A Hybrid load balance optimization model for Cloud IoT Edge

A

Thesis

Submitted for the Award of the Degree of

DOCTOR OF PHILOSOPHY

In

COMPUTER SCIENCE AND ENGINEERING

Submitted By

K RAGHAVENDAR

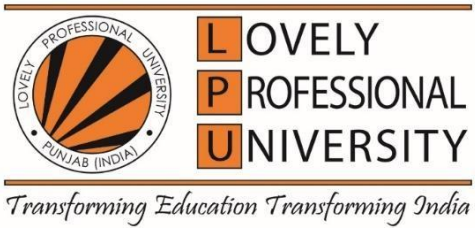
41900441

Supervised By

Dr Isha Batra(17451)

School of Computer Science and Engineering (Professor)

Lovely Professional University



LOVELY PROFESSIONAL UNIVERSITY, PUNJAB

2024

CANDIDATE'S DECLARATION

I hereby declare that the thesis entitled ***A Hybrid load balance optimization model for Cloud IoT Edge*** submitted for the 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 Supervisor Dr. Isha Batra, Professor, School of Computer Science and Engineering, Lovely Professional University, Punjab. This work has not been submitted for the award of any degree, diploma, associateship, and fellowship of any University or Institution.

Date: **Investigator**

K Raghavendar

School : Computer Science and Engineering

University Lovely Professional University

Punjab, India

CERTIFICATE

This is to certify that the thesis entitled *"***A Hybrid Load balance optimization model for Cloud IoT Edge***"* submitted by **K Raghavendar** for the award of 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 diploma, according to the best of my knowledge.

Signature of Advisor

Date:

Name: Dr. Isha Batra

Designation: Professor

School : Computer Science and Engineering

University : Lovely Professional University Punjab ,India

ABSTRACT

Virtual Machines (VMs) are primarily created and deployed to enable batch and stream processing applications using cloud computing and storage services. The requirement to provision distinct resources for every virtual machine (VM) drives up maintenance expenses even if this approach offers thorough monitoring of the underlying infrastructure. For adaptability and optimisation, even with automated solutions, careful capacity planning and manual intervention are needed. Because of their lower overhead and increased flexibility over virtual machines (VMs), containers are becoming the standard cloud processing unit as the industry moves towards addressing these issues.

This progress is best illustrated by the Internet of Things (IoT), whose incorporation into daily life is demonstrated by gadgets like autos with sophisticated detectors or cellphones with heart monitors. The Internet of Things (IoT) includes any entity or person that may send data over a network and be uniquely identifiable by an IP address. IoT operates in harmony in this scenario, with the cloud acting as the main data centre. To keep the connection between cloud and edge IoT devices steady and productive, load balancing and optimisation techniques that work are essential.

The analysis of massive data sets produced by the healthcare sector has gained importance in recent years. Techniques for large-scale data mining, especially the MapReduce paradigm, have shown to be successful. Big data processing is less efficient when there is data skewing, which is still a major problem. We suggest Skew Handling Based on Partition Tuning (PTSH) as a solution to reduce data skew problems in MapReduce. In contrast to the traditional one-stage MapReduce model, PTSH distributes key pairs more efficiently by using a two-stage algorithm and division optimisation technique. Partitions are dynamically changed in the event of data skew in order to preserve balance. PTSH proved to be more efficient and reliable than Hadoop's built-in fragmentation techniques when tested on a variety of real-world and simulated medical datasets.

The efficiency of the method is further increased by equitable and proximity-based partitioning, which greatly boosts MapReduce task performance. The partitioning tuning skew management approach significantly reduces the amount of time needed for rule extraction, making it especially well-suited for rule mining in healthcare data.

Issues with class imbalance arise when artificial intelligence (AI) is applied, particularly in cases where data is uneven and high dimensional.

In such cases, traditional feature selection methods frequently produce subpar results since they give samples from different classes the same weights.

When classifications costs differ, cost-effective learning strategies are used. To address these issues of class imbalance, a number of formal processes have been devised.

IoT networks have a difficult time guaranteeing a high quality of end-user experience since they depend on cutting-edge technology like sensors, controllers, GSM, UMTS, RFID, and 3G networks. These networks are expanding because cloud technologies are integrating processing power, network bandwidth, virtualization, and system software. Reducing operating expenses and power consumption, maintaining load balance, avoiding SLA violations, and improving machine performance all depend on effective capacity management. Tackling these problems requires strong resource management and decision-making based on IoT data.

This study looks at several resource provisioning techniques and pinpoints important elements for distributed system resource optimization. The main things that need to be improved are the approximation accuracy, data skew rate, and minimization rate. In order to allocate cloud resources efficiently, hybrid optimization presents a number of problems and complications that are further highlighted by the particular needs of the IoT ecosystem.

In addition, we suggest a sophisticated organisational framework that will allow for smooth connectivity and ideal load balancing between cloud and edge IoT devices. This architecture incorporates dynamic resource allocation algorithms that anticipate and adapt to changing demands by utilising real-time data analytics. Through the integration of machine learning models, the system is able to optimise resource distribution continually, reducing latency and increasing throughput.

The significance of hybrid optimisation strategies in handling the complex interaction between cloud and edge computing resources is highlighted by our findings. These methods guarantee scalability and stability in addition to improving system performance—two factors that are essential for sustaining the rapidly growing Internet of Things environment. Large-scale data processing in the healthcare and other industries can be made much more effective and efficient by resolving problems with data skew and class imbalance through creative segmentation and machine learning techniques.

The study shows that in order to handle the increasing needs of Internet of Things applications, a hybrid architecture that combines cloud and edge computing with sophisticated load balancing and resource management tactics is necessary. This strategy guarantees the best possible use of available resources, lowers operating expenses, and improves the overall functionality and dependability of IoT networks.

.

ACKNOWLEDGEMENT

I would like to present my deepest gratitude to **Dr. Isha Batra** for her guidance, advice, understanding and supervision throughout the development of this thesis and study. Despite her busy schedule he has been available at every step, devoting time and energy and the much needed counsel and advice. This enabled me to sail through the tough times and complete this enormous task.

I would like to thank to the **Research project committee members** for their valuable comments and discussions. A special thanks to the management of **Lovely Professional University** for their support in academic concerns and letting me involve in research study. The doctoral programme of LPU has made it possible for me to pursue my dream of research and upgrade my knowledge.

My sincere feeling of gratefulness also goes to my parents and family members who always motivated me in all the endeavors of my life including this research work in LPU. Finally, I would like to thank each and every person who has directly and indirectly helped and motivated me in this journey.

K Raghavendar

LIST OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

Abbreviations	Description
IoT	Internet of Things
SaaS	Software as a service
AWS	Amazon Web Services
PaaS	PaaS
IaaS	Infrastructure as a Service
CPU	Central processing unit
EC2	Elastic Compute Cloud
SQL	Structured Query Language
API	Application Programming Interface
IBM	International Business Machines
MEC	Multi-access edge computing
CAME	Computer Aided Market Engineering
QoS	Quality of Service
ORP-HS	Optimal Resource Provisioning- Harmony search
EC	Embedded Controller
VM	virtual machine
CoT	Cloud of Things
PTZ	Pan-tilt-zoom
DPTO	Delay-dependent priority-aware task offloading
EN	Enterprise Network
OSPF	Open Shortest Path First
NCN	Naval Communications Network
MECOM	Multi-Access Edge Content and Measurement Computing
ILP	Integer Linear Programming
SFC	System File Checker
NFV	Network functions virtualization
DG	Distributed Generation
PV-APF	Adaptive Project Framework-Photovoltaic System
CNN	Convolutional Neural Network
EEDOA	Electrical Engineering Digital Object Architecture

EEG	electroencephalogram
FES	Functional electrical stimulation
BCI	Brain-computer interface
DVS	Descriptive Video Service
SCD	Sequential Compression Device
CTOSO	collaborative task offloading scheme withserviceorchestration
DRL	Daytime running lamps
LBMM	Load Balance Min-Min
OLB	opportunistic load balancing
VMM	virtual machine migration
GIS	Geographic Information System
MMHHO	Mantaray adapted multi-objective Harrishawk optimization
MRFO	Manta Ray Forge Optimization
HHO	Harris Hawk Optimizing
PHEV	Plug-in hybrid electric vehicles
MPSO	Memetic Particle Swarm Optimization
ACO	Ant colonies optimization
RAM	Random Access Memory
FCFS	First Come First Serve
LEEN	Learning Energy Efficiency Network
EMR	Elastic MapReduce
MQL	Monitoring Query Language
ADN	Assessment Designated Node
CMDH	Communications and Delivery Handling
HTTP	Hypertext Transmission Protocol
XML	Extended Markup Language
AICLOUD	Artificial intelligence CLOUD
AUROC	Area under the receiver operating characteristics
MI	Mobile internet
AP	Advanced placement
RP	Recursive approach
SERA	selectively recursive approach
REA	Recursive Ensemble Approach

SMOTE	Synthetic Minority Over-sampling Technique
AI	Artificial intelligence
RFID	Radio-frequency identification
UMTS	Universal Mobile Telecommunications Service
GSM	Global System for Mobile communication
SLA	service-level agreement
QoS	Quality of Service
EPCs	Electronic Prescribing of Controlled Substances
LTE	Long Term Evolution
NFC	Near Field Communication
WSN	wireless sensor network
RDF	Resource Description Framework
OWL	Web Ontology Language
SQL	Structured Query Language
DDS	*Digital Data Services*
D2D	Diploma to Degree
M2M	Machine to Machine communications
MTDs	Medium-Term Debt Management Strategy
ECIoT	Edge-Centric IoT
HTDs	*Hi-Tech DETECTION SYSTEMS*
MILP	*Mixed-integer linear programming*
BGA	Ball Grid Array
RPL	Recognition of prior learning

CHAPTER 1

INTRODUCTION

This section discusses the concept of an IoT cloud, which encompasses a large network providing services to IoT devices and applications. IoT devices have the capacity to collect, process, analyze, and store data on a significant scale. The integration of IoT and cloud computing technologies streamlines processes related to data analytic s and cost management. Notably, consumer analysis stands out as a key factor driving the adoption of IoT, especially due to its advantageous implications for contemporary businesses.

