Comput. Sci.*, vol. 8, no. 4, pp. 1–10, 2016.
- [55] M. Kaur and R. Aron, "Equal distribution based load balancing technique for fog-based cloud computing," in *International Conference on Artificial Intelligence: Advances and Applications 2019*, pp. 189–198, Springer, 2020.
- [56] V. Moysiadis, P. Sarigiannidis, and I. Moscholios, "Towards distributed data management in fog computing," *Wireless Communications and Mobile Computing*, vol. 2018, 2018.
- [57] P. C. Zikopoulos, C. Eaton, D. DeRoos, T. Deutsch, and G. Lapis, *Understanding big data: Analytics for enterprise class hadoop and streaming data*. Mcgraw-hill New York, 2012.
- [58] X. Jiang, P. Hu, Y. Li, C. Yuan, I. Masood, H. Jelodar, M. Rabbani, and Y. Wang, "A survey of real-time approximate nearest neighbor query over streaming data for fog computing," *Journal of Parallel and Distributed Computing*, vol. 116, pp. 50–62, 2018.
- [59] A. Nahir, A. Orda, and D. Raz, "Replication-based load balancing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 27, no. 2, pp. 494–507, 2016.
- [60] M. R. H. J. Younis and A. M. El-Halees, "Hybrid load balancing algorithm in heterogeneous cloud environment," *Hybrid Load Balancing Algorithm in Heterogeneous Cloud Environment*, vol. 5, no. 3, 2015.

- [61] H. Sharma and G. S. Sekhon, "A review on load balancing in cloud using enhanced genetic algorithm," *International Journal of Computer Engineering & Technology*, vol. 8, no. 2, 2017.
- [62] C. Dsouza, G.-J. Ahn, and M. Taguinod, "Policy-driven security management for fog computing: Preliminary framework and a case study," in *Proceedings of the 2014 IEEE 15th International Conference on Information Reuse and Integration (IEEE IRI 2014)*, pp. 16–23, IEEE, 2014.
- [63] S. Aslam and M. A. Shah, "Load balancing algorithms in cloud computing: A survey of modern techniques," in *2015 National Software Engineering Conference (NSEC)*, pp. 30–35, IEEE, 2015.
- [64] S. Rehman, N. Javaid, S. Rasheed, K. Hassan, F. Zafar, and M. Naeem, "Min-min scheduling algorithm for efficient resource distribution using cloud and fog in smart buildings," in *International Conference on Broadband and Wireless Computing, Communication and Applications*, pp. 15–27, Springer, 2018.
- [65] S.-C. Wang, K.-Q. Yan, W.-P. Liao, and S.-S. Wang, "Towards a load balancing in a three-level cloud computing network," in *2010 3rd international conference on computer science and information technology*, vol. 1, pp. 108–113, IEEE, 2010.
- [66] H. Menon, A. Bhatele, S. Fourestier, L. Kale, and F. Pellegrini, "Applying graph partitioning methods in measurement-based dynamic load balancing," tech. rep., 2015.
- [67] S. Ningning, G. Chao, A. Xingshuo, and Z. Qiang, "Fog computing dynamic load balancing mechanism based on graph repartitioning," *China Communications*, vol. 13, no. 3, pp. 156–164, 2016.
- [68] A. M. Alakeel *et al.*, "A guide to dynamic load balancing in distributed computer systems," *International Journal of Computer Science and Information Security*, vol. 10, no. 6, pp. 153–160, 2010.
- [69] S. Nazir, S. Shafiq, Z. Iqbal, M. Zeeshan, S. Tariq, and N. Javaid, "Cuckoo optimization algorithm based job scheduling using cloud and fog computing in smart grid," in *International Conference on Intelligent Networking and Collaborative Systems*, pp. 34–46, Springer, 2018.

