National College of Ireland

Pareto Optimization Framework in Fog and Edge Computing

MSc Research Project
Cloud Computing

Shubham Gaur
Student ID: 22168044

School of Computing
National College of Ireland

Supervisor: Shreyas Setlur Arun

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Shubham Gaur
Student ID:	22168044
Programme:	Cloud Computing
Year:	2023
Module:	MSc Research Project
Supervisor:	Shreyas Setlur Arun
Submission Due Date:	15/12/2013
Project Title:	Pareto Optimization Framework in Fog and Edge Computing
Word Count:	9707
Page Count:	30

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	
Date:	30th January 2024

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	☐
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	☐
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	☐

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Pareto Optimization Framework in Fog and Edge Computing

Shubham Gaur
22168044

Abstract

Effective data processing at the source is necessary for the Internet of Things (IoT), frequently necessitating high speeds without depending on high bandwidth. Cloud computing and fog computing (FC) are used in tandem to achieve this. Fog computing is especially useful for real-time applications that require quick internet access. Meeting dynamic real-time needs is one of the main issues in FC, nevertheless, because of the restricted resources of fog nodes. As such, a primary challenge in the fog context is how best to distribute work across fog nodes. It is imperative to use scheduling algorithms that considers the diversity of the fog nodes and their job completion in order to minimize many aspects, such as cost and energy usage.

This work highlights important obstacles in the body of literature and provides a thorough taxonomy to improve our comprehension of the research questions pertaining to task scheduling in the cloud-fog environment. As a result, it offers a thorough analysis of work scheduling strategies in this particular setting, highlighting both their benefits and drawbacks. Multiple publications are reviewed in each of the four main kinds of approaches that are examined: deterministic mechanisms, heuristic-based, machine learning-based, and metaheuristic-based. A number of other factors include time taken for execution, utilization of resource, processing latency, network bandwidth, power consumed, execution deadlines, turn around time, unpredictability, and complexity are also compared across different strategies in the study.

The results show that 23% of the scheduling algorithms use machine learning algorithms, 38% use metaheuristic-based techniques, 30% use heuristic-based approaches, and 9% use deterministic techniques. Remarkably, energy usage stands up as the most important characteristic discussed in most of the articles, receiving 19% of the attention.

1 Introduction

As digital technologies develop further, a substantial amount of data is generated from various origins. Although cloud-computing systems can process and store this data, they are not able to fulfil the Internet of Things' (IoT) security and mobility requirements. To address these concerns, Cisco unveiled "Fog Computing (FC)" in 2012. By doing operations locally and using resources close to IoT edge devices, fog processing brings the services offered by cloud near to the network's edge and limits the volume and duration

of data transport. By using local resources, network traffic load is reduced, costs are minimized, latency is lowered, secrecy and security are improved, and so on.

When fog computing lacks effective resources, cloud resources are used, but are higher cost. The concept of fog computing and edge computing, which both are based on the concept for spreading processing resources much closer to end devices, are sometimes used interchangeably in the literature. These names will be used interchangeably to refer to this notion in this work.

When it comes to cloud-fog designs, task management is crucial. Optimal utilization of cloud and fog resource is imperative in order to enhance a number of qualitative factors, including job completion time, operating expenses, and energy usage. In a fog environment, work scheduling is crucial to cutting expenses and processing/communication delays. However, choosing an effective work scheduling technique is a problem that researchers face frequently. In order to achieve this objective, a comprehensive analysis and assessment of work scheduling strategies is necessary in the fog environment.

Many academics have studied the features of fog environments various work scheduling strategies. Within the framework of fog computing, a variety of job scheduling algorithms are categorized and examined into 4 primary categories: static, heuristic, dynamic, and hybrid. Along with outlining opportunities, the assessed qualitative factors for each approach, such as reaction time, cost, and energy use are also important. In the comprehensive evaluation of job scheduling algorithms in cloud-fog paradigm, various task scheduling tools and the strengths and drawbacks of these methods are discussed. There are also unresolved problems and potential paths for further research in the field of task scheduling. Similar to this, some publications worked on fog task scheduling, emphasizing heuristic and meta-heuristic methods and weighing the benefits and drawbacks of each technique. Nevertheless, their categorization did not incorporate the most current works in the field of fog-cloud job scheduling, nor did it cover more expansive categories like deterministic techniques.

In order to provide researchers with useful information for determining future directions for improving scheduling techniques, the studies also looked at qualitative factors and tools utilized in fog task scheduling. In the end, it divided the fog computing service management challenge into five different categories, evaluating the offered algorithms based on certain metrics and assessment tools and talking about the advantages and disadvantages of each research.

The study aims to address the following research question:

How the implementation of Pareto Optimization algorithm effectively enhance the operational efficiency and significantly reduce the energy consumption in fog and edge environments, addressing challenges associated with distributed computing environments.

The structure of the paper is as follows: Section 2 provides an official definition of fog computing along with an outline of its principles and advantages. In Section 3, problem is defined and iFogSim Design is explained. Section 4 covers the methodology, while sections 5 and 6 contains the details related to design specification and implementation of the approach. The iFogSim scalability test findings are shown in Section 7, where two basic resource management strategies are contrasted based on delay, energy consumed, and network utilisation. The article is concluded in Section 8, which also suggests possible next directions.

2 Related Work

Numerous research works have examined ways to provide affordable computing and service provision in Mobile Cloud Computing (MCC) architectures. The Energy-efficient dynamic Computation Offloading and Resources Allocation Scheme (ECOS) is an innovative solution to address challenges in Vehicular Fog Computing (VFC). By leveraging nearby vehicular nodes, VFC reduces cloudlet node overload, reduces service latency, and conserves energy in battery-powered cloudlets Yadav et al. (2020).

To minimize energy consumption while meeting latency requirements, offloading task and allocating resources to them, in Mobile Edge Computing (MEC) are optimized in this paper. As a promising architecture, MEC can significantly decrease the consumed energy for mobile devices and ensure satisfactory QOS in time-sensitive applications. This problem is decomposed into 3 smaller points: offloading selection, enhancing transmission energy, and allocation of sub carriers and processing resources Zhao et al. (2021).

By leveraging the emerging Fog computing paradigm, this study discusses challenges in resource allocation and edge application management in environments having restricted processing capabilities and resources. In Multi-objective IoT Application Placement in Fog (MAPO), a novel Pareto-based strategy is introduced to optimize application placement near data sources. This study utilizes real-world use case and simulated test beds to assess the effectiveness of MAPO Mehran et al. (2019).

Fog computing (FC) and the Internet of Everything (IoE), traditionally treated as separate paradigms, are explored in this paper Baccarelli et al. (2017). The authors describe the basic blocks and services of the technological platform and the used protocols. The paper concludes by situating the FoE model with broader landscape of recent work done, offering insight into its relevance and contribution to the field.

Rani and Garg (2021) address the challenges of energy consumption by cloud centers and propose a user friendly solution based on pareto based multi-objective discrete ant lion optimization algorithm(PBMO-DALO). The proposed algorithm uses pareto dominance parameter with spatial crowding distance to obtain optimal results. The algorithm's superiority over competing approaches is demonstrated showing uniform diversity and improved convergence.

Zhang et al. (2010) explores cloud computing as a recent paradigm for delivering services over the Internet. Despite its enormous potential, the paper emphasizes that cloud computing technology is still in its early stages with numerous issues to address despite its on-demand resource provisioning and scalability appeal to business owners.

Towards real-time, latency-sensitive applications, Mahmud et al. (2018) examines the escalating proliferation of IoT devices and sensors. To meet this demand, Fog computing is introduced, positioned as an middle layer in between the edge devices and the cloud centers. IoT fog computing extends computing, storage, and networking capabilities beyond traditional cloud computing. Based on this taxonomy, current research gaps in fog computing are identified. Moreover, the chapter provides valuable insights for advancing the field based on the identified challenges and gaps.

Data centers with significant energy consumption and operational costs are being established to meet the escalating demand for computing power Beloglazov and Buyya (2010). To maintain high Quality of Service (QoS) for customers in modern Cloud computing environments, a power-performance trade-off must be addressed. A virtualized cloud data center resource management policy is introduced in the proposed solution. VMs (Virtual Machines) are continually consolidated through live migration and idle

nodes are powered off with a focus on optimizing power consumption. The study's evaluation results indicate substantial energy savings from dynamic VM reallocation, so the policy should be developed and implemented further.

Dastjerdi and Buyya (2016) highlights how conventional cloud computing or edge computing systems cannot cope with huge amounts of data generated by the smart mobile Devices. In order to address these drawbacks, fog computing is the introduced paradigm to handle these unique challenges posed by IoT in managing and processing massive data volumes.

As the IoT devices and robust cloud services become more widespread, edge computing is emerging as a novel paradigm. Edge computing focuses on processing data at the network's edge, Shi et al. (2016) addressing challenges such as ensuring data privacy and security, conserving battery life, reducing bandwidth costs, and meeting response time requirements. This paper presents case studies ranging from cloud offloading to applications in smart cities and smart homes.