1.1 Introduction

Cloud computing provides access to data and information from a centralized pool of computer resources that can be purchased and utilized as needed. Three fundamental types are commonly employed to describe cloud systems: public, private, or hybrid, as depicted in Figure 1.3 [7] represents. Public clouds offer services over the internet, allowing users to access resources provided by third-party providers. Private clouds are dedicated to a single organization, providing enhanced control and security over data. Hybrid clouds combine elements of both public and private clouds, offering flexibility and scalability by enabling data and applications to be shared between them. Organizations must understand these distinctions when selecting the most suitable cloud infrastructure to meet their specific needs and objectives.

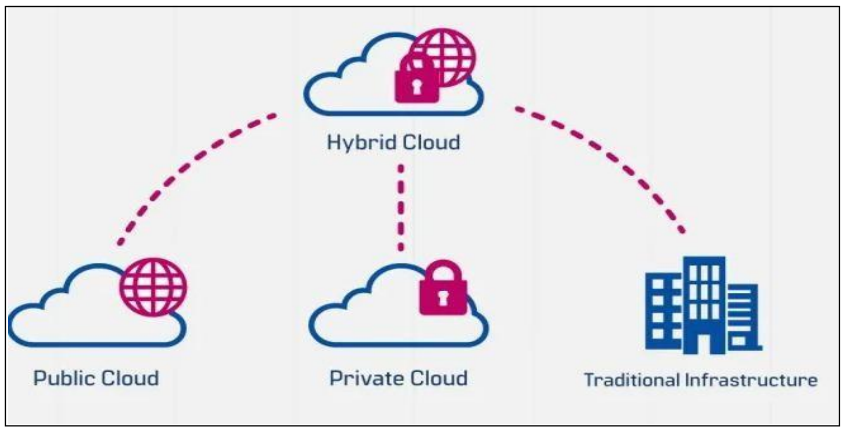


Figure 1.1: Hybrid cloud computing

The term Internet of Things also refers to how devices communicate with information that is not generally accessible, such as computers. You may add sensors to automobiles, kitchenware, and other items via the Internet of Things. One transition enabler is IoT. As a result, facilities and processes are automated, enhancing the effectiveness and intelligence of real-time monitoring and control. Because all relevant information is immediately available (in real-time, even with historical trend data), there are opportunities for creative data aggregation and analysis, which may lead to better decision-making under rapidly changing conditions. IoT and cloud computing work together to increase the effectiveness of routine tasks. The enormous amounts of data the IoT generates may be channeled through cloud storage [8]. Because most data centers operate on a pay-per-use model, you will only be charged for the cloud storage you use. Through more prominent corporations, data centers may also be able to finance lower total costs for IoT start-ups and companies on a smaller scale. Cloud computing, which promotes better cooperation, is another advantage of IoT big data. Improved teamwork is necessary for creative technologies. Developers can rapidly access data and communicate when data storage and display are made easy.

Storing data in the cloud and assigning resources to various regions enables IoT organizations to respond more efficiently. The cloud has evolved as the favoured platform for significant data expansion in recent years. Many organizations may now easily access the massive volumes of big data on the Internet. The concept of cloud computing involves using salable services on-demand through the Internet. In cloud computing, many physical and virtual servers are employed as needed. Cloud computing takes advantage of massive processing power and storage capacity to address issues like evaluating investment options and risk in financial portfolios, giving personalized medical information, powering computer games, and many more. The cloud combines virtualization, SOA (service-oriented architecture), and web services.

One of the fundamental principles that support the idea of cloud computing is the re-usability of IT abilities. For decades, businesses have been devoting time and resources to building infrastructure that may provide them with a competitive edge, and cloud computing has now made that possible. With cloud computing, idle processing power may be efficiently used and offered to people or companies in line with their needs. The transition of computers and IT infrastructure into a service that could be made accessible to everyone is referred to as the cornerstone of cloud computing. Three different models comprise cloud computing.

i. Software as a Service (SaaS)
ii. Platform as a Service(PaaS)

iii. Infrastructure as a Service (IaaS).

i) Software as a Service (SaaS)

It also goes by the titles service and application. Clouds execute specialized business operations and company processes with access to specific cloud capabilities rather than just providing cloud features. In a nutshell, they provide software and services that use cloud platforms or infrastructure. Clouds frequently offer a particular class of popular application characteristics. Even if it allows for better capabilities, cloud computing goes beyond only systems that offer platforms, infrastructure, and software as a service. Therefore, IaaS, PaaS, and SaaS may be viewed as specific use patterns for cloud systems connected to the current models utilized by Grid and Web Services and other systems. Cloud-based solutions are promising choices for the implementation and growth of these concepts.

ii) Platform as a Service (PaaS)

Through a platform, it offers computer resources that may be utilized to create and host applications and services. When the server's host engine performs software and repeats those operations in reply to user requests (for instance, access rate), PaaS usually uses specific APIs to control those actions. Applications created for something like the particular supplier of clouds can't be transferred to a different cloud server because every cloud provider reveals their API following the relevant vital capabilities; attempts are made, however, to integrate cloud features into generic models of programming (such as MS Azure).

Resource clouds provide consumers access to controlled, scaled resource-like services, boosting virtualization capabilities. As a result, a service interface may make various resources accessible. Data Storage Clouds deal with trustworthy data access and may have varying sizes to balance resource consumption with accessibility demands or excellence standards. Examples are Amazon S3, SQL Azure, and Compute cloud providers, which frequently offer the ability to deliver computing resources, often visualized, to run cloud-related services and applications. Compared to a plain computing service, infrastructure as a service provides additional functionality.

Using cloud servers encourages energy savings due to the absence of energy, the ongoing environmental changes worldwide, and the rising infrastructure expenses. Cloud computing still needs hundreds or thousands of machines connected over the Internet, even though software and infrastructure are beneficial and cost-effective. These servers must constantly be in operation mode to satisfy users' requests. The predictor restarts any inactive servers as necessary. These benefits will reduce the project's overall cost and result in energy savings. The power needed to run these servers is considerable. It has been observed that idle servers

use more energy than active ones. The quantity of wasted power may be reduced using various techniques, such as voltage adjustment, CPU speed change, display deactivation, sleep mode, server shutdown, etc.

The fifth-generation cloud computing technology enables the use of IT resources through the Internet. Resources include things like platforms, infrastructure, software, and hardware. Many businesses provide cloud-based computing services, including Microsoft Azure and Amazon's Amazon EC2 (Elastic Compute Cloud) offering in the form of Google App Engine. Google also offers its Amazon Elastic Compute Cloud (EC2) cloud computing service.

iii). Infrastructure as a Service (IaaS)

Infrastructure as a Service (IaaS) is a cloud computing model that offers visualized computing resources over the internet. With IaaS, businesses can rent visualized servers, storage, networking, and other fundamental computing resources on a pay-as-you-go basis. This model provides users with flexibility and scalability, allowing them to scale their infrastructure up or down according to their changing needs without the need for physical hardware investments. IaaS providers manage the underlying infrastructure, including data centers, servers, and networking hardware, while users have control over operating systems, applications, and development frameworks.

1.2 Generalized Cloud computing Architectural Design

As depicted in the image above, the primary feature of cloud computing is the centralization of data and applications on servers rather than individual users' computers. In this setup, the actual programs or applications reside on servers connected to the user's PC, with data transmitted between the server and the user's device. The back end infrastructure comprises servers, which may be housed in facilities similar to the user's location or elsewhere. This centralized architecture enables users to access computing resources and data remotely, offering flexibility and scalability. Additionally, it reduces the burden on individual devices, allowing for more efficient resource utilization and more accessible software and data management. This shift towards centralized computing marks a significant paradigm shift in how computing resources are accessed, utilized, and managed.

Cloud computing allows a programme to run anywhere in the cloud; ideally, the user is only concerned with the applications available, not with the underlying technology or the specific location of the computer hosting the programme. A user's PC and a server are connected over the internet. The servers that house a particular application are selected according to the application's programming, which ensures maximum performance by dividing the burden

across servers equally.

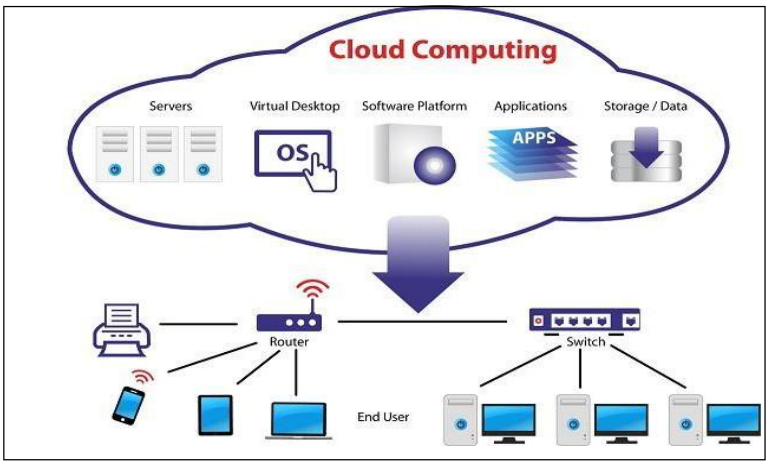


Figure1.2 Cloud Computing Fundamental Architecture [7]

1.2.1 Self-service offered on request

Customers can readily access computer hardware resources such as CPU time, network storage, software usage, etc., automatically (i.e., self-serve) at any given time without engaging with the individuals who provide such services directly.

1.2.2 Wide-scale network access

Multiple client applications, utilizing various heterogeneous systems such as mobile devices, laptops, or personal digital assistants, can access these computer services across a network, facilitating access for customers at their respective locations, including but not limited to the Internet.

1.2.3 Pooling of resources

A cloud service provider's computing resources can be used using the hypervisor approach or multi-tenancy, where different physical and visualized resources are regularly allocated and redistributed following client needs. Adopting such a pool-based computer architecture is motivated by economies of scale and specialization. A pool-based paradigm makes users who usually have no influence over or understanding of the locations, generation, and originality of these resources invisible consumers of physical computation resources (such as databases, CPUs, etc.). For instance, customers cannot choose where their data will be kept in the cloud.

1.2.4 Dynamic Resource Management

Consumers are freed from the obligation of upfront commitments or contractual agreements for computer resources, given their ability to scale usage as required. This flexibility enables

swift adjustments to meet fluctuating demands, eliminating the need for preemptive resource allocation. With the assurance of resource availability, consumers can readily increase consumption to accommodate peak demands. This dynamic scalability enhances operational efficiency and optimizes resource utilization, aligning resource allocation precisely with usage patterns. As a result, businesses can maintain agility and responsiveness in their operations, efficiently managing resource utilization while adapting to evolving demands in real-time.