- [70] D. Patel and A. S. Rajawat, "Efficient throttled load balancing algorithm in cloud environment," *International Journal of Modern Trends in Engineering and Research*, vol. 2, no. 03, pp. 463–480, 2015.
- [71] H. Qiao and P. Pal, "On maximum-likelihood methods for localizing more sources than sensors," *IEEE Signal Processing Letters*, vol. 24, no. 5, pp. 703–706, 2017.
- [72] Y. Yu, X. Li, and C. Qian, "Sdlb: A scalable and dynamic software load balancer for fog and mobile edge computing," in *Proceedings of the Workshop on Mobile Edge Communications*, pp. 55–60, ACM, 2017.
- [73] S. H. Abbasi, N. Javaid, M. H. Ashraf, M. Mehmood, M. Naeem, and M. Rehman, "Load stabilizing in fog computing environment using load balancing algorithm," in *International Conference on Broadband and Wireless Computing, Communication and Applications*, pp. 737–750, Springer, 2018.
- [74] M. J. Ali, N. Javaid, M. Rehman, M. U. Sharif, M. K. Khan, and H. A. Khan, "State based load balancing algorithm for smart grid energy management in fog computing," in *International Conference on Intelligent Networking and Collaborative Systems*, pp. 220–232, Springer, 2018.
- [75] W. Dou, X. Xu, X. Liu, L. T. Yang, and Y. Wen, "A resource co-allocation method for load-balance scheduling over big data platforms," *Future Generation Computer Systems*, vol. 86, pp. 1064–1075, 2018.
- [76] P. V. Krishna, "Honey bee behavior inspired load balancing of tasks in cloud computing environments," *Applied Soft Computing*, vol. 13, no. 5, pp. 2292–2303, 2013.
- [77] K. Li, G. Xu, G. Zhao, Y. Dong, and D. Wang, "Cloud task scheduling based on load balancing ant colony optimization," in *2011 Sixth Annual ChinaGrid Conference*, pp. 3–9, IEEE, 2011.
- [78] D. B. LD and P. V. Krishna, "Honey bee behavior inspired load balancing of tasks in cloud computing environments," *Applied Soft Computing*, vol. 13, no. 5, pp. 2292–2303, 2013.
- [79] I. De Falco, E. Laskowski, R. Olejnik, U. Scafuri, E. Tarantino, and M. Tudruj, "Extremal optimization applied to load balancing in execution of distributed programs," *Applied Soft Computing*, vol. 30, pp. 501–513, 2015.

- [80] K. R. Babu and P. Samuel, "Enhanced bee colony algorithm for efficient load balancing and scheduling in cloud," in *Innovations in bio-inspired computing and applications*, pp. 67–78, Springer, 2016.
- [81] M. K. Hussein and M. H. Mousa, "Efficient task offloading for iot-based applications in fog computing using ant colony optimization," *IEEE Access*, vol. 8, pp. 37191–37201, 2020.
- [82] D. C. Devi and V. R. Uthariaraj, "Load balancing in cloud computing environment using improved weighted round robin algorithm for nonpreemptive dependent tasks," *The scientific world journal*, vol. 2016, 2016.
- [83] A. Singh, D. Juneja, and M. Malhotra, "A novel agent based autonomous and service composition framework for cost optimization of resource provisioning in cloud computing," *Journal of King Saud University-Computer and Information Sciences*, vol. 29, no. 1, pp. 19–28, 2017.
- [84] M. G. R. Alam, N. H. Tran, C. T. Do, C. Pham, S. F. Abedin, A. K. Bairagi, R. Haw, and C. S. Hong, "Distributed reinforcement learning based code offloading in mobile fog," , pp. 285–287, 2014.
- [85] M. G. R. Alam, Y. K. Tun, and C. S. Hong, "Multi-agent and reinforcement learning based code offloading in mobile fog," in *2016 International Conference on Information Networking (ICOIN)*, pp. 285–290, IEEE, 2016.
- [86] T. C. Chen and C. T. Chen, "Method for configurable intelligent-agent-based wireless communication system," June 13 2000. US Patent 6,076,099.
- [87] S. Keshvadi and B. Faghieh, "A multi-agent based load balancing system in iaas cloud environment," *Int. Robot. Autom. J*, vol. 1, no. 1, 2016.
- [88] M. A. Elsharkawey and H. E. Refaat, "Mlrts: Multi-level real-time scheduling algorithm for load balancing in fog computing environment," *International Journal of Modern Education and Computer Science*, vol. 10, no. 2, p. 1, 2018.
- [89] B. Sotomayor, R. S. Montero, I. M. Llorente, and I. Foster, "Virtual infrastructure management in private and hybrid clouds," *IEEE Internet computing*, vol. 13, no. 5, pp. 14–22, 2009.