The notion of mobile edge computing (MEC) is explored in Mao et al. (2017) in vision of the IoT and 5G-communications, which are driving the paradigm shift in mobile computing. As a result of MEC, mobile computation, network management, and data management are decentralized to the network edges, such as base stations and access points, making it possible to run intensive and real-time computing applications on mobile devices with limited resource. In order to realize the 5G vision, MEC can significantly reduce latency and mobile energy consumption. An overview of state-of-the-art MEC research is presented, focusing on the management of joint radio- and computational resources.

The shift in trend from cloud computing to Edge Computing model is discussed in Mao et al. (2017). It discussed the MEC models and green MEC with heterogeneous servers. The study also delves into challenges in server selection, cooperation and computation migration including two time based management of resources, task partitioning online and large scale optimizing.

Yi et al. (2015) discusses the drawbacks and limitations of cloud computing and propose a comprehensive definition of fog computing as a solution for latency and mobility. The three layered architecture of fog, end user and cloud is discussed and an experimental fog computing platform is implemented using OpenStack for lower latency and higher bandwidth than cloud.

Ubiquitous demand for high quality mobile service and explosive growth of mobile traffic is discussed in Luan et al. (2015). The paper highlights the limitation of cloud computing lacks of location awareness and motivations behind fog computing by comparing both computing paradigm. The article explored communication challenges and integration with emerging technologies like 5G, SDN and NFV.

5G heterogeneous networks for Mobile Edge Computing demands for resource intensive mobile applications, but with the limited capabilities of smart mobiles devices energy efficient computation offloading(EECO) schemes is required. A solution classifying mobile devices, allocating radio resources and determining priorities is proposed by Zhang et al. (2016) to minimise energy consumption while meeting latency constraints. The demonstration reduces 18% energy consumption when compared with local implementation highlighting the effectiveness of EECO scheme.

Bellavista et al. (n.d.) explored the challenges while integrating fog computing with Internet of Things(IoT). The limitations of two layered architecture are highlighted and a unified architecture model is presented with fog computing closer to IoT devices. The

research emphasizes the need for different IoT applications solutions that can adapt with the guidelines for developing fog computing based applications.

2.1 Pareto Optimization Algorithm

Pareto optimization, often-referred to as multi-objective optimization, is an effective method for resolving conflicts between several competing objectives in optimization situations. Energy efficiency and latency reduction are the main goals of fog and edge job offloading.

Iteratively producing a collection of non-dominated solutions—also referred to as Pareto optimum solutions—is how the Pareto optimization method operates. Since no solution can enhance one goal without making another worse, these solutions offer the optimum trade-offs between the objectives Ngatchou et al. (2005).

By framing the challenge as a multi-objective problem, fog task offloading may be solved using the Pareto optimization technique. The jobs to offload and the fog or edge nodes to transfer them to would be the decision factors in this case. Reducing latency and energy usage would be the goal functions.

For fog and edge job offloading, a number of other techniques have been developed, including:

Round Robin (RR) Pradhan et al. (2016) is a straightforward method that distributes work across fog or edge nodes in a round-robin manner. Although RR is simple to use, latency or energy efficiency are considered in this method.

The First Come, First Served (FCFS) algorithm distributes work to edge or fog nodes according to the order in which it is received. FCFS is likewise simple to use, however it ignores latency and energy economy.

Natural selection serves as the inspiration for the Genetic Algorithm (GA), a metaheuristic algorithm. Although GA has shown to be successful in offloading fog and edge tasks, it can be computationally costly. Another metaheuristic algorithm that draws inspiration from ant behavior is Ant Colony Optimization (ACO). It was demonstrated to be successful in fog and edge job offloading, ACO can be parameter sensitive.

2.2 Task scheduling in Fog Computing

Several approaches have been developed in the field of cloud-fog integration to simplify the administration of these systems Alizadeh et al. (2020). Through workload distribution, these methods help reduce server congestion on cloud infrastructure. The edge devices dispatch their requests to the cloud or fog throughout this procedure. After the requests have been assessed within the fog, they are either processed there or sent to the cloud, based on the particular characteristics of the jobs.

Task scheduling takes on considerably more significance in scenarios such as the Internet of cars (IoC), where a variety of linked cars create huge amounts of data. Effective scheduling techniques are essential for reducing latency and guaranteeing timely task completion, which improves traffic control and vehicle safety.

Fog computing presents a complex work scheduling difficulty. The dynamic character of the fog paradigm, marked by fluctuating workloads, and mobility, is the cause of this complexity. As such, there is a need for efficient methods of managing resources and scheduling tasks in the fog-cloud paradigm. To tackle the issue, some approaches and

strategies include objective based optimizing, ML based techniques, or heuristic based approach.

Approaches based on heuristic techniques use pre-established rules or heuristics to distribute jobs across fog nodes while taking workload, availability, and proximity into account. These techniques are simple and effective, but they might not always produce the best results. Consequently, in order to improve scheduling performance, several researchers have improved heuristic algorithms, especially when edge nodes are clustered together on edge network to execute jobs decentralised and in parallel. Based on kind of operating system, execution time, or arrival time, these techniques groups tasks together.

However, optimization-based approaches formulate the scheduling of tasks challenge as an optimization issue and use techniques such as linear or integer programming to discover the optimal solution. These techniques can produce ideal results, but they can be computationally taxing.

2.3 Approaches for the Selection of Research

A thorough analysis of earlier research is a step in the Systematic Literature Review (SLR) process. In this part, we introduce an SLR-based method for examining fog computing job scheduling strategies. To find pertinent papers, we used six well-known databases: ACM, IEEE, ScienceDirect, Springer, Wiley, and MDPI. In order to streamline the selection process, we created a search string by combining a set of search terms including logical operators AND and OR. The search query looks like this:

(Scheduling OR Task scheduling) AND (Fog OR Fog Computing)

First, we searched the designated databases using keywords. We sifted the research and took into consideration those that were published after the beginning of 2018 due to the large volume of search results. Then, we narrowed down our choices by assessing the research according to their names and keywords. In order to further reduce the selection, we carefully read the study abstracts in the third phase. After carefully going over all of the material, we ultimately kept just the papers that were completely relevant to our study question for additional analysis.

2.4 System Architecture and Problem Synopsis

The cloud and edge combined offloading architecture is designed because cloud servers provide large computational power and storage resources along with a wide range of applications, on the other hand, edge servers is better in inexpensive communication, fast turn around times, and enhanced adaptability. The benefits of edge and cloud servers are seamlessly combined in this design, enabling the full utilisation of a variety of computing resources. Based on the unique needs of various applications, tasks are automatically offloaded to the best location, taking into account variables like processing capability, energy consumption, and latency. Three basic layers make up the architecture, as shown in Figure 1: the cloud layer, the edge layer, and the end-device layer.

The end-device layer includes a diverse range of sensor. Such devices commonly have compact sizes, battery life is constraint, and are capable of low computational and data management. As a result, to process data even more, these devices use wireless access points (APs) to have successful communication with cloud layer.

Majority of the edge layers is made up by the light weighted edge servers which offers computing service with ultra low latency. To avoid any overloading or delay in the

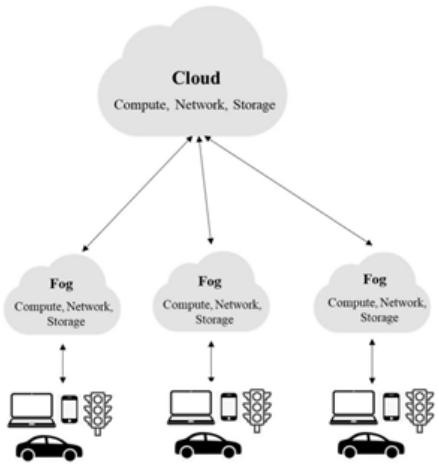

Figure 1: The architecture of the three-layer fog computing

communication, these edge servers can transmit workloads to cloud servers for processing using wired connections.

Multiple cloud-based servers with abundant processing and storage capacity are housed in the cloud layer, serving users concurrently from different geographic locations. On the other hand, higher communication costs are a result of more users using the system at the same time and longer data transmission lengths to different levels.

2.5 Resource Distribution

Innovative methods to resource allocation difficulties in Mobile Edge Computing (MEC) have been observed in recent study. The resource allocation problem in a multi-user MEC system was solved by applying an optimisation framework based on reinforcement learning. The introduction of a method for user pairing that prioritises maximum classification differences and sub-maximum viewpoints was based on optimising non-orthogonal multiple access (NOMA) system capacity improvements. Performance-criticizing learning algorithms were used in conjunction with asynchronous advantage participation to solve complex dynamic resource allocation challenges. Through deep reinforcement learning, the study investigated decentralised strategies for inter vehicle communication. The study investigated the best resource allocation techniques for ultra-dense networks (UDNs) by utilising an algorithm that relies on the alternating direction multiplier approach. Research also explored methods for allocating resources to reduce average service response times.

Additionally, studies looked at workload distribution in MEC, employing different strategies to reduce latency. These techniques included data insertion strategies inside MEC processes, genetic algorithm-based computing resource allocation, and game theory and Particle Swarm Optimisation (PSO) Shu and Li (2023) based MEC job allocation strategies. These strategies aimed to efficiently lower data placement expenses.