1.2.5 Measured Service

The cloud infrastructure utilizes effective methods to monitor the usage of resources for each user individually through its metering capabilities, even when computing resources are shared among multiple customers (i.e., multi-tenancy). Currently, three primary types of clouds are in use, which vary depending on the services they offer.

1.3 Cloud Deployment Models

The following deployment types can be separated in the cloud, similar to PaaS/IaaS/SaaS:

a) Private Clouds

In most cases, the pertinent company either owns or rents them. As far as the user is concerned, this is similar to Software as a Service (SaaS), even if services with cloud-enhanced capabilities may occasionally be offered. This is because the client does not have direct access to certain functionality. Businesses can use cloud services from third parties or offer clients outside the corporation their services. Companies may outsource their services to these cloud providers and save money and time on developing their infrastructure by allowing consumers to use cloud features for their needs. For instance, Amazon, Google Apps, and Windows Azure.

b) Public Clouds

Public clouds give up control over how resources are allocated and how data and code are maintained, even while they allow enterprises to outsource portions of their infrastructure to cloud providers. The specific company does not desire this in some situations. Hybrid clouds combine the use of both public and private cloud infrastructures to outsource work while keeping some control over sensitive data and attempting to save expenses as much as feasible. There aren't many hybrid clouds in use right now, despite early initiatives like the ones by IBM and Juniper already presenting essential technologies for their deployment. Because of the energy crisis and environmental changes, every electricity sector today is battling the contentious issue of power use. Hundreds of servers are running at once in cloud computing, but only a tiny portion of them are being used to handle clients' or users' requests, most of which are idle. The Green Scheduling Algorithm reduces these idle servers' power usage. In

addition to shutting down the inactive servers, the present method balances the added stress on Identifying which servers are inactive and manually shut them down might be pretty challenging. The recommended solution for sustainable energy scheduling method savings in cloud computing will be developed in C#. In this system, client-server communication is predicted by a neural network. The predictor predicts future demand and shuts down idle servers automatically.

Cloud computing is the idea of using scalable services in an online environment on demand over the Internet. In cloud computing, many actual and virtual servers are employed as required. Cloud computing takes advantage of massive processing power and storage capacity to address issues like evaluating investment options and risk in financial portfolios, giving personalized medical information, powering computer games, and many more. To provide the services consumers need, cloud computing uses a sizable number of computers. The work scheduling method [7] is employed for efficacy and a higher efficiency rate. While maintaining service quality, it uses fewer resources and offers significantly superior resource utilization rates.

One fundamental principle that supports the idea of cloud computing is the reuse of IT skills. Businesses have spent years and significant resources building infrastructure, and now pay-per-use cloud computing services are available. Unused processing power may be efficiently utilized and offered to people or enterprises using cloud computing according to their needs. The basis of cloud computing is the transformation of IT infrastructure and computing resources into a utility that can be made available to everyone.

The processing units become less efficient if assets are already being wasted, underused, or left idle in the cloud, making correct scheduling necessary. Therefore, using a task scheduling algorithm leads to improved scheduling, decreased resource utilization, increased resource usage and overall efficiency.

Routine duties are evolving due to the Internet of Things (IoT) proliferation. This technology facilitates the integration of various physical products, including vehicles, homes, and appliances, by incorporating hardware, software, sensors, and internet connectivity to process, transmit, and gather data. The IoT encompasses any artificial or natural object capable of transferring data over a network and can be identified by an IP address [1]. The expansive network of interconnected IoT devices, facilitated by the Internet, enables seamless communication through cost-effective sensors, heralding an era of unprecedented connectivity where billions of smart devices interact with humans.

Services online are required. International sectors, including construction, energy, healthcare,

and precision manufacturing, have all been influenced by IoT [2]. It's excellent news for computer scientists and system designers because internet service providers are constantly increasing their network options. Studies focus on IoT applications and advances. IoT systems and gadgets are already being created to accommodate future expansion [3] conveniently.

A) Cloud loT framework

An extensive, effective, and scalable system for handling and processing the massive volumes of data produced by connected devices is created by integrating IoT devices, edge computing, and cloud services into a cloud-based IoT (Internet of Things) architecture. The device, edge, and cloud layers make up the architecture's three main levels, and each is essential to the pipeline for processing and data flow.

i) **Architecture Components:** Device Layer: IoT devices, including sensors, actuators, and other connected hardware, collect data from the physical environment. These devices transmit data to the edge layer using various communication protocols like MQTT (Message Queuing Telemetry Transport), CoAP (Constrained Application Protocol), and HTTP (Hypertext Transfer Protocol).

ii) Edge Layer: The edge layer includes edge gateways and nodes that handle data preprocessing, aggregation, filtering, and preliminary analytics. By processing data closer to its source, this layer reduces latency and conserves bandwidth, ensuring faster response times and efficient use of cloud resources. Edge computing enables real-time processing and decision-making, which is crucial for time-sensitive applications. Cloud Layer: In the cloud layer, preprocessed data is sent to centralized cloud servers for advanced processing, storage, and analytics. The cloud platform leverages extensive computational resources to perform complex data analysis, machine learning tasks, and generate actionable insights. The cloud also provides scalable storage solutions to manage large volumes of data.

iii) Load Distribution Techniques:

The hybrid load balancing method used by the framework optimizes both resource utilization and performance. By allocating work in accordance with preset guidelines and standards, static load balancing guarantees a consistent and effective use of resources. By adjusting in real-time to shifts in workload, resource availability, and network circumstances, dynamic load balancing maximizes responsiveness and performance.

In Below Figure 1.3 represents illustrates the integration of IoT with cloud computing mechanisms. While increased machine connectivity alone may not directly benefit customers, the success of IoT hinges on devices joining the network and exchanging data globally.

Remote users can access crucial information through the cloud computing network [4], enabling flexibility in device usage, including digital phones across various platforms such as large screens and PCs. These advantages extend beyond business-owned equipment, benefiting a broader spectrum of users. However, the proliferation of IoT devices also substantially increases data generation, placing a significant strain on the internet infrastructure. The sheer volume of data generated necessitates robust network capabilities to ensure efficient transmission and processing, highlighting the importance of saleable and reliable internet infrastructure to support the evolving IoT ecosystem.

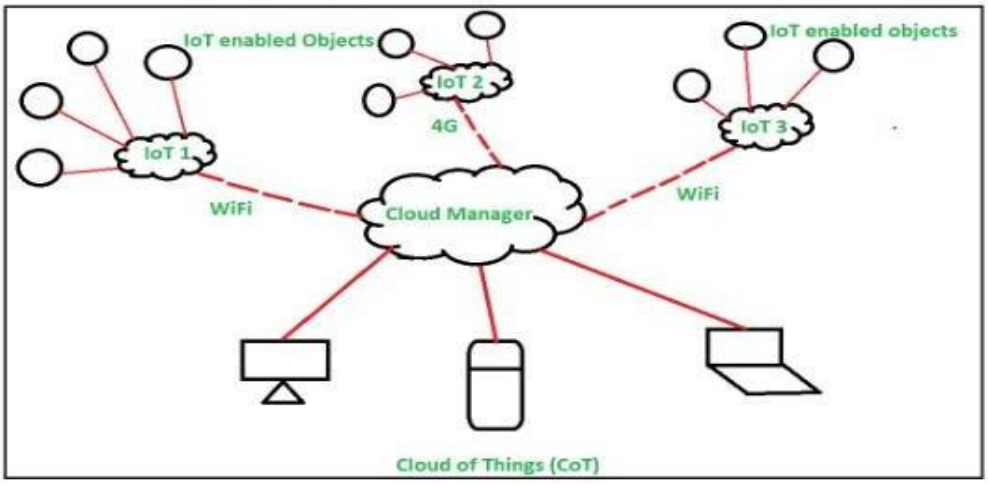


Figure 1.3: IOT with Cloud computing

Businesses must provide solutions to ease the burden by exchanging enormous volumes of data and fixing the issue. Cloud computing has entered the heart of IT and now provides scalability in delivering business applications and as a service platform (SaaS). Through development, both businesses move their software to the cloud. Most cloud service providers will need to send your data over your standard internet connection or another direct channel.

Metamorphosis itself is a recent phenomenon that is increasingly needed in today's fast-paced culture. The data can be evaluated and changed with the aid of technology in good time and a sophisticated new format. This scalability is made possible by the cloud, which has proven to be a dependable platform for data transfer over conventional Internet networks and a specialized direct connection link. Although the traditional method is less than ideal, many firms utilize a direct connection to the cloud transformation data at the same time [5] because of the excellent quality of the data and the security it provides throughout the transformation phase.

Almost entirely, the cloud has developed to play a sizable role in the network's environment. Simply put, the cloud may be seen as an IoT enabler. The cloud is undoubtedly, the best solution for any needs concerning corporate data-based. As a result of the technique's

development, programmers are better able to quickly design useful apps and establish more efficient information utilizing devices and the Internet.

1.3.1 Cloud Technology Support IoT

The overarching objective is to increase productivity while putting the security of the data being processed, transferred, or stored through cloud computing and IoT at risk. All services work well together because of the connection's opposing nature. The Internet of Things acts as this network; the cloud serves as their primary computer access point. We will surely undergo many changes as time goes on; some will be gradual, while others will be more noticeable [6]. As firms like Amazon, AWS, Google, and Microsoft establish themselves as the undisputed leaders in creating cloud-based IoT services, the rivalry may get much stiffer. As clouds increase in size and popularity and offer pay-per-use pricing to companies, many new cloud services are emerging. Fundamentally, it implies that businesses must pay to utilize computers. In Below Figure. 1.2 represents, cloud computing and IoT have been integrated,

1.4 Cloud computing integrated with Internet of Things

The Internet of Things (IoT) paradigm connects diverse gadgets and programmes. One of the critical goals of the Internet of Things is to cater to the specific requirements of end-users and convert the vast and varied data generated by these countless IoT devices into valuable insights [1]. Both industry and academia are intrigued by the potential of the IoT to enhance people's daily lives [2]. With the rapid advancement of IoT and mobile communication technologies, there has been a significant increase in the number of IoT devices and apps. This has made a wide range of user-friendly services available to end users [3,4].