- [90] J. Wang, D. Li, and M. Y. Hu, “Fog nodes deployment based on space-time characteristics in smart factory,” *IEEE Transactions on Industrial Informatics*, 2020.
- [91] S. Mohanty, P. K. Patra, M. Ray, and S. Mohapatra, “A novel meta-heuristic approach for load balancing in cloud computing,” *International Journal of Knowledge-Based Organizations (IJKBO)*, vol. 8, no. 1, pp. 29–49, 2018.
- [92] S. A. A. Naqvi, N. Javaid, H. Butt, M. B. Kamal, A. Hamza, and M. Kashif, “Meta-heuristic optimization technique for load balancing in cloud-fog environment integrated with smart grid,” in *International Conference on Network-Based Information Systems*, pp. 700–711, Springer, 2018.
- [93] S. Verma, A. K. Yadav, D. Motwani, R. Raw, and H. K. Singh, “An efficient data replication and load balancing technique for fog computing environment,” in *2016 3rd International Conference on Computing for Sustainable Global Development (INDIA-Com)*, pp. 2888–2895, IEEE, 2016.
- [94] K. Dasgupta, B. Mandal, P. Dutta, J. K. Mandal, and S. Dam, “A genetic algorithm (ga) based load balancing strategy for cloud computing,” *Procedia Technology*, vol. 10, pp. 340–347, 2013.
- [95] A. Meftah, A. E. Youssef, and M. Zakariah, “Effect of service broker policies and load balancing algorithms on the performance of large scale internet applications in cloud datacenters,” *International journal of advanced computer science and applications*, vol. 9, no. 5, pp. 219–227, 2018.
- [96] Y. Mao, D. Ren, and X. Chen, “Adaptive load balancing algorithm based on prediction model in cloud computing,” in *Proceedings of the Second International Conference on Innovative Computing and Cloud Computing*, p. 165, ACM, 2013.
- [97] R. Beraldi, C. Canali, R. Lancellotti, and G. P. Mattia, “Distributed load balancing for heterogeneous fog computing infrastructures in smart cities,” *Pervasive and Mobile Computing*, p. 101221, 2020.
- [98] R. Beraldi, C. Canali, R. Lancellotti, and G. P. Mattia, “A random walk based load balancing algorithm for fog computing,” in *2020 Fifth International Conference on Fog and Mobile Edge Computing (FMEC)*, pp. 46–53, IEEE, 2020.

- [99] A. U. Rehman, Z. Ahmad, A. I. Jehangiri, M. A. Ala'Anzy, M. Othman, A. I. Umar, and J. Ahmad, "Dynamic energy efficient resource allocation strategy for load balancing in fog environment," *IEEE Access*, vol. 8, pp. 199829–199839, 2020.
- [100] S. P. Singh, R. Kumar, A. Sharma, and A. Nayyar, "Leveraging energy-efficient load balancing algorithms in fog computing," *Concurrency and Computation: Practice and Experience*, p. e5913, 2020.
- [101] S. P. Singh, A. Sharma, and R. Kumar, "Design and exploration of load balancers for fog computing using fuzzy logic," *Simulation Modelling Practice and Theory*, vol. 101, p. 102017, 2020.
- [102] M. Zahid, N. Javaid, K. Ansar, K. Hassan, M. K. Khan, and M. Waqas, "Hill climbing load balancing algorithm on fog computing," in *International Conference on P2P, Parallel, Grid, Cloud and Internet Computing*, pp. 238–251, Springer, 2018.
- [103] A. Chawla and N. S. Ghumman, "Package-based approach for load balancing in cloud computing," in *Big Data Analytics*, pp. 71–77, Springer, 2018.
- [104] N. Kumar and D. Shukla, "Load balancing mechanism using fuzzy row penalty method in cloud computing environment," in *Information and Communication Technology for Sustainable Development*, pp. 365–373, Springer, 2018.
- [105] 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 Transactions on Industrial Informatics*, vol. 14, no. 10, pp. 4548–4556, 2018.
- [106] F. M. Talaat, M. S. Saraya, A. I. Saleh, H. A. Ali, and S. H. Ali, "A load balancing and optimization strategy (lbos) using reinforcement learning in fog computing environment," *Journal of Ambient Intelligence and Humanized Computing*, pp. 1–16, 2020.
- [107] M. H. Shahid, A. R. Hameed, S. ul Islam, H. A. Khattak, I. U. Din, and J. J. Rodrigues, "Energy and delay efficient fog computing using caching mechanism," *Computer Communications*, 2020.
- [108] M. Bhatia, S. K. Sood, and S. Kaur, "Quantumized approach of load scheduling in fog computing environment for iot applications," *Computing*, pp. 1–19, 2020.