The main goals of task offloading were to break down tasks into smaller components, choose between local execution and offloading, figure out the number of offloading tasks, then create similar edge server situations. Nevertheless, problem of assigning the jobs produced by various smart devices to heterogeneous servers on edge is brought about by the everyday use IoT technologies. The complex link between dumping costs and time

delays has not been well addressed in the existing literature, nor has it been modelled successfully for this multi-objective situation.

3 Problem Defined

Among the many smart devices that industrial facilities utilise in the context of the Industrial Internet of Things in the 5G environment are smartphones, smart cameras, and augmented reality (AR) devices—all of which are referred to as "multi-user." These gadgets are essential to intelligent production lines in manufacturing environments. The tasks created by these smart devices typically demand the engagement of numerous Mobile Edge Computing (MEC) servers, due to the significant volume of data that has to be processed; this is known as the "multi-MEC" feature. The purpose of this work is to verify the effectiveness of using the Particle Swarm Optimisation offloading technique for detecting suitable offloading resource and, as a result, to produce task offloading results that minimise the total task processing latency.

In this study, we assume the existence of M smart devices and N MEC servers, operating in an Industrial Internet of Things context. Every smart device creates tasks that are assigned to be computed on a particular MEC server. Note that we only handle non-divisible tasks, and every smart device sends in a single task to be processed. It is possible to carry out each job locally or by offloading it to MEC servers for distant execution. Each job may therefore be handled at one of N+1 possible places, which includes the possibilities of local execution or offloading to N MEC servers. Three core components of the study are examined: the computational model, the energy consumption model, and the time delay model. Table 2 provides clarification on the meanings of the symbols used in the system modelling.

3.1 Model of time delay

The overall delay for completing job i on server j includes transmission delay with server delay. This may be computed as:

$$T_i^j = T_{tran,i}^j + T_{mec,i}^j$$

We create the notation where Di is the amount of data needed to execute a job, and C_i is the quantity of CPU cycles required to process each bit of data. Consequently, D_i multiply by C_i yields the volume computation. Moreover, the job volume divided by the MEC server's CPU frequency yields the MEC calculation latency, as shown by the following equation:

$$T_{mec,i}^j = \frac{D_i * C_i}{C_{s,j}}$$

The channel transmission rate may be obtained using Shannon's formula in order to factor in the MEC's transmission delay in our calculations. In this instance, we have the subsequent:

$$r_{i,j} = W * (1 + \frac{s_i * A_{i,j}}{W * N_0})$$

where W denotes transmission bandwidth, N_0 denotes noise power spectral density, S_i shows the power transmitted by individual device, and $A_{i,j}$ is the gain from device i to j

Figure 2: Symbol Table for MEC Task Allocation

Symbol	Meaning
D_i	The amount of data for the i-th task
C_i	CPU cycles needed to process each bit of data
$C_{u,i}$	CPU frequency of the i-th device
$C_{s,j}$	CPU frequency of the j-th MEC server
W	Transmission bandwidth
S_i	Transmission power of the i-th device
$A_{i,j}$	Channel gain
N_0	Noise power spectral density
$r_{i,j}$	Transmission rate from local device i to the MEC server j
E_{maxj}	Maximum energy consumption constraint of the j-th MEC server
g	Penalty factor
T_{ij}	The total delay from local device i to the MEC server j
E_{ij}	The energy consumption from the i-th device to the j-th MEC server
$E_{calc,ij}$	Calculation energy consumption from the i-th device to the j-th MEC server
$E_{tran,ij}$	Transmission energy consumption from the i-th device to the j-th MEC server
$T_{mec,ij}$	Calculation delay from the i-th device to the j-th MEC server
$T_{tran,ij}$	Transmission delay from the i-th device to the j-th MEC server
M	The number of tasks
N	The number of MEC servers

server. Consequently, it is possible to turn the transmission delay into:

$$T^j_{tran,i} = \frac{D_i * C_i}{r_{i,j}}$$

Next, for j MEC server, the overall latency in finishing the i job is determined by:

$$T^j_i = \frac{D_i * C_i}{C_{s,j}} + \frac{D_i * C_i}{W * (1 + \frac{S_i * A_{i,j}}{W * N_0})}$$

3.2 Model of energy consumption

The i job on the j computing server has 2 components to its power usage: the energy consumed in the processing and computation and the energy used in the transmission.

$$E^j_i = E^j_{calc,i} + E^j_{tran,i}$$

where U denotes voltage and R_i is dependent on the effective switching capacitance.

Transmission energy use is incurred by the tasks sent to the server. Given that the delay model has computed the delay in transmission, the energy consumed in transmission is

$$E^j_{tran,i} = S_i * T^j_{tran,i}$$

where Si stands for each local device's transmit power. Next, the whole amount of energy used is:

$$E^j_i = R_i * U^2 * C_{i,j} * D_i * C_i + \frac{D_i * C_i}{r_{i,j}} + S_i$$

3.3 Calculation model:

Reducing the delay issue is the main goal in the current situation. On the other hand, if we ignore the queuing delay, there's a chance that a particular MEC server with more computational power will find itself in a scenario where it's overloaded. The majority of jobs are delegated to this MEC server as queuing time is not a factor, which results in high energy consumption and maybe exceeds its maximum energy consumption limit. In these situations, job offloading to a different MEC server can improve performance while reducing strain on the severely loaded server. In order to achieve a more balanced approach, we thus suggest to introduce a penalty function and divide the load equally among all computing servers.

$$Func(X) = \sum_{j=1}^{N} \sum_{i=1}^{M} T_i^j + P(X)$$

$$P(X) = (\sum_{j=1}^{N} \sum_{i=1}^{M} (E_i^j - E_{max}^j)) * g$$

The power usage limit for the j computing server is denoted by E_{max}^j. The offloading vector X = x1, x2,..., xM is utilized, with total job size to be executed is represented by X. For each element xi, there are two definitions: either xi = 0 (local execution) or $xi \in [1, N]$ (identification of the MEC server used to handle the particular job). Weights for the penalty and delay functions are included to improve the optimization of the method.

The above introduced set of equations and computational model specifically centred around the task allocation in fog and edge computing and play important role in iFogSim simulator. Although the manual calculations doesn't include the use of these equations, they were essential for iFogSim decision making process. The outlined models influenced how tasks are allocated to devices in fog and edge environment withing the simulation, replicating the complex dynamics of fog and edge computing. The simulator allowed us to observe critical metrics by executing task allocation based on these models. It is crucial to recognize the importance of these models and equations for understanding the basics of energy consumption in fog and edge computing. Their integration in iFogSim simulator enhance the understanding of simulator's functionality and its alignment with real world scenarios.

3.4 Mobile Edge Computing

ETSA in 2014, introduced Mobile Edge Computing (MEC) to offer mobile devices with computing powers of cloud servers. The host level and the system level are the two separate levels that make up the MEC reference architecture, as described by ETSA in the literature. A MEC platform manager, infrastructure manager, and a host are included at the MEC host level. MEC orchestrators, management of life-cycle of user application, and operator support systems are all included at the system level in MEC. Beyond the MEC system, there is also a network layer that includes local networks, external networks, cellular networks, and relevant external elements that represents MEC system entry scenario. Ben Sada et al. (2023) discussed a multi layered task offloading framework including the cloud, device and cloudlet layers. When this framework is compared to more sophisticated compute offloading technologies, it performs better.

MEC gives the traditional Radio Access Network (RAN) low latency, high bandwidth transmission capabilities, and the ability to conduct localised business operations. It also makes it easier to roll out services across small distances. This reduces the amount of bandwidth and latency that mobile networks must handle. During business downturns, localised deployment plays a crucial role in lowering network load and bandwidth demands, which results in lower network operating costs and better network resource utilisation. Figure 3 shows the computation offloading flow diagram.

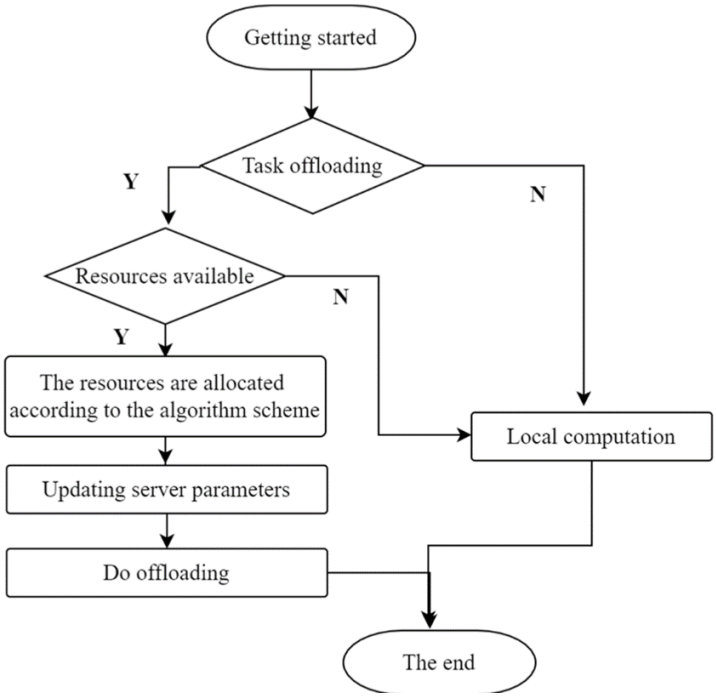

Figure 3: Flow diagram for computational offloading

3.5 Application Situations for Edge Computing in Mobile

One important component of Mobile Edge Computing (MEC) is computational offloading methods, which have found a wide range of applications in different circumstances. MEC has been deployed in important domains like 5G, IoT, Vehicular communication, automated vehicles, AR/VR, or other contexts in which offloading plays crucial role because of its close proximity to edge devices also they innate the computing capabilities and storage power.