Nevertheless, IoT devices' computational and energy demands have significantly increased [5,6]. Due to IoT devices' limited computing and energy capabilities, these applications cannot be efficiently serviced [7]. By leveraging cloud computing, IoT devices can offload computing tasks to powerful cloud servers, effectively addressing the limitations of computing resources [8, 9]. On the other hand, transferring tasks over long distances will result in considerable delays in transmission.

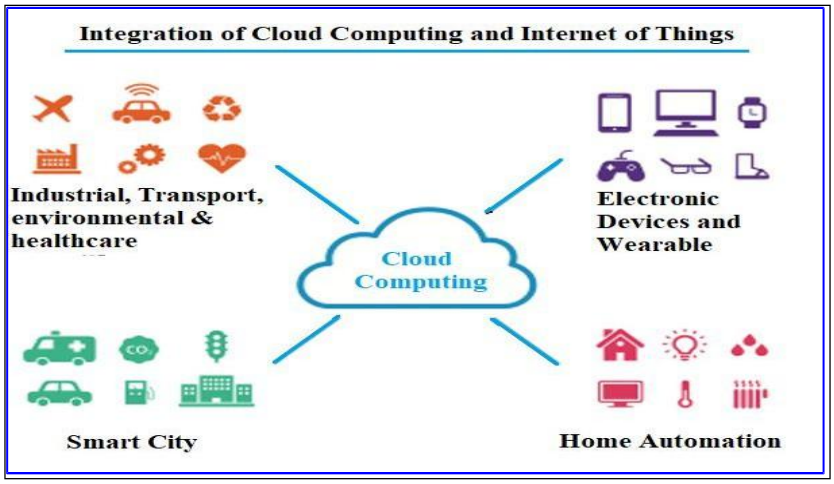


Figure1.4: Integration of cloud computing and IoT

Numerous challenges, such as diverse client needs, various device types, essential communication demands, limited network bandwidth, restricted computing power, operational expenses, and more, impact the effectiveness of the IoT network. An effective resolution to the IoT resource allocation problem (IRAP) will significantly enhance system performance, establishing it as a crucial subject. Efficient resource scheduling and allocation play a pivotal role in effectively managing data centers. They help achieve optimal load balancing, maximize resource utilization, and minimize carbon emissions [10]. Data centre computers often handle applications, while intelligent sensor data is regularly sent to cloud data centers. Due to the growing demands of IoT applications, there is a pressing need to address the issues of rising energy consumption and performance deterioration in computing nodes caused by data transmission and movement. Implementing IoT applications has become a critical concern. With fog computing, the processing of IoT applications is shifted to the edge of the network instead of relying on cloud platforms.

Regarding IoT systems, the data collected by sensors is typically transmitted and stored in the cloud for analysis. However, this can strain resources and lead to latency problems. This cloud-based IoT environment is not equipped to handle the demands of these latency-sensitive conditions [12]. Fog computing is a cutting-edge approach to distributing processing and storage resources to IoT devices with limited resources. In fog computing, multiple compact devices with limited computing power, called edge devices, are deployed close to the IoT sensor layer. After the data is processed and sent to the cloud layer for storage, this helps to eliminate any delays and offers more excellent computing capabilities compared to regular IoT sensors. In a fog computing system, devices or sensors can assign tasks to nearby fog nodes with the necessary computational power or storage rather than sending them to the cloud module located further away [13].

1.4.1 Cloud-Edge to IoT communication

In this scenario, each local scheduler of a cloud edge node can directly communicate with another cloud edge scheduler node to allocate tasks to distributed IoT nodes. Each cloud-edge scheduler node may maintain a list of remote edge node schedulers with which they can interact, or a central directory may contain details about every scheduler. The communication framework facilitates decentralized scheduling between the cloud edge and IoT devices. If a job cannot be assigned to readily available and suitable local resources, the scheduler may contact other remote schedulers to discover resources. Each scheduler is authorized to manage jobs using one or more local job queues.

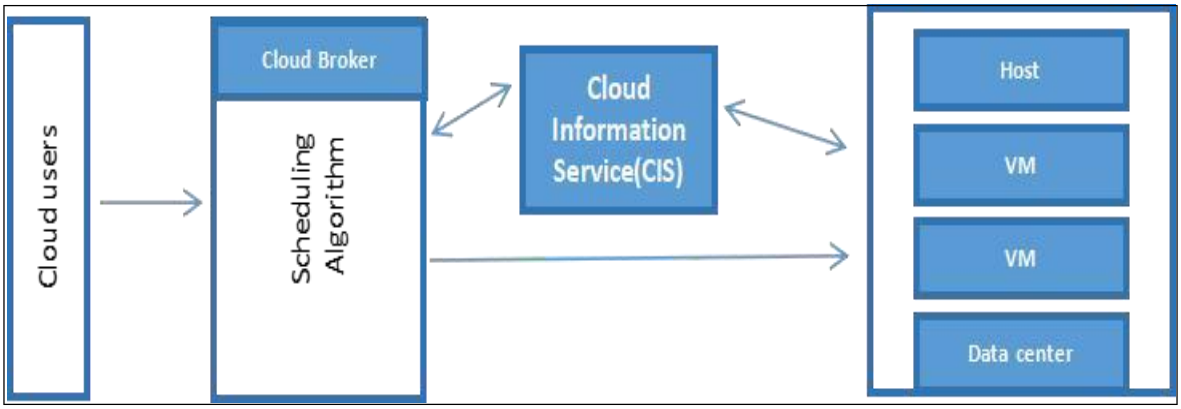


Figure1.5: Direct communications inspired-out scheduling

1.5 Resource Discovery

Tasks that cannot be immediately completed are transferred to an edge task pool. Local planners have the autonomy to select appropriate tasks for scheduling on their hardware and software without necessitating direct communication with the cloud edge. Policies must be defined to ensure that all functions in the pool are executed simultaneously. Figure 1.6 represents depicted above, illustrates how a distributed scheduling system efficiently handles diverse workloads.

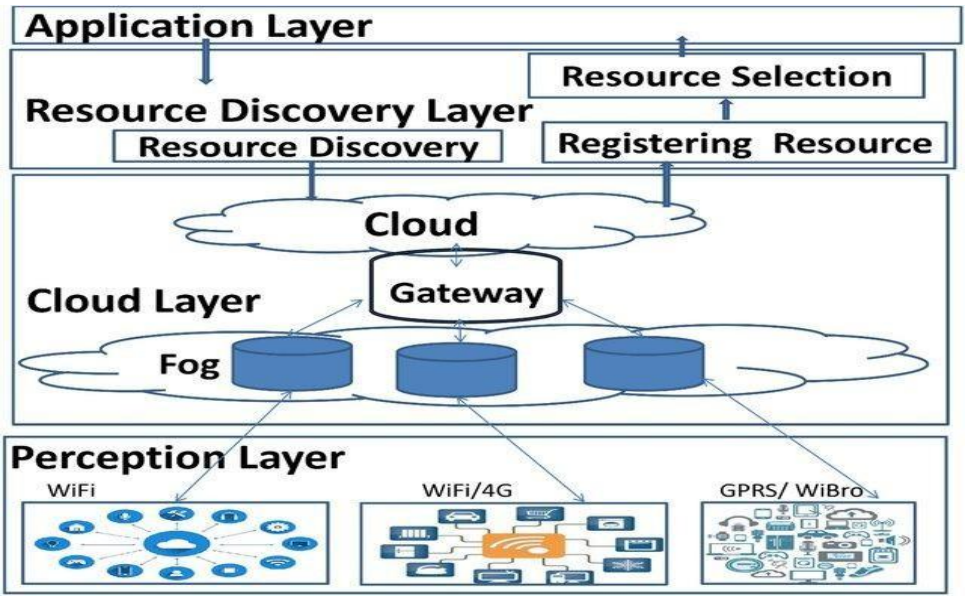


Figure 1.6: Resource discovery

Finding a list of trustworthy resources for posting job requests is the objective of resource discovery. A scheduler requires a method to handle the dynamic nature of the cloud by utilizing dynamic state data about available resources while making decisions. Figure 1.6 represents the scheduled distribution using a task pool. This decision-making process is analogous to a typical compiler on a single-CPU machine. The compiler must be mindful of the quantity of registers and operational units, as well as their availability. It should also consider the amount of RAM accessible, cache configuration, and communication latencies that may arise while utilizing these services. With this knowledge, a compiler can organize instructions more efficiently to minimize resource idle time. Similarly, a scheduler must remain aware of the resources it can utilize, their activity levels, the time required to access them, and the elapsed time since they were last used. It enables the scheduler to optimize resource allocation and minimize latency in task distribution.

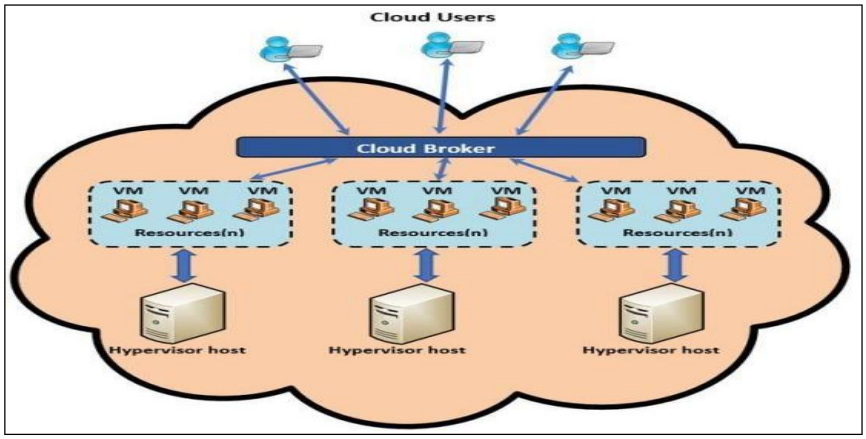


Figure1.7: Schedule distribution using a task pool

The scheduler maximizes work scheduling efficiency by effectively leveraging the available resources. Resource discovery in a cloud environment typically utilizes the pull, push, or push-pull models to find assets available for task reporting and completion.

1.5.1 Resource Selection

The scheduler optimizes job utilization by selecting resources from a predefined list that best matches the constraints and requirements for CPU, RAM availability, and disk storage. This resource selection process involves determining a list comprising all chosen resources capable of fulfilling the essential requirements for a task or job list. The relationship between accessible resources (R available) and selected resources (R selected) is defined as follows: selected resources are a subset of accessible resources.