- [109] Y. Xie, Y. Zhu, Y. Wang, Y. Cheng, R. Xu, A. S. Sani, D. Yuan, and Y. Yang, “A novel directional and non-local-convergent particle swarm optimization based workflow scheduling in cloud–edge environment,” *Future Generation Computer Systems*, vol. 97, pp. 361–378, 2019.
- [110] C. Li, J. Tang, T. Ma, X. Yang, and Y. Luo, “Load balance based workflow job scheduling algorithm in distributed cloud,” *Journal of Network and Computer Applications*, vol. 152, p. 102518, 2020.
- [111] N. Rizvi and D. Ramesh, “Fair budget constrained workflow scheduling approach for heterogeneous clouds,” *Cluster Computing*, vol. 23, no. 4, pp. 3185–3201, 2020.
- [112] V. De Maio and D. Kimovski, “Multi-objective scheduling of extreme data scientific workflows in fog,” *Future Generation Computer Systems*, 2020.
- [113] R. Ding, X. Li, X. Liu, and J. Xu, “A cost-effective time-constrained multi-workflow scheduling strategy in fog computing,” in *International Conference on Service-Oriented Computing*, pp. 194–207, Springer, 2018.
- [114] A. Rehman, S. S. Hussain, Z. ur Rehman, S. Zia, and S. Shamsirband, “Multi-objective approach of energy efficient workflow scheduling in cloud environments,” *Concurrency and Computation: Practice and Experience*, vol. 31, no. 8, p. e4949, 2019.
- [115] X. Zhou, G. Zhang, J. Sun, J. Zhou, T. Wei, and S. Hu, “Minimizing cost and makespan for workflow scheduling in cloud using fuzzy dominance sort based heft,” *Future Generation Computer Systems*, vol. 93, pp. 278–289, 2019.
- [116] M. A. Serhani, H. T. El-Kassabi, K. Shuaib, A. N. Navaz, B. Benatallah, and A. Beheshti, “Self-adapting cloud services orchestration for fulfilling intensive sensory data-driven iot workflows,” *Future Generation Computer Systems*, 2020.
- [117] L. F. Bittencourt and E. R. M. Madeira, “Hcoc: a cost optimization algorithm for workflow scheduling in hybrid clouds,” *Journal of Internet Services and Applications*, vol. 2, no. 3, pp. 207–227, 2011.
- [118] R. Xu, Y. Wang, Y. Cheng, Y. Zhu, Y. Xie, A. S. Sani, and D. Yuan, “Improved particle swarm optimization based workflow scheduling in cloud-fog environment,” in

- International Conference on Business Process Management*, pp. 337–347, Springer, 2018.
- [119] K. J. Naik and D. H. Naik, “Minimizing deadline misses and total run-time with load balancing for a connected car systems in fog computing,” *Scalable Computing: Practice and Experience*, vol. 21, no. 1, pp. 73–84, 2020.
- [120] D. Puthal, M. S. Obaidat, P. Nanda, M. Prasad, S. P. Mohanty, and A. Y. Zomaya, “Secure and sustainable load balancing of edge data centers in fog computing,” *IEEE Communications Magazine*, vol. 56, no. 5, pp. 60–65, 2018.
- [121] G. Javadzadeh and A. M. Rahmani, “Fog computing applications in smart cities: A systematic survey,” *Wireless Networks*, vol. 26, no. 2, pp. 1433–1457, 2020.
- [122] S. Elsherbiny, E. Eldaydamony, M. Alrahmawy, and A. E. Reyad, “An extended intelligent water drops algorithm for workflow scheduling in cloud computing environment,” *Egyptian informatics journal*, vol. 19, no. 1, pp. 33–55, 2018.
- [123] S. Liao, J. Wu, S. Mumtaz, J. Li, R. Morello, and M. Guizani, “Cognitive balance for fog computing resource in internet of things: An edge learning approach,” *IEEE Transactions on Mobile Computing*, 2020.
- [124] M. Kaur and R. Aron, “Focalb: Fog computing architecture of load balancing for scientific workflow applications,” *Journal of Grid Computing*, vol. 19, no. 4, pp. 1–22, 2021.
- [125] T. Biswas, P. Kuila, and A. K. Ray, “A novel workflow scheduling with multi-criteria using particle swarm optimization for heterogeneous computing systems,” *Cluster Computing*, pp. 1–17, 2020.
- [126] R. Aron, “Resource provisioning strategy for scientific workflows in cloud computing environment,” in *Cloud Computing for Optimization: Foundations, Applications, and Challenges*, pp. 99–122, Springer, 2018.
- [127] C.-F. Lai, H.-Y. Weng, H.-Y. Chou, and Y.-M. Huang, “A novel nat-based approach for resource load balancing in fog computing architecture,” *Journal of Internet Technology*, vol. 22, no. 3, pp. 513–520, 2021.