3.5.1 IoT, or the Internet of Things

Before these data packets reach the main network, MEC is useful in processing and aggregating the tiny packets created by IoT services. This method improves the connection and adaptability of IoT apps, which is especially helpful for battery-operated IoT devices. By reducing the time consumed in transmission between devices and servers, MEC adoption improves the quality of services and devices while also using less battery life Mehrabi et al. (2019). Supporting long-term corporate goals requires this.

The active IoT devices globally is predicted to exceed 10B by year 2020 and rise to 22 billion by 2025, according to a December 2019 IoT platform study issued by IoT Analytics, a research organisation that specialises in Internet of Things. These gadgets gather a tremendous quantity of data, thus sending it all to the cloud service centre for processing would put a heavy demand on the distant cloud.

Unfortunately, a lot of IoT devices have low processing power or are incapable of processing data effectively. By minimising energy usage and reducing transmission time between servers and devices, MEC deployment improves the continuity for devices to meet extended business objectives. For instance, the compute offloading method of User Equipments (UEs) with an emphasis on Mobile Edge Computing (MEC) in the context of the Internet of Things.

3.5.2 Services for Smart Buildings

These days, a lot of smart buildings are developing integrated technologies, and the majority of these systems need or produce local data. Thus, a local IoT gateway is employed to enable data connection across processing equipment. Processing and data control, such as entry control, temperature management, intelligent signalling, object monitoring, and surveillance, are handled by MEC ETSI-Industry-Specification-Group (2020).

The data is moved closer to the device enabling quick processing and storage than central servers. This suggests that rather than constantly depending on a central computing environment, MEC have capacity to function locally. The general effective objective of smart buildings is aligned with the design of edge computing. In addition to lowering latency, it also results in considerable cost savings since data is processed more quickly, allowing for quicker reaction times and real-time decision-making. The edge system has helped in more efficient understanding of these variations, analysing how they affect the underlying framework.

3.5.3 Task Offloading for MEC

The act of moving computational work from Mobile Devices (MDs) to expanded cloud or grid platforms is known as computing offloading Feng et al. (2022). This allows end applications to respond to user requests to return calculated output. With the Mobile Edge computational (MEC) technology, users can summon computational capacity near the region for data computing by using MEC server as an intermediary. This strategy deviates from the standard practice of sending data to a central cloud located at a distance for processing. Figure 4 shows the flowchart that illustrates the MEC system's compute offload procedure.

A computation task can be run at the edge server or locally. Local execution entails the computing task and output result being completed by the mobile device itself. An offloading request must be made prior to remote execution. The MEC server assesses if the user's service request is legitimate: The MEC server updates the status of system energy consumption and computational resources based on resource utilisation and user task data volume, if the request is accepted. The service cannot be rendered to an unauthorised user. The server computes the results and sends them back to the mobile device after processing the tasks supplied by the device in accordance with the computing resources that have been allotted.

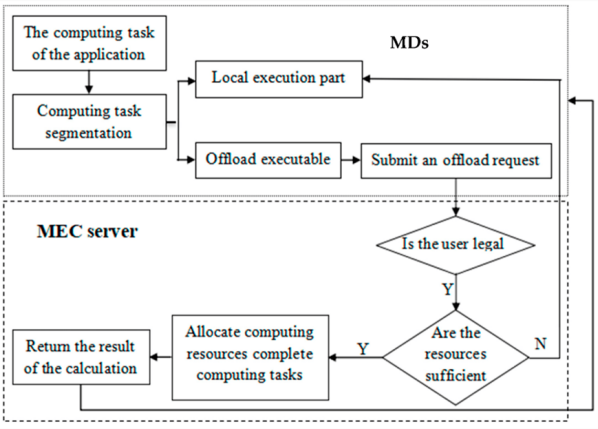

Figure 4: Process of unloading tasks

In edge settings, heuristic algorithms are crucial in addressing the problem of task scheduling by guaranteeing the best possible resource allocation for activities that users request. Rather of using exhaustive search techniques, these algorithms provide useful and efficient answers by depending on approximations, recommendations, or rules of thumb. The Grey Wolf optimization, Particle Swarm Optimisation and other key prototypes are some examples in this field.

Heuristic algorithms are excellent at discovering workable solutions when faced with limitations, but they have two drawbacks: they tend to converge slowly and become trapped in locally optimum solutions. The unanticipated difference between viable and ideal solutions might make it difficult to satisfy the criteria of low-delay activities. The reinforcement learning technique Wang et al. (2021) is used to solve scheduling issues in intricate edge settings as a reaction to these difficulties.

By continually correcting disparities between feasible and optimum solutions, reinforcement learning speeds up convergence and raises the general quality of solutions. It is limited, nonetheless, when it comes to solving continuity and high dimension difficulties. Deep learning techniques, on the other hand, are particularly good at identifying features in data since they emphasize conveying perception and input. Deep Reinforcement learning make use of deep neural network to detect environmental features combining with federated learning for resolving challenging system issues. With this method, edge nodes act as intelligent agents that pick up scheduling rules without needing to know anything about their surroundings at all.

3.5.4 iFogSim

The case studies that illustrates the process of modelling an IoT ecosystem and contrasts the approaches to resource management strategies are included in the article. For evaluating the toolkit scalability, time taken and RAM used in the execution are analysed. The Internet of Things (IoT) paradigm presents an opportunity to include, into the Internet environment, a variety of "things," such as consumer electronics and household appliances, such as refrigerators, cameras, sensors, and medical equipment. This idea opens up new avenues for creative interactions between people and objects, enabling the development of urban areas, cutting-edge infrastructure, and maximising resource use. Integration, transmission, and analysis of data produced by smart devices—like

sensors—are made easier by the Internet of Things. Real-time analytics and decision-making have not received enough attention as a primary goal of IoT, despite the quick development of technology facilitating connectivity and data transfer[5]. Most IoT data processing options available today processes data by shifting it to the cloud.

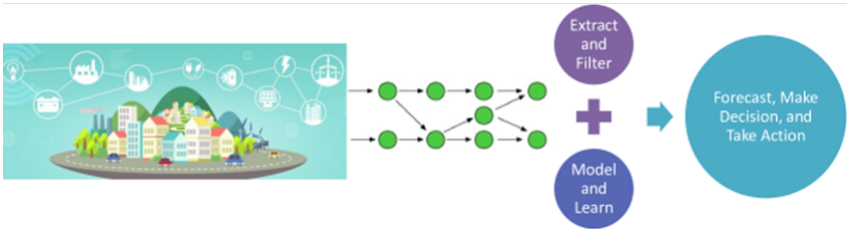

Figure 5: Common Objectives for IoT Environments

Many Internet of Things applications, like stream processing, are dispersed by design and are frequently integrated into environments with a large number of networked computer devices with a variety of functionalities. An assessment environment is required to investigate different resource management and scheduling strategies in order to promote the growth of real-time analytics in fog computing through innovation and progress. This includes factors like task allocation, migration, and consolidation as well as operator—which is often used interchangeably with application module—and operator. Real-world IoT environments are excellent testbeds, but they are sometimes too costly and lack the repeatability and controllability needed for methodical testing.

In order to get around this restriction, we suggest using a simulator called iFogSim. This simulator offers a regulated and affordable substitute for real-world IoT testing settings by simulating resource allocation with application scheduling strategies over the edge and cloud resources under different situations and conditions. In order to evaluate resource management strategies in Fog settings, this article explores the architecture, design, and implementation of iFogSim for analysing latency effects, energy consumed in execution, cost of runing the operations Mahmud et al. (2022). To measure performance measures, the framework simulates network connections, cloud processing, and edge devices. The detect-analyse-execute application paradigm is the main one being considered. In this approach, sensors broadcast data to Internet of Things networks, Fog devices subscribe to and analyse sensor data, and actions are taken in response to the insights obtained, which are then directed towards actuators.

4 Methodology

In this segment, fog computing is described formally as a decentralized computing and processing model that expands and extends the services of cloud to the edge of the network. As seen in Figure 6, this paradigm incorporates its own infrastructure and seamlessly combines cloud and edge resources. The core of fog computing constitutes integration of code development, connectivity, and data repository. The deployment of application components in the cloud and on devices like smart gateways and routers that are positioned in between endpoints and the cloud is essential to its functioning. Fog computing is made to work with heterogeneous resources and interfaces, facilitate mobility, communicate with the cloud, and perform dispersed data analytics. This addresses

the application requirements with minimal delay with extensive widespread coverage geographically.