In the data preparation module, data is extracted from the dateset and processed using appropriate software tools. Before transferring the dateset to the database procedure, the software performs data cleansing to remove redundant and special characters as part of the pre-processing step. Additionally, the software identifies and eliminates duplicate values during the pre-processing stage. In our case study of a movie lens data collection, we have exemplified this pre-processing step by identifying and removing duplicate values and unnecessary information from our hive application table.

1.6 Data per-processing and Data Model Phase

Partitioning separates the data into files after per-processing and allocates the files to a future job's work process using Map Reduce. Before distributing a task to the map reduces, partitioning first checks the values of the data chunks. The portioning module provides a function to Map Reduce to store data in HDFS. At this level, partitioning separates tables into partitions so related data types can be grouped based on a column or partition key. A system table may include one or more partition keys to identify a specific partition. We can simplify the process of conducting queries on data slices using partitions.

The Figure1.8 represents shows how partitioning and clustering are allocated and how these phases are carried out internally. Every reducer on the cluster is in operation. It could be overly limiting for applications that individually process each intermediate cluster key-value pair in a reduced step. Separating clusters can significantly minimize data skew. If cluster splitting is not permitted, the entire cluster must be allocated a single reducer. When cluster splitting was not allowed, the cluster as a whole had to take only one reduction.

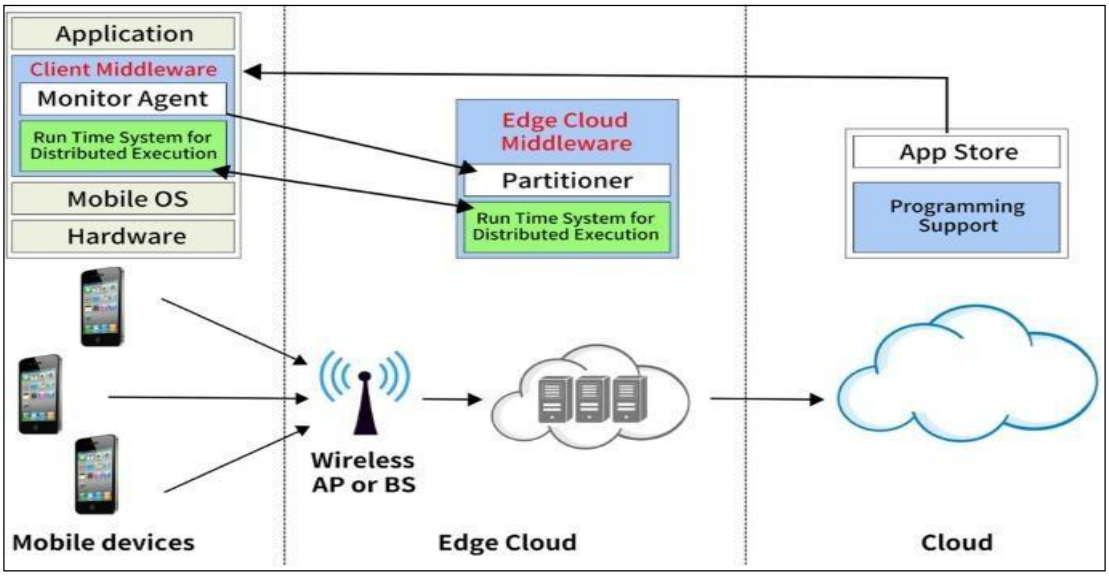


Figure 1.8: Internal working of partition and cluster phase

Following the completion of the partitioning phase operation, the partition is divided into buckets based on the database column hash function to provide additional context for specific information and to be used for more efficient searches. This process is referred to as packet

1.7 Resource Scheduling

There are two aspects to task scheduling. The list of all the tasks' minimum completion time frames for the available resource was initially established in the first phase. The expected resource is given a minimal job in the second step, picked from the minimum task set created in the first stage. The steps are continued until all jobs are assigned to the available resources.

Consider the following: m1 and m2 are two devices; t1, t2, and t3 are three tasks. The length of time that each job took to complete on each machine is shown in Table1. 1. The mechanism of job scheduling .

Table 1.1: Execution time of tasks on machines

	t1	t2	t3
m1	140	20	60
m2	100	100	70

Stage 1: Since machine m1 needs less time than the other machine, task t1 can be assigned to machine m2. Since they both take less time, tasks t2 and t3 can likewise be given to machine m1. Task T2 is eventually allocated to machine M1 due to its quicker completion time than

other jobs. Once task T2 is assigned to Machine M1, its completion time is updated.

Table1.2: Execution time of tasks on machines after stage1

	t1	t3
m1	160	80
m2	100	70

Stage 2: Tasks t1 and t3 are allocated to machines m2 and m2, respectively, as they can be completed more quickly on those computers. Eventually, task t3 is assigned to machine M2 since it finishes quicker than task t1.

Table1.3: Execution time of tasks on machines after stage2

	T1
M1	160
M2	170

In Stage 3, Task T1 is allocated to Machine M1 since it is the sole unfinished task. In the initial stage of the example, jobs with the shortest turnaround times are chosen, leading to the assignment of job T2 to Machine M2. The completion time of Task T2 in Stage 2 is employed to update the readiness time of the machine. Consequently, Task T3 is assigned to Machine M2, and the process continues. Following this, Task T1 is dispatched to Machine M1.

The load-balancing challenge can be conceptualized as akin to graph colouring. In this analogy, an un directed network comprising N nodes represents the system's finite elements. These nodes are assigned P colours, symbolizing processors, to minimize a cost function, where the function's output is associated with the time needed to execute the program under a specific colour configuration.

Researchers endeavour to devise objective functions that mitigate communication overhead and load imbalances, encapsulated as a summation component. By formulating these objective functions, the aim is to optimize the assignment of colors to nodes to minimize both communication costs and imbalances in workload distribution across processors. This

approach facilitates the efficient execution of tasks across distributed systems.

1.8 Cost function calculation

We intend to distribute the elements throughout these device processors to reduce the load disparity generated by one processor having more elements than another and the contact between the components. In order to record a cost function optimized if the maximum completion time of the code is lowered, it must be straightforward yet independent of the specifics of the code. Following that, the components would be distributed in line with the cost function, and that would be all.

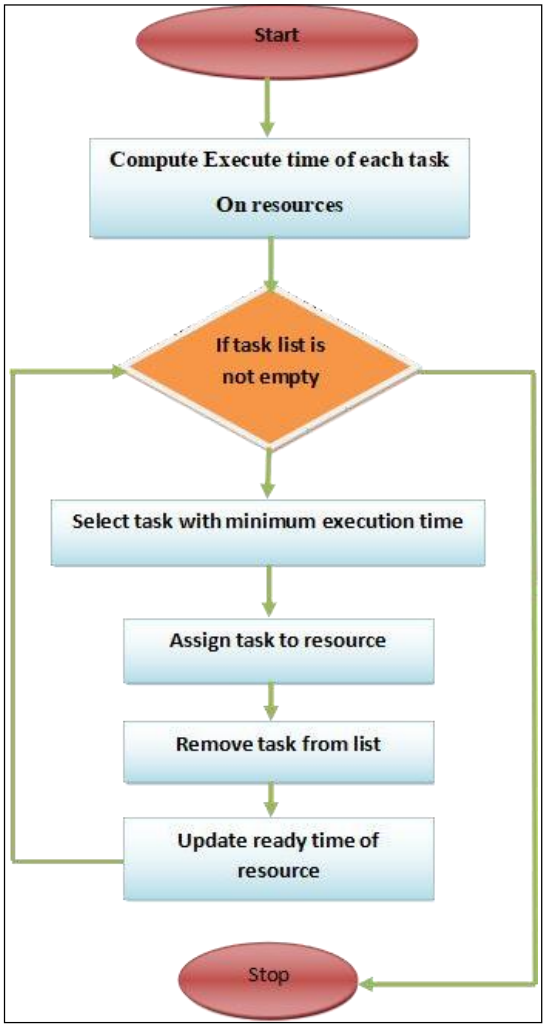


Figure1.9: Flowchart of Scheduling

1.9 Research gaps

1. According to recent research, cloud IoT Edge computing is expanding, enabling clients to migrate various applications to remote cloud data centers.
2. Relocating apps typically results in prolonged communication latency and high network loads because cloud users are often dispersed from distant cloud data centers, significantly impacting user experience.

3. Numerous load-balancing algorithms have focused solely on optimizing performance rather than energy efficiency Many conflicting objectives, such as minimizing response time, maximizing resource utilization, and reducing energy consumption, are often involved in load balancing.
4. Developing multi-objective optimization algorithms that effectively balance several goals may have research gaps. There might be gaps in research for developing methods to dynamically adjust load balancing techniques in response to security events, ensuring the system's defence against intrusions.

1.10 Problem identification

1. **Problem Identification 1:** To support the mobile IoT environment, resource distribution and balancing are essential components of the cloud computing idea.
2. **Problem Identification 2:** To manage the ongoing data production and streaming, IoT devices require a lot of bandwidth and should have a reliable supply.
3. **Problem Identification 3:** Automation in the cloud and intelligent technologies significantly reduce stress about load balancing, resource balancing, traffic routing, and duplicate avoidance. However, sophisticated data analytic technologies are required to collect and evaluate the data.
4. **Problem Identification 4:** Unreliable connections pose problems for significant cloud services; this has previously occurred with Amazon AWS and Microsoft Azure.
5. **Problem Identification 5:** Another challenge is the correlation between the edge, cloud, and IoT computing paradigms. Linking to each paradigm can be challenging, and shifting compute optimization has proven ineffective.
6. **Problem Identification 6:** Data aggregation and integration present additional challenges in a hybrid cloud service model.
7. **Problem Identification 7:** Despite advancements, making real-time decisions remains challenging due to expenses, form factor restrictions, latency, battery life, and unstructured data concerns in IoT devices. IoT Edge applications must address latency, bandwidth, and downtime issues.

1.11 Objectives

1. To design a hybrid approach for load balancing optimization in Cloud-IoT edge applications by employing resource optimization and data skewing techniques.
2. To illustrate data aggregation and data integration techniques to overcome data skew problems.

3. To design an adaptive data skew optimization model to minimize the resource imbalance problem.
4. To assess the effectiveness of the load balancing optimization system for data skew under various conditions, including optimization, data skew, and average consumption rates.