- [128] A. Asghar, A. Abbas, H. A. Khattak, and S. U. Khan, “Fog based architecture and load balancing methodology for health monitoring systems,” *IEEE Access*, vol. 9, pp. 96189–96200, 2021.
- [129] D. Baldo, A. Mecocci, S. Parrino, G. Peruzzi, and A. Pozzebon, “A multi-layer lorawan infrastructure for smart waste management,” *Sensors*, vol. 21, no. 8, p. 2600, 2021.
- [130] M. K. Saroa and R. Aron, “Fog computing and its role in development of smart applications,” in *2018 IEEE Intl Conf on Parallel & Distributed Processing with Applications, Ubiquitous Computing & Communications, Big Data & Cloud Computing, Social Computing & Networking, Sustainable Computing & Communications (ISPA/IUCC/BDCloud/SocialCom/SustainCom)*, pp. 1120–1127, IEEE, 2018.
- [131] H. Wadhwa and R. Aron, “Fog computing with the integration of internet of things: architecture, applications and future directions,” in *2018 IEEE Intl Conf on Parallel & Distributed Processing with Applications, Ubiquitous Computing & Communications, Big Data & Cloud Computing, Social Computing & Networking, Sustainable Computing & Communications (ISPA/IUCC/BDCloud/SocialCom/SustainCom)*, pp. 987–994, IEEE, 2018.
- [132] A. Choudhary, M. C. Govil, G. Singh, L. K. Awasthi, and E. S. Pilli, “Task clustering-based energy-aware workflow scheduling in cloud environment,” in *2018 IEEE 20th International Conference on High Performance Computing and Communications; IEEE 16th International Conference on Smart City; IEEE 4th International Conference on Data Science and Systems (HPCC/SmartCity/DSS)*, pp. 968–973, IEEE, 2018.
- [133] M. Kaur and R. Aron, “Energy-aware load balancing in fog cloud computing,” *Materials Today: Proceedings*, 2020.
- [134] R. K. Naha, S. Garg, S. K. Battula, M. B. Amin, and D. Georgakopoulos, “Multiple linear regression-based energy-aware resource allocation in the fog computing environment,” *arXiv preprint arXiv:2103.06385*, 2021.
- [135] M. Al-khafajiy, T. Baker, M. Asim, Z. Guo, R. Ranjan, A. Longo, D. Puthal, and M. Taylor, “Comitment: A fog computing trust management approach,” *Journal of Parallel and Distributed Computing*, vol. 137, pp. 1–16, 2020.

- [136] S. Ijaz, E. U. Munir, S. G. Ahmad, M. M. Rafique, and O. F. Rana, "Energy-makespan optimization of workflow scheduling in fog-cloud computing," *Computing*, pp. 1–27, 2021.
- [137] M. H. Kashani, A. Ahmadzadeh, and E. Mahdipour, "Load balancing mechanisms in fog computing: A systematic review," *arXiv preprint arXiv:2011.14706*, 2020.
- [138] G. S. Singh and T. Vivek, "Implementation of a hybrid load balancing algorithm for cloud computing," *Int. J. Adv. Technol. Eng. Sci*, vol. 3, no. 1, pp. 73–81, 2015.
- [139] J. Oueis, E. C. Strinati, and S. Barbarossa, "The fog balancing: Load distribution for small cell cloud computing," in *2015 IEEE 81st vehicular technology conference (VTC spring)*, pp. 1–6, IEEE, 2015.
- [140] R. Beraldi, A. Mtibaa, and H. Alnuweiri, "Cooperative load balancing scheme for edge computing resources," in *2017 Second International Conference on Fog and Mobile Edge Computing (FMEC)*, pp. 94–100, IEEE, 2017.
- [141] H. Rafique, M. A. Shah, S. U. Islam, T. Maqsood, S. Khan, and C. Maple, "A novel bio-inspired hybrid algorithm (nbiha) for efficient resource management in fog computing," *IEEE Access*, 2019.
- [142] H. Arshad, "Evaluation and analysis of bio-inspired techniques for resource management and load balancing of fog computing," *Int J Adv Comput Sci Appl*, vol. 9, no. 7, 2019.
- [143] A. Fahs and G. Pierre, "Proximity-aware traffic routing in distributed fog computing platforms," 2019.
- [144] N. Javaid, A. A. Butt, K. Latif, and A. Rehman, "Cloud and fog based integrated environment for load balancing using cuckoo levy distribution and flower pollination for smart homes," in *2019 International Conference on Computer and Information Sciences (ICCIS)*, pp. 1–6, IEEE, 2019.
- [145] H. A. Khattak, H. Arshad, S. ul Islam, G. Ahmed, S. Jabbar, A. M. Sharif, and S. Khalid, "Utilization and load balancing in fog servers for health applications," *EURASIP Journal on Wireless Communications and Networking*, vol. 2019, no. 1, p. 91, 2019.