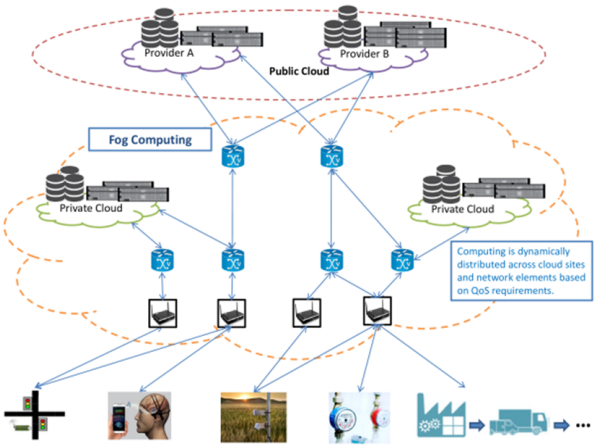

Figure 6: Distributed Data Processing in Fog Environment

The Fog Computing integrates the advantages of both paradigm cloud and edge computing. It uses on-demand scalability that cloud resources provide in addition to the processing in close proximity of IoT devices, which optimises their response times. To sum up, fog computing is a comprehensive strategy that combines the advantages of cloud and edge computing to meet dispersed processing, real time response, and a broad geographic footprint.

Fog computing is popular because it provides a number of benefits. Reducing network traffic is one of the main advantages as it lowers the possibility of congestion and high delay brought on by unchecked growth in data transfer. By using edge device resources, fog computing creates a platform for data generation from sensors that can be filtered and analysed. By placing filtering operators nearby to the source of data generated, can significantly decrease the amount of traffic that is routed to the cloud.

The Fog computing reduces the response delay significantly, which is especially helpful for critically complex applications that need analysis in actual time. Cloud robotics, advanced aircraft control, and automatic brakes in automobiles are a few examples of such uses. Cloud-based robotics situations, where fast data collection, control system processing, and actuator input are critical to motion control effectiveness, might result in delays or unavailability of the control system owing to communication issues. To overcome this difficulty, fog computing executes control system operations near the robots to guarantee real-time reaction times.

Finally, fog computing deals with certain bottlenecks that might occur in the paradigm while processing the original data IoT devices generate, despite its nearly limitless resources. By dispersing the data processing architecture and improving scalability, fog computing can filter and process large amounts of incoming data on edge devices. To summarise, fog computing's efficacy and suitability for a range of crucial applications stem from its capacity to decrease network traffic, minimise latency, and disperse data processing.

4.1 Architecture

As shown in Figure 7, the Fog computing environment has a hierarchical design with Fog nodes dispersed across the network to act as bridges between the central cloud infrastructure and end devices or sensors. IoT sensors are arranged in this configuration at the lowest layer, distributed over different geographic regions. These sensors monitor their surroundings and use gateways to send recorded results to higher layers for further analysis and decision making. Concurrently, IoT actuators operated in lowest tier, managing systems or mechanisms. Actuators are made to react to changes in the surroundings that are detected by sensors. Sensors transmit immutable values in sequences that make up IoT data streams.

Any network component that may host application modules is called a Fog Device in the architecture Hu et al. (2017). Gateways connects the sensor devices to the internet. These Fog devices include on-demand cloud resources that are dispersed across many data centres.

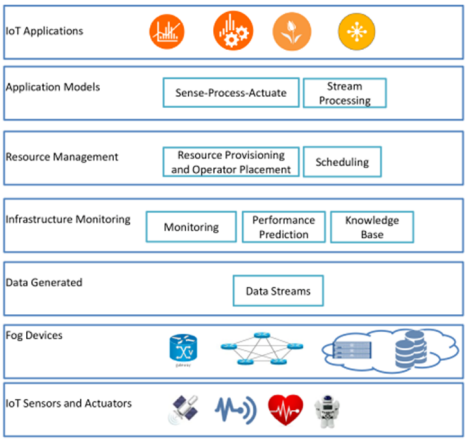

Figure 7: Computer Architecture for Fog

For fog and IoT contexts, the architecture also specifies two application models and three primary services:

Components for Monitoring: These components monitor the availability and use of fog network and resources in the system. When necessary, the information is shared to other services by keeping an eye on the functionality and state of the apps and services installed on the infrastructure. Maintaining a thorough understanding of the effectiveness and state of the system depends heavily on component monitoring.

The central part of the design is resource management, which includes components that efficiently manage resources to satisfy application-level Quality-of-Service (QoS) requirements while reducing resource waste. Placement and Scheduler components are essential to this domain as they are crucial to keeping track of the resources that are available (data supplied by the Monitoring service) and determining which application modules are the best fit for hosting.

Power monitoring, which tackles the particular problem of energy usage in Internet of Things solutions, is a crucial component of resource management. Energy management is made more difficult by fog computing, which, in contrast to cloud data centres, comprises a large number of devices with different power consumption characteristics.

Consequently, the power monitoring component, which keeps track of and reports on the energy usage of Fog devices during simulation, is an essential element to the design. Prior to implementing applications and resource management strategies in production environments, it is imperative to assess their influence on energy usage.

The application model used by the architecture to create applications meant to be deployed in the Fog is Distributed Data Flow(DDF) model. According to this approach, an application is thought of as a group of modules that stand in for different data processing components.The modules i and j are inter dependent as the data from i module acts as input for the j module. With the use of this application model, an application may be shown as a directed graph, with application modules represented as vertices and data flowing between them as directed edges. Two example applications that are modelled using the Distributed Data Flow model are provided later in the study.

4.2 jMetal Framework

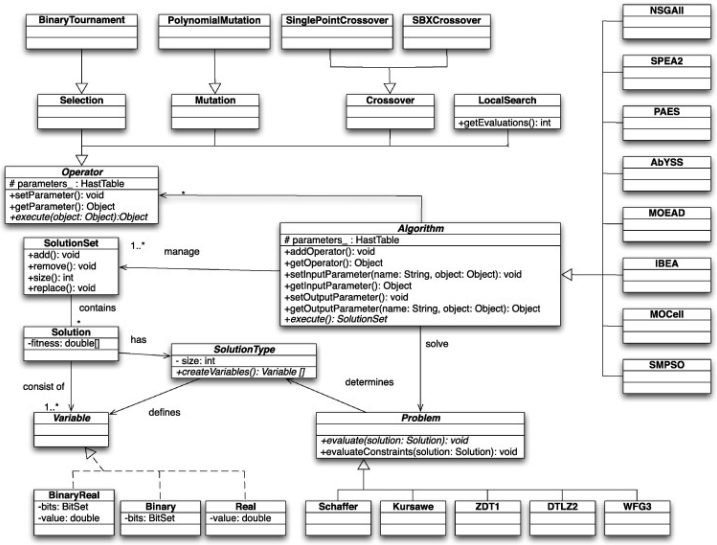

Figure 8: jMetal Base classes

Multi objective optimization algorithm are used to optimize the problems that involve two or more conflicting objectives. jMetal is a flexible and user friendly java based framework developed for multi objective optimization using metaheuristic algorithm. The core components 8 of the object oriented architecture of jMetal includes jMetal Solution that represents potential solutions, jMetal Operator that defines operators on the solutions, jMetal Problem which includes the optimization problems and jMetal Algorithm including the abstract classes for metaheuristic algorithm Durillo et al. (2010).

5 Design Specification

We made use of CloudSim's fundamental event simulation features in order to develop the iFogSim architecture's functionalities. Data centers and other CloudSim entities exchange messages using message passing procedures, more precisely sending events. Consequently, in iFogSim, the interchange of events across Fog computing components is managed by

CloudSim's core layer. The figures 9 depict the main classes of iFogSim. We get into these classes' specifics and how they interact in this section. Modelling the components of the architecture as iFogSim classes, the simulated entities and services compose the iFogSim implementation.

The primary components of iFogSim include:

Entities: Fog devices, sensors, data centres, and other parts of the Fog computing ecosystem are represented by these basic simulation units.

Events: Within the simulation, events contain actions or state changes. These events help the many components of iFogSim communicate with one another and exchange information.

Services: Fog computing component capabilities and behaviours, such resource management, monitoring, and power monitoring, are encapsulated as services in iFogSim.

Simulation Clock: An essential component that establishes the sequence in which events in the simulation are carried out is the simulation clock.

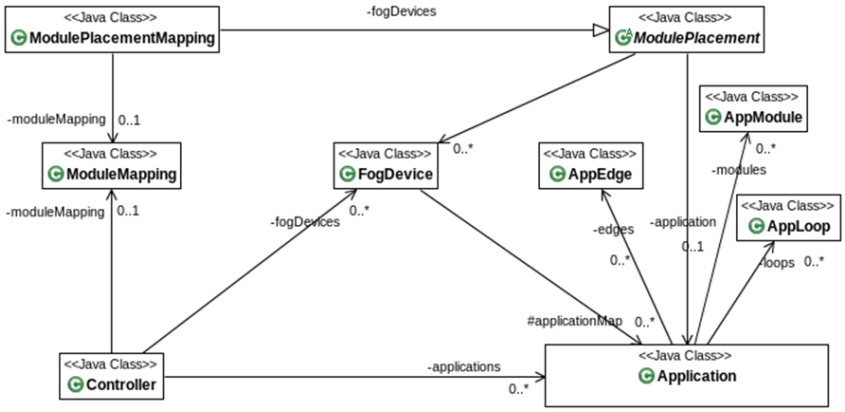

Figure 9: iFogSim main classes

These elements work together in a coordinated manner to replicate the dynamics of fog computing, which makes it possible to assess resource management strategies and application scheduling tactics. Built on top of CloudSim, the iFogSim architecture offers a framework for investigating and evaluating the functionality of Fog computing systems in a range of settings.