1.12 Research methodology

The research methodology for achieving the objectives above involves a multi-faceted approach. Initially, a comprehensive literature review will be conducted to understand existing methods and techniques related to load balancing optimization, data skew mitigation, resource optimization, and adaptive models in Cloud-IoT edge applications. Subsequently, a hybrid approach will be formulated, integrating resource optimization and data skewing techniques. This will involve the development of algorithms and models for load balancing optimization and adaptive data skew optimization. To illustrate the effectiveness of the proposed approach, data aggregation and integration techniques will be applied to address data skew problems. The research will include simulations and experiments to evaluate the performance of the load-balancing optimization system under various conditions, such as optimization scenarios, data skew scenarios, and different average consumption rates. Additionally, case studies and real-world application scenarios will be analyzed to validate the practical applicability of the

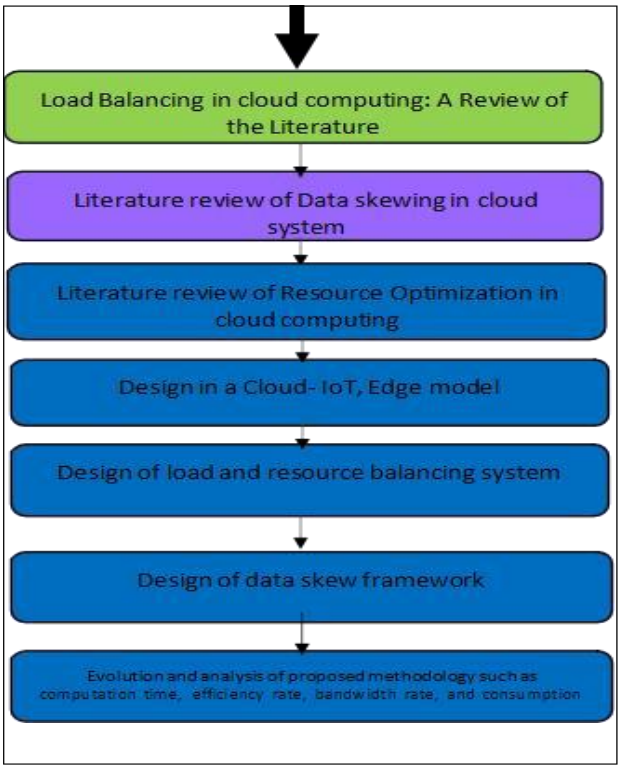


Figure1.10: Research Methodology Flowchart

proposed methodology.

1.13 Motivations

The motivation for this work stems from the growing complexities and challenges faced in cloud computing and storage services, particularly concerning the provisioning of virtual machines (VMs) for processing applications. While VMs have been the traditional approach, their maintenance costs and the need for rigorous capacity planning pose significant challenges. To address these issues, the current trend is shifting towards utilizing containers as the most minor processing units in the cloud. By doing so, the aim is to streamline maintenance and optimize resource utilization, ultimately enhancing efficiency and reducing operational over head. In healthcare, the evaluation of vast amounts of data has become increasingly crucial. Techniques like the Map Reduce Paradigm have shown effectiveness in large-scale data mining, but data skewing remains a persistent challenge. This work introduces a partitioning tuning skew handling method to mitigate data skew issues in Map Reduce. By employing a two-stage algorithm and division optimization approach, this method effectively distributes key pairs in digital partitioning, improving the performance of Map Reduce tasks. The assessment, conducted with real-world and simulated medical data, demonstrates the robustness and effectiveness of this approach, particularly in rule mining for healthcare information associations.

Class imbalance is a prevalent issue in real-world applications of AI, particularly in scenarios with high-dimensional and unbalanced datasets. Conventional element selection techniques often fall short in such situations. This work addresses class imbalance by introducing cost-effective learning approaches considering the costs of mis classifying different classes. By implementing formal procedures tailored to address class imbalance issues, this research aims to enhance the performance of AI applications across various domains.

The management of IoT services presents another significant challenge, particularly concerning resource provisioning and capacity management. IoT networks rely on many technologies, and effective resource management is essential for ensuring high-quality end-user experiences while minimizing operational costs. This work delves into resource provisioning strategies and identifies critical factors for optimizing resource utilization in distributed systems. By focusing on reducing the rate of data skew approximation errors and improving resource allocation efficiency, this research aims to address the complexities inherent in cloud-based IoT ecosystems.

1.14 Research Assumptions

The thesis's research assumptions are rooted in several critical premises across various domains. Firstly, it assumes that transitioning from traditional virtual machines (VMs) to containerization represents a viable and beneficial shift in cloud computing infrastructure. This assumption posits that containerization offers a more efficient and cost-effective approach to resource provisioning and maintenance, improving overall operational efficiency in cloud environments.

Secondly, the thesis assumes that data skewing is a prevalent challenge in large-scale data mining, particularly within the healthcare industry. It presupposes that existing approaches to handling data skew issues in Map Reduce, such as partitioning tuning skew handling methods, can effectively mitigate these challenges and enhance the performance of data mining tasks.

Furthermore, the research assumes that class imbalance is a significant issue in real-world artificial intelligence applications, particularly in scenarios with high-dimensional and unbalanced datasets. It posits that conventional element selection techniques may not adequately address class imbalance issues and that cost-effective learning approaches tailored to varying mis classification costs can improve AI application performance.

Lastly, the thesis assumes that effective resource provisioning strategies and capacity management are essential for optimizing resource utilization in distributed systems, particularly in Internet of Things (IoT) services. It presupposes that identifying key factors and challenges associated with IoT service management can pave the way for more efficient decision-making and resource allocation in cloud-based IoT environments. These assumptions collectively form the basis for the research's exploration and contributions across multiple domains.

1.15 Major Contribution of Thesis

The thesis makes several significant contributions to cloud computing, data mining, artificial intelligence, and the Internet of Things (IoT). Firstly, it proposes a paradigm shift in cloud computing from traditional virtual machines (VMs) to containerization, offering a more efficient and cost-effective approach to resource provisioning and maintenance. By advocating for containers as the most minor processing units in the cloud, the thesis addresses the high maintenance costs associated with VMs and streamlines capacity planning, thereby improving overall operational efficiency.

Secondly, the thesis introduces novel techniques for handling data skewing in large-scale data mining, particularly within the healthcare industry. By developing partitioning tuning skew handling methods, the research effectively mitigates data skew issues in Map Reduce, significantly enhancing the performance of data mining tasks. Additionally, the thesis addresses class imbalance challenges in AI applications by proposing cost-effective learning approaches tailored to varying classification costs. Lastly, in the realm of IoT, the research focuses on resource provisioning strategies and capacity management, aiming to optimize resource utilization in distributed systems. By identifying key factors and challenges associated with IoT service management, the thesis lays the groundwork for more efficient decision-making and resource allocation in cloud-based IoT environments. Overall, these contributions offer valuable insights and solutions to pressing challenges across multiple domains, paving the way for advancements in cloud computing, data mining, AI, and IoT technologies.

1.16 Organization of the Thesis

The thesis titled **"A Hybrid Load Balance Optimization Model for Cloud IoT Edge"** addresses the challenges encountered in optimizing resource allocation and load balancing in cloud-based Internet of Things (IoT) edge environments. By combining various optimization techniques, including cloud computing paradigms and IoT service management strategies, the research aims to enhance the efficiency and effectiveness of resource provisioning in distributed systems.

The organization of the thesis is structured as follows

Chapter 1 serves as an introduction, providing an overview of Cloud IoT Edge environments, associated challenges, and existing approaches to resource allocation and load balancing. Additionally, it outlines the motivation behind the research, its objectives, and the contributions made to the field.

Chapter 2 conducts a comprehensive literature review, examining existing research on load balancing and resource optimization in cloud-based IoT edge environments. This chapter identifies gaps in the current literature and sets the stage for the proposed hybrid load balance optimization model.

Chapter 3 analyzes the specific challenges and intricacies of load balancing in Cloud IoT Edge environments. Drawing on insights from the literature review, this chapter proposes a hybrid optimization model that integrates cloud computing and IoT service management strategies to address these challenges effectively.

Chapter 4 presents the implementation of the hybrid load balance optimization model. Through simulation and experimentation, the effectiveness and efficiency of the proposed model are evaluated, and the results are analyzed in detail.

Chapter 5 introduces innovative approaches to reduce energy consumption and optimize resource utilization in Cloud IoT Edge environments.

Chapter 6 Focuses on data transmission in decentralized IoT environments, and explores the hybrid optimization model proposed earlier.

Chapter 7 concludes the thesis, summarizing the significant contributions made in addressing the challenges of resource allocation and load balancing in Cloud IoT Edge environments. It also discusses the implications of the research findings and suggests potential directions for future research in this field.

1.17 Summary

In this chapter, the Internet of Things (IoT) refers to any artificial or natural thing that can send data over a network and is identifiable by an IP address. The Items Cloud Network Internet allows uninvited interaction with low-cost IoT sensors, implying far wider inter-conceitedness eventually, billions of intelligent devices will communicate with people. Cloud computing has made its way into the core of IT, providing scalability in the delivery of business applications as well as a service platform (SaaS). Both businesses are converting to cloud-based apps through development.

CHAPTER 2

REVIEW OF LITERATURE

2.1 Introduction

This section provides an overview of the existing resource scheduling models, its associated cloud frameworks and the performance parameters improved in it. In general, MCC is intended to deliver cloud services through resource scheduling to the mobile devices for running complex applications with unlimited computing capabilities. The mobile applications may need different resources according to the nature of task it runs. These significant needs of diverse applications differ, making the challenge of improving the execution performance to be non-deterministic. The major research challenges that arise while resource scheduling tasks include how to identify the parameters which have greater impact on the performance of the application, how to obtain optimal solution in a dynamically changing environment and how to choose resources for improving the application execution and yield a profit to the app vendor.

Resource scheduling is a complex task that is performed in a step-by-step process. Major steps taken during the resource scheduling process are partitioning an application, preparation for resource scheduling, and decision to offload or not. Deciding what to be offloaded is typically done during application partitioning. The different granularities of the application can be considered for resource scheduling like object level, method level, class level etc. Applications are comprised of both compute-intensive tasks and GUI-related tasks. The task which is responsible for the GUI cannot be offloaded. So, the compute-intensive part is partitioned either statically or dynamically. Annotation used by application developers is a popular style of static partitioning where the developer writes local or remote with a code for partitioning purposes.