- [146] N. Bila, E. de Lara, K. Joshi, H. A. Lagar-Cavilla, M. Hiltunen, and M. Satyanarayanan, “Jettison: Efficient idle desktop consolidation with partial vm migration,” in *Proceedings of the 7th ACM european conference on Computer Systems*, pp. 211–224, 2012.
- [147] H. Bano, N. Javaid, K. Tehreem, K. Ansar, M. Zahid, and T. Nazar, “Cloud computing based resource allocation by random load balancing technique,” in *International Conference on Broadband and Wireless Computing, Communication and Applications*, pp. 28–39, Springer, 2018.
- [148] K. Saharan and A. Kumar, “Fog in comparison to cloud: A survey,” *International Journal of Computer Applications*, vol. 122, no. 3, 2015.
- [149] V. Stantchev, A. Barnawi, S. Ghulam, J. Schubert, and G. Tamm, “Smart items, fog and cloud computing as enablers of servitization in healthcare,” *Sensors & Transducers*, vol. 185, no. 2, p. 121, 2015.
- [150] C. Li, H. Zhuang, Q. Wang, and X. Zhou, “Sslb: self-similarity-based load balancing for large-scale fog computing,” *Arabian Journal for Science and Engineering*, pp. 1–12, 2018.
- [151] M. B. Kamal, N. Javaid, S. A. A. Naqvi, H. Butt, T. Saif, and M. D. Kamal, “Heuristic min-conflicts optimizing technique for load balancing on fog computing,” in *International Conference on Intelligent Networking and Collaborative Systems*, pp. 207–219, Springer, 2018.
- [152] M. Zakria, N. Javaid, M. Ismail, M. Zubair, M. A. Zaheer, and F. Saeed, “Cloud-fog based load balancing using shortest remaining time first optimization,” in *International Conference on P2P, Parallel, Grid, Cloud and Internet Computing*, pp. 199–211, Springer, 2018.
- [153] L. Liu, D. Qi, N. Zhou, and Y. Wu, “A task scheduling algorithm based on classification mining in fog computing environment,” *Wireless Communications and Mobile Computing*, vol. 2018, 2018.
- [154] B. Tang, Z. Chen, G. Hefferman, T. Wei, H. He, and Q. Yang, “A hierarchical distributed fog computing architecture for big data analysis in smart cities,” in *Proceedings of the ASE BigData & SocialInformatics 2015*, p. 28, ACM, 2015.