Several essential classes are included in the iFogSim implementation to represent various elements of the Fog computing environment:

FogDevice: This class describes the physical properties of fog machines and how they are connected to other fog machines, sensors, and actuators. It is derived from the PowerDatacenter class in CloudSim and contains characteristics that define the communication capability of Fog devices, including available memory, CPU information, storage capacity, and uplink/downlink bandwidths.

Methods: This class's methods manages the deployment of application modules, with Fog device's resources scheduling amongst them. These methods can be overridden by developers to apply unique rules.

Sensing In the architecture, instances of this class stand in for IoT sensors. The class contains attributes that specify the properties, connection, and output features of the

sensor. It also contains information on the latency of the connection between the sensor and the gateway Fog device. This class distributes the tuple inter-transmission times and determines the rate of transmission.

Tuples These are the basic building blocks of the Fog, and in iFogSim, the class Tuple instance is used to create these building blocks. Tuples, extends the CloudSim class named Cloudlet, are identified by their kind, source and destination application modules, and the amount of data they contain. Their processing needs are expressed in Millions of Instructions per second.

An actuator This class defines the attributes of a network connection by modelling an actuator. It has a property about the delay between the actuator and the gateway connection. Additionally, this class defines a function to be called when a tuple from an application module arrives.

Utilisation A directed graph employed to depict an application, where vertices are modules that process input data and edges are the data relationships between these modules. Applications may be modelled with the help of several classes and entities, which makes for an all-encompassing representation in iFogSim. A number of extra classes are included in the iFogSim architecture to represent various facets of fog applications:

Module App The App Module Class represents the processing components of fog application. CloudSim PowerVm class is extended for defining this class. Directed Acyclic Graph (DAG) represents the application using the output tuples created by AppModule instance. Selectivity model can be employed for determining the volume of output tuples based on burst or fractional selectivity.

AppEnd An AppEdge instance represents the interdependence of two application modules. The AppEdge class captures the tuple that each edge carries in addition with the length of data wrapped with each tuples and the processing requirements. Periodic and event-based application edges are supported by iFogSim. While event-based edges send tuples whenever the module receives tuple permitted by selectivity model, tuples emission is carried by that edge, periodic edges emit tuples at regular intervals.

AppLoop Control loops that the user may find interesting are specified using the AppLoop class. Control loops can be specified by developers in iFogSim in order to analyse overall latency. An AppLoop constitutes modules list from loop starting point till where it concludes. This enables programmers to specify and examine certain control loops in the Fog application.

The sequential flowchart presented in Figure 10 depicts the steps involved in tuple emission and subsequent execution in iFogSim. A sensor creates a tuple and sends it to the gateway that the sensor is attached to. On tuple arrival at the device, processTupleArrival() callback function is executed. The tuple is transmitted right away, without any additional processing, if it has to be forwarded to another Fog device. If the application is set up on the receiver end device, the tuple is executed. checkCloudletCompletion() method is executed subsequent to execution of the tuple. Apart from these fundamental tuple processing features, iFogSim integrates simulated services, such as:

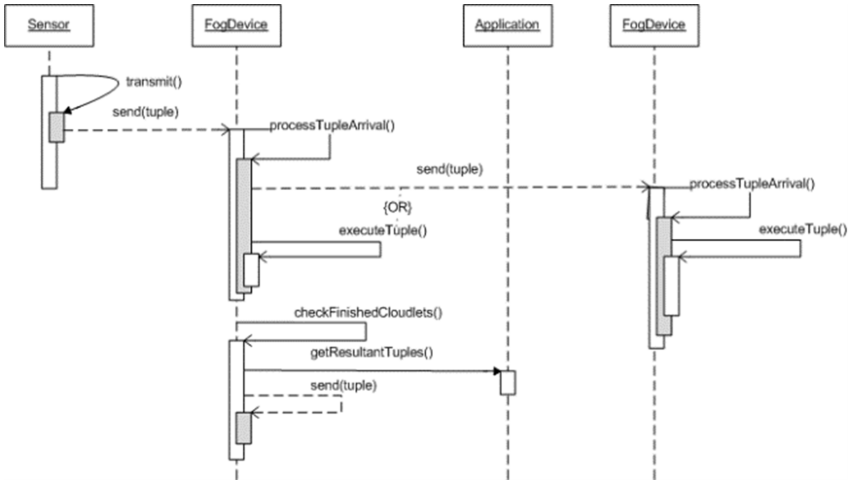

Figure 10: Sequence Diagram

Service for Monitoring Power

A power model (such as PowerModelLinear) is linked to every Fog device, which is an instance of the FogDevice class. Tuple processing logic is included in the executeTuple() function of the FogDevice class. It uses the associated power model to update the device's power usage in response to variations in resource utilisation.

Service for Resource Management

Placement and Scheduling are the two tiers of resource management that iFogSim offers for applications. These can be extended and customised in accordance with particular application needs because they are abstracted as distinct policies.

Application Placement and Application Scheduling are two essential components of resource management for Fog applications in iFogSim:

Application Scheduling: When an application is submitted, the dispersion of modules among fog devices is handled by placement policy. Certain goals, such minimising energy use, optimising operating expenses, cutting down on network utilisation, and minimising end-to-end latency, drive the placement process. Various placement policies are built upon the abstract class ModulePlacement, which must be expanded to accommodate new policies that are customised to meet particular needs.

Application Time Management Allocating the resource of fog to application module is another level in this service. All active application modules on a device share its resources evenly according to iFogSim's default resource scheduler. On the other hand, developers can override the updateAllocatedMips function in the FogDevice class to customize this scheduling scheme. Because of this customization, the application scheduling strategy may be adjusted to suit the particular requirements and features of the Fog environment that is being used.

Placement and Scheduling, the two tiers of resource management, provide developers the freedom to create and modify techniques that fit the unique objectives and limitations of their Fog applications.

5.1 A GUI for graphical interface

A graphical user interface (GUI) built into iFogSim makes it easier to describe the actual network architecture. The iFogSim application logic serves as the foundation for this GUI, which gives users the ability to graphically represent physical components including fog devices, sensors, actuators, and connecting connections. Through the GUI, users may enter the defining features of these entities. To save and load the developed topologies interface, JSON format is used. Physical topologies may be constructed programmatically with Java APIs or via the GUI.

Through GUI Users may visually create a topology by drawing and defining the components of the physical network using the graphical user interface.

Programmatically, Java APIs give developers the ability to build physical topologies, offering flexibility and automation in topology construction. A sample physical architecture developed using the GUI is seen in Figure 11, along with the connections between a sensor, gateway, cloud virtual machine, and other devices.

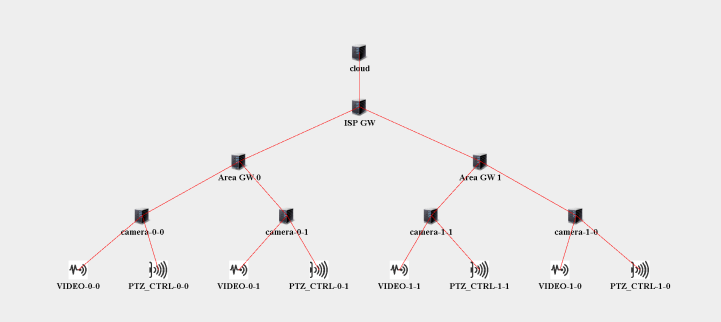

Figure 11: Network topology GUI

6 Implementation

USE CASE: INTELLIGENT VEHICLE PARKING APPLICATION

The implementation of smart vehicle parking application in fog and edge environment includes modules representing various functionalities like image capture, object detection, or alarm control. The underlying infrastructure include fog devices like smart cameras and proxy servers, each having their characteristics such as MIPS, bandwidth, or RAM. The application supports two deployment models as discussed above, the cloud based deployment involves all task to be offloaded to the cloud server, on the other hand, in edge ward deployment the task are divided between smart cameras and the proxy server, which acts a fog computing unit. For multi objective optimization, the Non-dominated Sorting Genetic Algorithm(NSGAII) is employed in the application having two objectives defined: efficiency and energy consumption. These objective are then being utilised to update the parameters of the application such as transmission time, or number of camera per area. Utilising distributed camera networks for smart car parking is the main topic of the case study. This system, which has applications in manufacturing, transportation, public safety, security, makes use of a dispersed network of cameras to monitor a space

effectively and automatically analyse data.The characteristics of the system includes the following:

Minimal-latency Interaction

Goal: By adjusting the control (PTZ) settings of many cameras sensors without any time delay depending on taken pictures, enable efficient object coverage.
Challenge: Needs a set of camera control methods and high speed transmission with minimal delay between the devices.

Managing A Lot of Data

Goal: Handle continually incoming video frames from cameras without adding to the load on the network.
Challenge: The network may become overwhelmed by the amount of data produced by video cameras, thus efficient data handling techniques are required.