Dynamic partitioning incurs an extra cost as, during the execution of an application, it is taking extra effort to identify the code for local or remote execution. Once the partitions are ready, the next step is establishing a connection with a cloud server, defining the proxy process on both the smart mobile device and a remote cloud server. The device should be robust enough to handle failure if a connection breaks with the cloud server. It must act intelligently by running a computation on the local device itself and provide results to the user. Since program states are transferred, re-executing a portion of the computation will not affect the correctness of the program. The next major step is whether to offload or not, i.e., resource scheduling decision. Various profilers like network, device and program profiler collect information related to

network, application, battery level, and CPU cycle, which help the solver decide for the resource scheduling.

Mobile cloud computing (MCC) is the conceptual structure that mixes three technologies, including mobile net, mobile computing, and cloud computing, to allow cell users to offload records processing and store onto clouds via wireless networks and cell devices. The finest motivation of making use of MCC is gaining benefits of cloud computing technology by way of leveraging cell strategies. The dynamic networking surroundings results in greater complex provider deployments and implementations, evaluating with primary cloud computing. In subsequent phase we can see distinct approaches for handling computation extensive applications that are nevertheless challenging for executing at mobile side. According to the survey cloud computing includes various technology "distributed computing", "parallel computing" and "grid computing". It emerged in autumn 2006 by means of Google engineer Christopher accountable for "Google one hundred and one" mission after that IBM has a joint mission with goggle on "cloud" and afterwards many groups released various plans on "cloud computing" emerged as the synonym for the subsequent era net revolution. Inside the recent years the important it giants for extraordinary cloud carrier vendors are Amazon cloud for storing purpose, google cloud for development, fashion micro cloud for safety and security. The various 10 listed principal techniques of data generation the "cloud computing" continue to be the first-class. In keeping with a latest observe by ABI research, a New York-based firm, greater than 240 million corporations will use cloud services thru cell devices by using 2015.

Cloud computing era affords (EAAS) the entirety as a service; garage, sources, computing sources, development surroundings, testing, security and many others. Many formal definitions were proposed in each academia and industry, the only provided by way of U.S. NIST (National Institute of Standards and Technology). with the explosive boom in cellular applications, platforms and end user needs, disadvantages at the cell device end (e.g., computation and garage capability, energy, shared Wi-Fi medium) considerably hinder further enhancements in application quality of service (QOS)- typical measures of QOS encompass consumer experienced put off, provider reliability/availability and information privateers.

Mobile cloud computing (MCC) goals to triumph over those boundaries by means of integrating cloud computing into the mobile environment to enable cell customers and mobile software vendors to elastically make use of resources in an on-demand fashion. In cellular cloud computing, assets are commonly assigned to a group of computers, which give these sources to clever cell gadgets. Cellular cloud computing concept percentage two most important viewpoints, considered one of them is processing and storage is inside the cell tool different is the whole thing is done at the cloud out of doors the cellular devices. At the same

time as both of them have their advantages and downsides however cloud computing offer platform-unfastened packages.

2.2 Various Mobile Cloud Computing Algorithms

Since the mobile devices have some constraints, there arises a need get sources from outside assets. One of the methods to overcome the trouble is getting sources from a cloud, but the right of entry to such systems isn't always assured or/and is just too high priced. In [36] provides the tips for a framework that mimics a traditional cloud company the use of cell devices inside the region of users. The framework detects nearby nodes which might be in a static mode, which means to be able to continue the same location or follow the same movement sample. If nodes in that state are determined, then the target company for the application is modified, reflecting a virtual issuer created on the-fly among users. In situations like downloading an outline report at a some places, collocation increases the probabilities of humans willing to carry out commonplace tasks. To store the resources like electricity and processing electricity, the collocated cell gadgets can collaboratively act as a nearby cloud and break up the project into smaller sub tasks to be finished on one of kind devices [37]. The effects can then be aggregated and shared. The proposed method allows heading off a connection to infrastructure-based cloud companies whilst preserving the principle blessings of resource scheduling.

In [38], it is referenced that the use of all forms of nearby resources (such as smartphones, tablets, and computers) can collaborate to form a local cloud to achieve a common goal. This approach aims to overcome resource sparseness, power consumption, and low connectivity issues faced in traditional mobile cloud computing. The workload sharing is dynamic and proactive, relying on a cost model to benefit all participants. The architecture includes a resource handler, a job handler, and a value handler. The resource handler discovers collocated resources, the value handler calculates the optimal distribution of jobs to maximize benefits, and the job handler distributes the subtasks, runs the jobs, and collects the results from the sender.

In [39], the SPACCE concept proposes leveraging the computational capacity of PCs to enable distributed collaboration. SPACCE, a complex ad hoc cloud computing environment, can dynamically migrate server functionalities among a set of private, non-dedicated computers based on current needs. This migration optimizes redundant computational resources, enhancing response times for shared software among users. SPACCE facilitates collaborative parameters by offloading server roles to PCs lacking dedicated applications or sufficient computational power on demand. Collaboration among mobile devices within a networked

environment is effective for common tasks, but sometimes requires migration of executable blocks to resource-rich platforms.

In [40], the stack-on-demand asynchronous exception (SOADA) execution mechanism is introduced for scheduling resources in a nearby cloud. This mechanism maintains a stack to store execution states, enabling only the current state at the top of the stack to be migrated. This approach ensures that only necessary data segments are transferred to the destination site, minimizing data transfer overhead even for large images. Asynchronous exception handling is utilized for capturing states in mobile devices, employing a dual-method hierarchy to reduce overhead. However, scheduling resources to the cloud introduces latency as a critical parameter.

In [41], the proposed cloud architecture advocates a two-tier approach to reduce latencies. Instead of relying on a distant "cloud," the approach suggests addressing resource limitations in mobile devices through nearby cloudlets. These cloudlets are decentralized and widely distributed components of internet infrastructure, offering compute cycles and storage resources that can be utilized by nearby mobile devices. Access to cloudlets can be facilitated through WiFi, which not only conserves energy but also provides higher bandwidth compared to other internet services.

In [42], various cloud computing service models are detailed, with a focus on TAAS (Testing as a Service). The architecture includes several service components and highlights key features of the TAAS structure. The discussion extends to mobile cloud computing architecture, applications, and pertinent issues such as cloud environment security, data privacy, and performance. Jerry GAO illuminates the definition, scope, importance, and mobile testing processes on the cloud, comparing them with traditional mobile testing approaches.

Technical benefits discussed include reduced test environment costs through multiple virtual machines (VMs) hosted on a single server, isolated VM environments to prevent crashes from affecting others, and freely movable or copied VM image files.

.In [43], the discussion revolves around the impact of cloud-based software testing, highlighting factors such as domain knowledge requirements, flexibility, cost-effectiveness, security, and economies of scale. Cloud computing is depicted as a burgeoning paradigm that encourages software testers to refine their skills in adapting to its dynamic environment.

In [44], a pioneering approach is introduced to enhance the reliability and security of IoT systems. The hybrid system integrates blockchain and edge computing technologies, incorporating a Proof-of-Contribution mechanism. This innovative approach aims to bolster IoT network security by incentivizing active participation from nodes through rewards for

tasks like transaction processing and resource provision. The study underscores the necessity for innovative solutions to mitigate trust and security challenges in IoT environments, advocating the integration of advanced technologies to ensure a resilient and trustworthy IoT landscape.

In [45], the focus is on cloud environments for software testing, where different clouds offer varied testing methodologies. Service managers play a crucial role in coordinating between core activities across various layers of a standard cloud environment and selecting appropriate clouds based on client product testing requirements. The importance of cloud-based testing for mobile applications is emphasized, along with discussions on various types of Cloud-Based Testing (CBT).

In [46], cloud testing is explored as a shared testing environment that eliminates the need for users to set up and maintain multiple testing platforms. It ensures portability and compatibility of websites and mobile applications across different environments, addressing issues such as functionality, interoperability, and performance. Users record test scripts locally and submit them to the cloud for automated testing using the latest tools available. While cloud testing offers a wide range of static and dynamic testing services, applications hosted on remote clouds may exhibit lower controllability, uncertainties, and observability compared to traditional in-house hosted applications. The transformative impact of automation in the testing domain is also highlighted.

.In [47], the implementation and architecture of the Cloud-Based Automated Software Testing Environment (CASTE) are detailed. This system imposes specific conditions: the device under test must be accessible online, and the testing infrastructure is hosted within the cloud. The CASTE operates continuously, providing an automated testing environment accessible at all times. Various resource scheduling algorithms are employed to optimize testing efficiency and resource utilization.

In [48], the study provides precise definitions of critical components and introduces efficient algorithms and methodologies. Their simulations demonstrate that under heavy system loads, the HACAS (Hybrid Adaptive Cuckoo Search) algorithm effectively selects applications with high energy efficiency and minimal power consumption. Compared to the First-Come-First-Serve (FCFS) algorithm, the HACAS algorithm shows a 30% increase in profitability under simulated conditions. Moreover, during light system loads, the HACAS algorithm’s adaptive service selection scheme successfully balances device loads, achieving a 60% reduction in load variance compared to random service selection schemes.

In [49], the study addresses the MCC (Mobile Cloud Computing) challenge scheduling problem, focusing on minimizing power consumption under stringent time constraints for task graphs in MCC environments. They propose an initial task scheduling strategy that minimizes delays and subsequently reduces power consumption by migrating tasks between local cores and the cloud. Additionally, a linear-time rescheduling algorithm is introduced to efficiently manage task migrations, effectively reducing overall computational complexity.

In [50], the discussed scheduling algorithm operates on QoS-driven attributes within cloud computing. Tasks are prioritized based on their attributes, aiming to minimize completion times by allocating each task to services with the shortest completion times. While demonstrating satisfactory performance in experiments, the algorithm's focus on QoS-driven attributes inherently achieves load balancing without explicitly addressing power consumption considerations.

In [51], attention shifts to addressing challenges related to mobile devices such as battery power, mobility, and load imbalance through a distributed parallel scheduling technique for mobile cloud computing. A master device oversees parameters like battery status and network conditions, distributing tasks among slave devices based on their capabilities. This approach enhances mobile device performance, optimizes network quality, and achieves load balancing, thereby reducing battery consumption without addressing system-wide power consumption.

In [52], Li et al. propose a heuristic load balancing algorithm leveraging ant colony optimization principles in cloud environments. This algorithm considers CPU load, network load, available memory, and other performance metrics to balance resource utilization. The pheromone update mechanism effectively distributes workloads across cloud resources, extending system lifespan and improving overall performance. However, it does not explicitly handle fault tolerance issues within cloud environments.