- [155] N. Verba, K.-M. Chao, J. Lewandowski, N. Shah, A. James, and F. Tian, “Modeling industry 4.0 based fog computing environments for application analysis and deployment,” *Future Generation Computer Systems*, vol. 91, pp. 48–60, 2019.
- [156] S. B. Nath, H. Gupta, S. Chakraborty, and S. K. Ghosh, “A survey of fog computing and communication: current researches and future directions,” *arXiv preprint arXiv:1804.04365*, 2018.
- [157] F. Jalali, K. Hinton, R. Ayre, T. Alpcan, and R. S. Tucker, “Fog computing may help to save energy in cloud computing,” *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 5, pp. 1728–1739, 2016.
- [158] C.-W. Tsai and J. J. Rodrigues, “Metaheuristic scheduling for cloud: A survey,” *IEEE Systems Journal*, vol. 8, no. 1, pp. 279–291, 2013.
- [159] A. M. Chirkin, A. S. Belloum, S. V. Kovalchuk, M. X. Makkes, M. A. Melnik, A. A. Visheratin, and D. A. Nasonov, “Execution time estimation for workflow scheduling,” *Future generation computer systems*, vol. 75, pp. 376–387, 2017.
- [160] G. Natesan and A. Chokkalingam, “Optimal task scheduling in the cloud environment using a mean grey wolf optimization algorithm,” *International Journal of Technology*, vol. 10, no. 1, pp. 126–136, 2019.
- [161] N. Siasi, A. Jaesim, and N. Ghani, “Tabu search for efficient service function chain provisioning in fog networks,” in *2019 IEEE 5th International Conference on Collaboration and Internet Computing (CIC)*, pp. 145–150, IEEE, 2019.
- [162] S. L. MIRTAHERI and H. R. SHIRZAD, “Optimized distributed resource management in fog computing by using ant-olony optimization c,” *Future Trends of HPC in a Disruptive Scenario*, vol. 34, p. 206, 2019.
- [163] D. Patel, M. K. Patra, and B. Sahoo, “Gwo based task allocation for load balancing in containerized cloud,” in *2020 International Conference on Inventive Computation Technologies (ICICT)*, pp. 655–659, IEEE, 2020.
- [164] R. Mahmud and R. Buyya, “Modelling and simulation of fog and edge computing environments using ifogsim toolkit,” *Fog and edge computing: Principles and paradigms*, pp. 1–35, 2019.

- [165] H. Gupta, A. Vahid Dastjerdi, S. K. Ghosh, and R. Buyya, “ifogsim: A toolkit for modeling and simulation of resource management techniques in the internet of things, edge and fog computing environments,” *Software: Practice and Experience*, vol. 47, no. 9, pp. 1275–1296, 2017.
- [166] J. L. de Souza Toniolli and B. Jaumard, “Resource allocation for multiple workflows in cloud-fog computing systems,” in *Proceedings of the 12th IEEE/ACM International Conference on Utility and Cloud Computing Companion*, pp. 77–84, 2019.
- [167] A. Markus and A. Kertesz, “A survey and taxonomy of simulation environments modelling fog computing,” *Simulation Modelling Practice and Theory*, vol. 101, p. 102042, 2020.
- [168] Z. Li, J. Ge, H. Yang, L. Huang, H. Hu, H. Hu, and B. Luo, “A security and cost aware scheduling algorithm for heterogeneous tasks of scientific workflow in clouds,” *Future Generation Computer Systems*, vol. 65, pp. 140–152, 2016.
- [169] L. Nieroda, L. Maas, S. Thiebes, U. Lang, A. Sunyaev, V. Achter, and M. Peifer, “irods metadata management for a cancer genome analysis workflow,” *BMC bioinformatics*, vol. 20, no. 1, pp. 1–8, 2019.
- [170] S. Kunal, A. Saha, and R. Amin, “An overview of cloud-fog computing: Architectures, applications with security challenges,” *Security and Privacy*, vol. 2, no. 4, p. e72, 2019.
- [171] G. B. Berriman, E. Deelman, J. C. Good, J. C. Jacob, D. S. Katz, C. Kesselman, A. C. Laity, T. A. Prince, G. Singh, and M.-H. Su, “Montage: a grid-enabled engine for delivering custom science-grade mosaics on demand,” in *Optimizing Scientific Return for Astronomy through Information Technologies*, vol. 5493, pp. 221–232, International Society for Optics and Photonics, 2004.
- [172] R. Graves, T. H. Jordan, S. Callaghan, E. Deelman, E. Field, G. Juve, C. Kesselman, P. Maechling, G. Mehta, K. Milner, *et al.*, “Cybershake: A physics-based seismic hazard model for southern california,” *Pure and Applied Geophysics*, vol. 168, no. 3, pp. 367–381, 2011.
- [173] X. Cai, P. Li, and X. Wu, “Artificial plant optimization algorithm with double selection strategies for dv-hop,” *Sensor Letters*, vol. 12, no. 9, pp. 1383–1387, 2014.