Extensive, Extended Processing

Goal: Learn the best PTZ parameter calculation approach by continuously updating the camera control strategy.
Challenge: The study of the strategy for controlling actions for an extended duration requires powerful processing capabilities due to its computationally expensive nature. The intricacy of smart car parking systems and the requirement for effective data management, communication, and long-term processing capabilities are demonstrated by this case study. By simulating such a system with iFogSim, various tactics, network topologies, and resource management rules may be investigated in order to maximise the network's performance.

Dispersed Video Encoding

For more effective analysis, decentralized video processing is used, enabling each camera to handle its video stream locally.

Coordinated CCTV Monitoring

Goal: Arrange various cameras with distinct fields of vision (FOVs) in order to efficiently monitor a specified region.
Procedure: In order to obtain the best possible picture of the region, coordination entails synchronising the adjustment of Pan-tilt-zoom (PTZ) settings.

Tracking and Detecting Motion

Motion inside the field of view of smart cameras triggers the start of video feeds to the smart car parking application. The program recognizes and follows moving car from parking spaces in the video stream, continuously modifying the PTZ settings to provide the best possible tracking.

Notification of Events and User Alert

Detection: The system looks for anomalous occurrences in the video streams that need to be attended to.
User Notification: The user receives collected video streams via the Internet and is notified via the smart car parking program.

In order to address issues with centralized techniques, this system architecture places a strong emphasis on decentralized processing. In order to create a more responsive and successful network, the emphasis is on the real-time coordination of cameras, effective monitoring of moving objects, and timely warning of unusual events. The iFogSim simulation of this smart car parking system makes it easier to experiment with different resource management and deployment tactics for best results.

Application Model: camera, image capture, Object detection, alarm Control(PTZ), and display screen are the five main processing-intensive modules that make up the smart car parking programme 12. Numerous CCTV cameras transmit real-time stream to the program, with camera's motion control continually modifies the PTZ settings. The modules described includes the below functionalities:

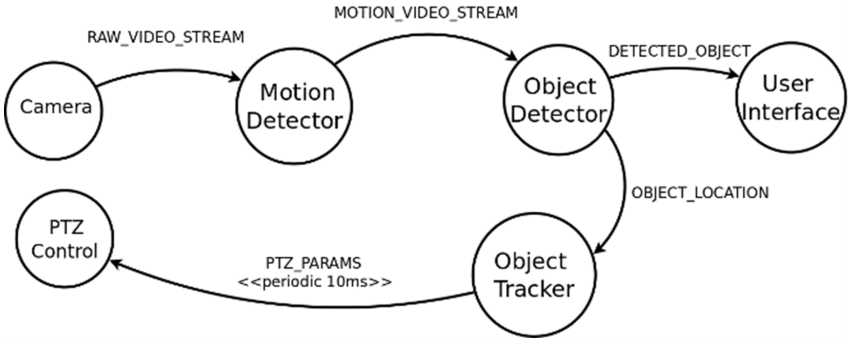

Figure 12: Application Model

Detecting Motion The intelligent camera utilized for this example have this module built into them.
Functionality: To detect object motion, it continually examines the unprocessed video streams that the camera records.
Action on Detection: The real time camera stream forwarded to the Object Detection module to process stream when any object movement is observed inside the camera Field of View (FOV).

Object Detection The object detection module is tasked with receiving video signals from smart cameras that have picked up motion from objects.
Processing: The items in motion from the video frames are isolated and then compared with already identified entities present in the previous frames.
Tracking Activation: Coordinates are computed and tracking for the identified object is started if it is new to the region.

Object Monitoring The Object tracker module receives the latest coordinated of the tracked object.
Optimization: To efficiently capture the tracked objects, it establishes the best Pan-tilt-zoom (PTZ) arrangement for each camera positioned throughout the region.
Information Conveyance: The PTZ control of cameras receives information from time to time.

PTZ Regulator This module controls the physical camera of each smart camera, adjusting it to fit the ideal PTZ settings provided by the Object Tracker module.
Function: system actuator that is integrated within the smart cameras themselves.

Interface User By delivering filtered video streams that include every monitored item to the user's device, programme creates an interface. The input stream filtered by the object detector are used in this scenario.

Table 13 describes the attributes of the application modules.

Tuple Type	CPU Length	N/W Length
RAW_ VIDEO_ STREAM	1000	20000
MOTION_ VIDEO_ STREAM	2000	2000
DETECTED_ OBJECT	500	2000
OBJECT_ LOCATION	1000	100
PTZ_ PARAMS	100	100

Figure 13: Inter-module edges

7 Evaluation

In this part, application case study is simulated in a fog computing environment, and two placement techniques (cloud-only and edge-ward) are evaluated according to metrics such as latency, network utilization, and energy consumption. Furthermore, the Network consumed and execution time of iFogSim are evaluated in relation to its scalability in a variety of simulation situations.

To demonstrate the adaptability of iFogSim, the smart car parking application is evaluated on a range of physical infrastructure configurations. There are somewhere from one and sixteen monitored zones, and each one has four smart cameras installed. Configurations (1 - 5) includes one, two, four, eight, and sixteen regions that are monitored and are among the simulated scenarios. Every monitored region include four intelligent imaging device linked with an access point that handles Internet access. Table 14 lists the network latency between devices.

The network is organized so that smart cameras are at the network edge while cloud center is on top. Smart cameras receive live video feeds as tuples and use them to identify motion. The cameras' PTZ control is represented by an actuator model. There are two approaches used for deploying modules in application: cloud only and edge ward.

Cloud Only Deployment: Except the object detection module, all others are present on the cloud.

Source	Destination	Latency (ms)
Camera	Area Switch	2
Area GW	ISP Gateway	2
ISP Gateway	Cloud DC	100

Figure 14: Intelligent Surveillance physical topology

Edge Ward Deployment: Cameras in a monitored area are connected to the Internet via Wi-Fi gateways that are equipped with the Object Detector and Object Tracker modules.

The iFogSim smart car parking simulation provides valuable insights into resource consumption by the application. On further break down, the key components of the output are:

- ***Execution Time*** is representing the total time take by the application to execute the simulation completely. Ideally the execution time should be short as it indicates quick turnaround times and real-time analysis.
- ***Loop Delays*** is indicating the time taken for each stage in the application including all tuples. Potential bottleneck can be identified by monitoring loop delay value for optimization in the area.
- ***CPU execution Delay*** shows delay in execution for any specific tuple. Analysing execution delays can be crucial for identifying critical components in the application and their optimization.
- ***Energy Consumption*** indicates the amount of energy consumed by each fog device, fog and cloud included. The cloud devices and various fog devices with their respective energy consumption level are included. This parameter is essential for monitoring the efficiency of the deployed services which indicates lower energy levels are expected as more energy efficient application.
- ***Network Usage*** indicates the network used by the modules and tuples while execution and how it is affected by the number of devices connected. Lower network usage is better as it can help in monitoring and identifying communication overhead between devices.

Energy Consumption

When the application switches to Fog devices, the amount of consumed energy in the cloud data centers decreases. This decrease remains consistent with the advantages of fog computing, which maximize resource efficiency and lessen the burden on centralized infrastructure.

As the number of regions under monitoring increases, cameras—which detect motion in recorded video frames—consume more energy 15. This association draws attention to the energy requirements of motion detection algorithms.

The investigation highlights Fog-based deployment's energy-efficient benefits, especially in situations when monitoring needs are higher.

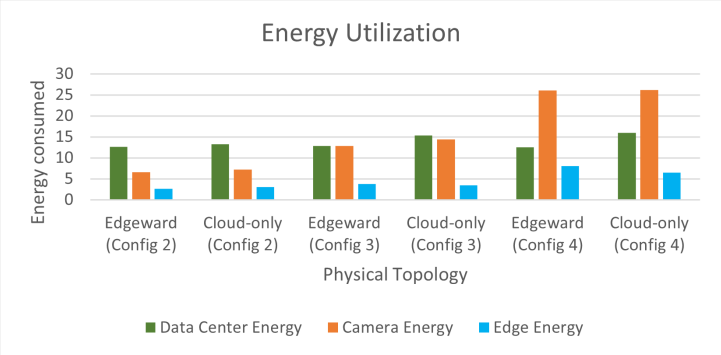

Figure 15: Energy Consumption

Simulation Time

Figure 16 shows the simulation times for a range of topologies and input workloads, illustrates how the number of devices and transmission rate grow with execution time. Even with a sizable number of gates added, simulation can still be completed in a respectable amount of time—12 seconds—because the simulation's growth is almost linear.

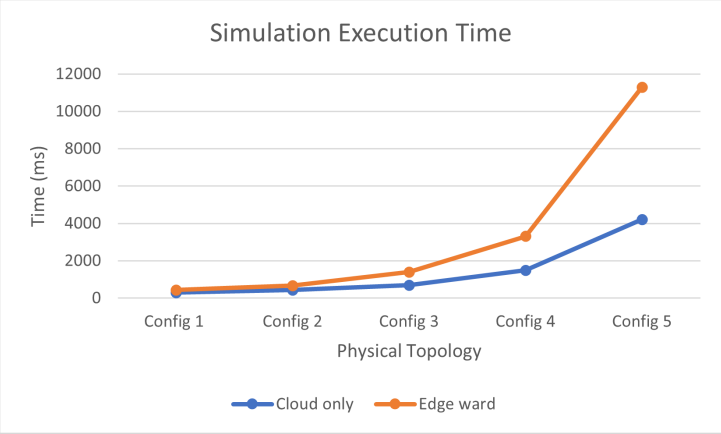

Figure 16: Simulation Time

Network Utilisation

Figure 17 represents the utilization of network for both the deployments, cloud only and edge ward. A significant increase in the load on the network can be observed as the number of connected devices are increased, especially in the case of cloud only deployment. However, it is clear that the edge ward deployment handles the load more effectively and seems more reliable in controlling network traffic.