- [174] W. Cai, W. Yang, and X. Chen, “A global optimization algorithm based on plant growth theory: plant growth optimization,” in *2008 International conference on intelligent computation technology and automation (ICICTA)*, vol. 1, pp. 1194–1199, IEEE, 2008.
- [175] K. Guney, A. Durmus, and S. Basbug, “A plant growth simulation algorithm for pattern nulling of linear antenna arrays by amplitude control,” *Progress In Electromagnetics Research*, vol. 17, pp. 69–84, 2009.
- [176] S. Akyol and B. Alatas, “Plant intelligence based metaheuristic optimization algorithms,” *Artificial Intelligence Review*, vol. 47, no. 4, pp. 417–462, 2017.
- [177] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, “Optimization by simulated annealing,” *science*, vol. 220, no. 4598, pp. 671–680, 1983.
- [178] A. M. Fathollahi-Fard, K. Govindan, M. Hajiaghaei-Keshteli, and A. Ahmadi, “A green home health care supply chain: New modified simulated annealing algorithms,” *Journal of Cleaner Production*, vol. 240, p. 118200, 2019.
- [179] S. Bahadori-Chinibelagh, A. M. Fathollahi-Fard, and M. Hajiaghaei-Keshteli, “Two constructive algorithms to address a multi-depot home healthcare routing problem,” *IETE Journal of Research*, pp. 1–7, 2019.
- [180] S. M. Mousavi and R. Tavakkoli-Moghaddam, “A hybrid simulated annealing algorithm for location and routing scheduling problems with cross-docking in the supply chain,” *Journal of Manufacturing Systems*, vol. 32, no. 2, pp. 335–347, 2013.
- [181] H. Eskandar, A. Sadollah, A. Bahreininejad, and M. Hamdi, “Water cycle algorithm—a novel metaheuristic optimization method for solving constrained engineering optimization problems,” *Computers & Structures*, vol. 110, pp. 151–166, 2012.
- [182] A. Sadollah, H. Eskandar, A. Bahreininejad, and J. H. Kim, “Water cycle algorithm with evaporation rate for solving constrained and unconstrained optimization problems,” *Applied Soft Computing*, vol. 30, pp. 58–71, 2015.
- [183] Y. Luo, Y. Chen, and J. Wu, “Energy efficient fog computing with architecture of smart traffic lights system,” in *2021 3rd East Indonesia Conference on Computer and Information Technology (EIConCIT)*, pp. 248–254, IEEE, 2021.

- [184] Z. Ning, J. Huang, and X. Wang, “Vehicular fog computing: Enabling real-time traffic management for smart cities,” *IEEE Wireless Communications*, vol. 26, no. 1, pp. 87–93, 2019.

LIST OF PUBLICATIONS

– **International Journals(SCI/SCOPUS)**

- Kaur, M., & Aron, R. (2021). A systematic study of load balancing approaches in the fog computing environment. *The Journal of Supercomputing*, 1-46. Impact factor: 2.6 [SCI]
- Kaur, M., Aron, R. FOCALB: Fog Computing Architecture of Load Balancing for Scientific Workflow Applications. *J Grid Computing* 19, 40 (2021). <https://doi.org/10.1007/s10723-021-09584-w> [SCI]
- Kaur, M., & Aron, R. (2020). Energy-aware load balancing in fog cloud computing. *Materials Today: Proceedings*. Elsevier. [SCOPUS]
- Resource utilization-based load balancing framework for fog computing based smart application, *International Journal of Grid and High performance computing(IJGHPC)*, IGI Global [ESCI/SCOPUS] [Accepted]

– **International Conferences(SCI/SCOPUS)**

- Saroa, M. K., & Aron, R. (2018, December). Fog computing and its role in development of smart applications. In *2018 IEEE Intl Conf on Parallel Distributed Processing with Applications, Ubiquitous Computing Communications, Big Data Cloud Computing, Social Computing Networking, Sustainable Computing Communications (ISPA/IUCC/BDCLOUD/SocialCom/SustainCom)* (pp. 1120-1127). IEEE.
- Kaur, M., & Aron, R. (2020). Equal Distribution Based Load Balancing Technique for Fog-Based Cloud Computing. In *International Conference on Artificial Intelligence: Advances and Applications 2019* (pp. 189-198). Springer, Singapore.
- An energy-efficient load balancing approach for fog environment using scientific workflow applications. *International Conference on Distributed Computing and Optimization Techniques (ICDCOT – 2021)*, Springer. (Accepted and Presented).

- Fog clustering-based architecture for load balancing in scientific workflows. 4th International Conference on Computational Intelligence Data Engineering(ICCIDE-2021), Springer. (Accepted and Presented)
- **Communicated papers (SCI)**
- An energy-efficient load balancing approach for scientific workflows in fog computing, Wireless Personal Computing, Springer [Communicated]