7.1 Discussion

Promising approaches to tackle data processing issues in the Internet of Things (IoT) include fog and edge computing. Fog and Edge computing, in contrast to conventional cloud-centric methods, make use of devices at the network edge to lower latency and

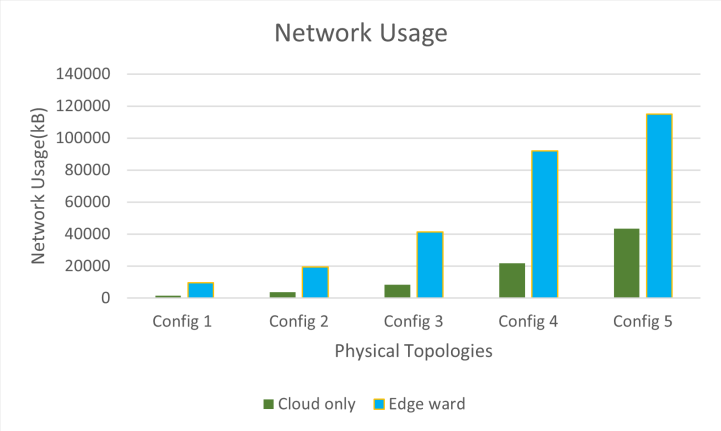

Figure 17: Network Utilization

congestion. In this study, iFogSim—a simulation tool used for modelling and simulation of environments in Fog and Edge, and IoT applications—was described. With iFogSim, resource management strategies based on Quality of Service (QoS) parameters—like latency—can be examined and contrasted for a range of workload scenarios.

The case studies that were presented showed how well iFogSim evaluated resource management strategies, such as placing application modules exclusively in the cloud and pushing applications towards edge devices. The simulation's scalability was confirmed, demonstrating iFogSim's potential to facilitate IoT-scale simulations.

8 Conclusion and Future Work

Addressing the challenges related to task and resource scheduling in fog and edge computing is essential for optimizing resource utilization and increase the efficiency of the application. The research work provides valuable insights into existing strategies and underscore the significance of energy consumption as a crucial factor. By evaluating various qualitative factors, this work guides the future research in the direction of fog computing task scheduling efficiently.

Policies for Power-Aware Resource Management: It's critical to address the problem of prolonging the battery life of Fog devices. Future research might investigate rules that dynamically migrate operators based on the battery life of devices.

Strategies for Priority-Aware Resource Management in Multi-Tenant Environments: A potential avenue of study is to examine scheduling rules for situations where various application instances share the same pool of resources but have varying Service Level Objectives (SLO).

Modelling Fog Device Failures: Research might concentrate on obtaining failure models for common IoT and fog device malfunctions. Subsequently, these models may be employed to assess and contrast scheduling and recovery strategies that consider dependability for diverse applications. Examining combined Edge-Network resource scheduling is an important issue to research in Internet of Things settings, where heterogeneous network and sensing resources are shared among several applications with different Quality of Service needs.

Modelling and Comparing various techniques of virtualization for fog Environments:

Upcoming research works might take into account for evaluating overall effectiveness of different techniques of virtualization, such as full virtualization, operating system-level virtualization like containers, and para-virtualization (instances of hardware-level virtualization).

References

Alizadeh, M. R., Khajehvand, V., Rahmani, A. M. and Akbari, E. (2020). Task scheduling approaches in fog computing: A systematic review, *Digital Communications and Networks* .

Baccarelli, E., Naranjo, P. G. V., Scarpiniti, M., Shojafar, M. and Abawajy, J. H. (2017). Fog of everything: Energy-efficient networked computing architectures, research challenges, and a case study, *IEEE Access* **5**: 9882–9910.

Bellavista, P., Berrocal, J., Corradi, A., Das, S. K., Foschini, L. and Zanni, A. (n.d.). A survey on fog computing for the internet of things, *Pervasive and Mobile Computing* **52**.

Beloglazov, A. and Buyya, R. (2010). Energy efficient allocation of virtual machines in cloud data centers, *2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing*.

Ben Sada, A., Naouri, A., Khelloufi, A., Dhelim, S. and Ning, H. (2023). A context-aware edge computing framework for smart internet of things, *Future Internet* .

Dastjerdi, A. V. and Buyya, R. (2016). Fog computing: Helping the internet of things realize its potential, *Computer* .

Durillo, J. J., Nebro, A. J. and Alba, E. (2010). The jmetal framework for multi-objective optimization: Design and architecture, *IEEE Congress on Evolutionary Computation*.

ETSI-Industry-Specification-Group (2020). Multi-access edge computing (mec); framework and reference architecture, *ETSI GS MEC 003 V2.2.1 (2020-12)*.

Feng, C., Han, P., Zhang, X., Yang, B., Liu, Y. and Guo, L. (2022). Computation offloading in mobile edge computing networks: A survey, *Journal of Network and Computer Applications* .

Hu, P., Dhelim, S., Ning, H. and Qiu, T. (2017). Survey on fog computing: architecture, key technologies, applications and open issues, *Journal of Network and Computer Applications* .

Luan, T. H., Gao, L., Li, Z., Xiang, Y., Wei, G. and Sun, L. (2015). Fog computing: Focusing on mobile users at the edge, *Networking and Internet Architecture*.

Mahmud, R., Kotagiri, R. and Buyya, R. (2018). *Fog Computing: A Taxonomy, Survey and Future Directions*, Springer Singapore, Singapore.

Mahmud, R., Pallewatta, S., Goudarzi, M. and Buyya, R. (2022). ifogsim2: An extended ifogsim simulator for mobility, clustering, and microservice management in edge and fog computing environments, *Journal of Systems and Software* .

Mao, Y., You, C., Zhang, J., Huang, K. and Letaief, K. B. (2017). A survey on mobile edge computing: The communication perspective, *IEEE Communications Surveys Tutorials* .

Mehrabi, M., You, D., Latzko, V., Salah, H., Reisslein, M. and Fitzek, F. H. P. (2019). Device-enhanced mec: Multi-access edge computing (mec) aided by end device computation and caching: A survey, *IEEE Access* .

Mehran, N., Kimovski, D. and Prodan, R. (2019). Mapo: A multi-objective model for iot application placement in a fog environment, *Proceedings of the 9th International Conference on the Internet of Things*, IoT '19, Association for Computing Machinery, New York, NY, USA.

Ngatchou, P., Zarei, A. and El-Sharkawi, A. (2005). Pareto multi objective optimization, *Proceedings of the 13th International Conference on, Intelligent Systems Application to Power Systems*, pp. 84–91.

Pradhan, P., Behera, P. K. and Ray, B. (2016). Modified round robin algorithm for resource allocation in cloud computing, *Procedia Computer Science* .

Rani, R. and Garg, R. (2021). Pareto based ant lion optimizer for energy efficient scheduling in cloud environment, *Applied Soft Computing* .

Sabella, D., Vaillant, A., Kuure, P., Rauschenbach, U. and Giust, F. (2016). Mobile-edge computing architecture: The role of mec in the internet of things, *IEEE Consumer Electronics Magazine* .

Shi, W., Cao, J., Zhang, Q., Li, Y. and Xu, L. (2016). Edge computing: Vision and challenges, *IEEE Internet of Things Journal* .

Shu, W. and Li, Y. (2023). Joint offloading strategy based on quantum particle swarm optimization for mec-enabled vehicular networks, *Digital Communications and Networks* .

Wang, Y., Yu, T. and Sakaguchi, K. (2021). Context-based mec platform for augmented-reality services in 5g networks, *2021 IEEE 94th Vehicular Technology Conference (VTC2021-Fall).*

Yadav, R., Zhang, W., Kaiwartya, O., Song, H. and Yu, S. (2020). Energy-latency tradeoff for dynamic computation offloading in vehicular fog computing, *IEEE Transactions on Vehicular Technology* .

Yi, S., Hao, Z., Qin, Z. and Li, Q. (2015). Fog computing: Platform and applications, *2015 Third IEEE Workshop on Hot Topics in Web Systems and Technologies (HotWeb).*

Zhang, K., Mao, Y., Leng, S., Zhao, Q., Li, L., Peng, X., Pan, L., Maharjan, S. and Zhang, Y. (2016). Energy-efficient offloading for mobile edge computing in 5g heterogeneous networks, *IEEE Access* .

Zhang, Q., Cheng, L. and Boutaba, R. (2010). Cloud computing: State-of-the-art and research challenges, *Journal of Internet Services and Applications* .

Zhao, M., Yu, J.-J., Li, W.-T., Liu, D., Yao, S., Feng, W., She, C. and Quek, T. Q. S. (2021). Energy-aware task offloading and resource allocation for time-sensitive services in mobile edge computing systems, *IEEE Transactions on Vehicular Technology* .