In [53], a load balancing algorithm for grid computing employing ant colony optimization is introduced. This algorithm calculates pheromone levels based on various resource parameters such as MIPS, communication bandwidth, number of processors, and memory capacity. By assigning tasks to resources with optimal capabilities, the algorithm aims to enhance throughput and overall grid performance. However, it does not incorporate specific Quality of Service (QoS) requirements into its optimization process.

In [54], a service for scheduling resource-intensive mobile applications from resource-limited mobile handheld systems (MHS) to nearby surrogates with ample resources is defined. This service focuses on offloading Java-based applications effectively. Experimental and simulation results validate the service's efficiency in resource scheduling, demonstrating its effectiveness in practical deployment scenarios.

In [55], the Clone Cloud technique is defined as a method for dynamically scheduling execution blocks from mobile devices to the cloud to enhance mobile device performance. At the initiation of a service, clones are created on the cloud side to mirror the smartphone's image. Compared to smartphones, clones are resource-rich and not constrained by battery limitations. The primary advantage of Clone Cloud implementation is noted for its performance enhancement. Chun illustrates applications like virus scanning, image search, and behavior profiling, which are computationally intensive and benefit from this approach. However, software control is limited to either access or exit levels, and local processes cannot be migrated.

In [56], a cloud-related technique is proposed that dynamically performs resource scheduling decisions based on available resources on mobile devices. This method leverages the elasticity of applications, allowing components to be offloaded to or retrieved from the cloud as needed.

In [57], an approach evaluates time and energy considerations for both local and resource scheduling computations. Tasks are offloaded if resource scheduling proves less time and energy consuming than local computing. Following a resource scheduling decision, a clustering algorithm is invoked to optimize decisions for tasks interacting with each other, aiming to reduce communication power consumption. However, this approach overlooks memory requirements, computation complexity, and availability.

In [58], considerations include power consumption, memory usage, and execution time within a multi-criteria application function. However, the introduction of a multi-criteria utility function to simplify optimization raises concerns about parameter weighting and engineering complexity.

In [59], a novel approach involves creating clones within the cloud or utilizing a directory service (Clone2Clone) to establish clones on behalf of mobile devices. Each mobile device has a dedicated VM in the cloud, enabling secure peer-to-peer networks for content sharing, searching, and distributed code execution. This method eliminates reliance on unpredictable and energy-inefficient wireless networks by leveraging stable, high-bandwidth connections within the cloud. Additionally, computationally intensive tasks can be offloaded to the cloud, enhancing performance particularly in environments with low bandwidth and high latency. The approach includes establishing a secure tunnel between the mobile device and its cloud clone for all internet traffic, presenting the cloud clone as a local resource to the mobile device. This setup improves web navigation, facilitates page compression and caching, and includes functionalities like blocking unwanted advertisements and scanning for viruses prior to installation. However, a constant connection to the cloud clone is required, which may not be energy-efficient for all scenarios.

In [60], cloud computing represents a paradigm shift in delivering computing resources via the internet, catering to various sectors like finance, healthcare, and government. Despite its benefits, ensuring robust security remains crucial. Key security issues in cloud computing include data breaches, identity management, compliance, and securing virtualized environments. Providers must implement robust security measures to maintain trust and competitiveness.

In [61], advancements in high-speed internet, Web 2.0 applications, and virtualization have propelled cloud computing to the forefront of technology. Users access web-based applications through browsers, benefiting from dynamically scalable resources delivered as services over networks. Data centers form the backbone of cloud computing, hosting vast computing and storage needs. Latency optimization is critical as data-intensive applications generate substantial data volumes, necessitating efficient data movement within cloud networks.

In [62], cloud computing offers flexible IT resources and services over the internet, demanding novel software engineering approaches to deliver agile, scalable, and secure solutions. Agile Service Networks emerge as a paradigm for cloud software engineering, enabling dynamic service interactions among global applications. These networks enhance technical aspects such as Quality of Service (QoS) and service continuity, while also maximizing business value and utility.

In [63], Jiang et al. discuss their experience deploying scientific workflows on the ExoGENI national test bed. This deployment dynamically allocates computational resources using high-speed circuits from backbone providers. Such dynamically allocated bandwidth-provisioned circuits enhance scientific applications' ability to access large datasets and perform computations across remote sites efficiently.

In [64], the system utilizes HTTP protocol to transmit decompressed images, enabling client-side image processing like sharpening and histogram equalization directly within a web browser without additional software. This architecture simplifies maintenance and reduces computational load on clients. The employed coding algorithm efficiently achieves lossless image compression, enhancing the practical feasibility of the cloud-based system for various applications.

In [65], cloud computing and social media are pivotal technologies advancing healthcare applications. The paper details the design and prototype implementation of a secure social healthcare network operating on the cloud. The system integrates trust-aware role-based access control to address emotional support needs within a real cloud computing environment.

In [66], the system categorizes client performance into clusters across multiple runs, leveraging remote client logging for runtime trace collection. Automated log analysis, combining machine/OS characteristics with kernel-level statistics such as I/O rates and system calls, identifies root causes of issues. An example demonstrates detection of a configuration bug injected into a cluster by altering TCP buffer size.

Zhu et al. [67] discuss traditional limitations of cloud computing services deployed in centralized data centers, advocating for distributed clouds to optimize performance and costs for distributed applications. They introduce Nebula, a decentralized cloud leveraging volunteer edge resources. The paper outlines key properties, design considerations, and illustrates benefits through a distributed Map Reduce application scenario.

In [68], the objective is to develop an efficient media delivery system that optimizes locality. The proposed architecture consists of a novel 3-layer approach: a core cloud for application management, a 2-tier edge cloud supporting geo-dispersed user groups, and dynamic peer-to-peer overlays for locally clustered user groups. This architecture concurrently manages multiple streaming sessions, each treated as an independent entity. Experiments conducted on Planet Lab demonstrate the feasibility and effectiveness of dynamically constructing and maintaining streaming delivery across both user-level P2P overlays and the edge cloud.

In [69], the study explores the trade-offs involved in scheduling computations between an infrastructure cloud and a mobile edge-cloud. Two classes of applications are deployed and analyzed on both cloud types, focusing on application runtime and total battery power consumption. Results indicate scenarios where the edge-cloud outperforms the infrastructure cloud in terms of latency and battery efficiency for specific application classes.

In [70], the paper addresses security risks inherent in cloud computing, particularly focusing on the ambiguities surrounding security mechanisms in on-demand Platform-as-a-Service (PaaS) environments like Windows Azure. The study evaluates security controls and mechanisms implemented across Azure components using an industry-standard framework. It aims to assess how well these security measures meet consumer needs, particularly in identity management and data protection.

In [71], the paper discusses a distributed edge cloud infrastructure designed to provide computation and storage capabilities, addressing current challenges. Edge resources are increasingly attractive due to their availability in volunteer computing, file sharing, and content delivery network (CDN) environments. With the proliferation of powerful multi-core, multi-node desktops and home machines, coupled with high-bandwidth Internet connectivity, edge resources naturally provide locality benefits for data and user processing.

In [72], the study investigates tactical cloudlets and evaluates five different provisioning mechanisms. It aims to demonstrate the feasibility of cyber-foraging in tactical environments by integrating cloud computing technologies closer to the edge. This approach enables tactical cloudlets to enhance situational awareness and decision-making capabilities at the edge, even when disconnected from the enterprise network.

In [73], cloud computing has rapidly gained popularity across various sectors due to its cost-effectiveness and accessibility of data. Recognized as one of the top 10 technologies, cloud computing is an internet-based service model offering computing and storage capabilities to users in sectors such as finance, healthcare, and government. The paper emphasizes the importance of Inter Cloud Data Transfer Security as a critical factor in cloud security, which differentiates providers in a competitive market.

In [74], the paper highlights the increasing importance of edge services in the context of the evolving Internet of Things (IoT). Unlike traditional datacenter-based cloud solutions, edge services require low latency, reduced bandwidth between edge and core networks, and resilience in localized and secure edge environments. The authors propose the Edge Cloud model, integrating service nodes at network edges using the OpenStack framework. They illustrate the model's benefits through applications like an indoor localization system and scalable services.

In [75], cloud computing promises cost-effective and scalable access to computing resources, but concerns such as vendor lock-in, privacy, and data control persist. The paper explores alternative cloud computing models, particularly community clouds at the edge. These community network clouds leverage user-contributed resources to build collaborative cloud infrastructures, either independently or to supplement existing cloud services. The study analyzes their role in community cloud computing alongside mobile, social, and volunteer computing initiatives.

In [76], Cloud Computing enables data upload and application usage over the internet, but increasing device connections strain cloud infrastructure. Fog computing, also known as Edge computing, moves computation from centralized clouds to network devices near endpoints, improving response times. The paper proposes integrating an app store-like platform on network devices to facilitate distributed computation, enhancing cloud network efficiency.

In [77], cloud computing faces challenges like high latency and network congestion, prompting a shift towards Edge computing for improved application response times. The paper evaluates Docker as a container-based technology for Edge Computing, focusing on deployment, resource management, fault tolerance, and caching capabilities. Docker's fast deployment, small footprint, and performance make it a viable platform for Edge Computing applications.

In [78], Cloud Computing enables dynamic scalability without new infrastructure investments, but data security is paramount. The paper discusses cloud computing's security issues and strategies to address data privacy concerns. It defines cloud computing, explores security challenges faced by service providers, and proposes techniques to enhance cloud security.

In [79], industrial systems increasingly rely on software spanning decentralized edge to centralized datacenters and clouds. However, managing applications across these domains lacks homogeneity, leading to inefficiencies and duplicate efforts. The concept of Continuous Computing is introduced to create a seamless environment for multi-domain applications, facilitating workload mobility from cloud to edge. A functional reference model is presented, evaluating cloud technologies for their suitability in supporting multi-domain applications.

In [80], Mobile Edge Computing (MEC) integrates mobile and cloud capabilities within access networks, aiming to unify telecom and IT at the mobile edge. The paper discusses the progress of MEC and introduces WiCloud, a platform enabling edge networking, proximate computing, and data acquisition for innovative services. It also addresses challenges hindering commercial deployment of MEC.

In [81], Edge Scale utilizes serverless cloud computing to provide storage and processing across a hierarchy of data centers distributed geographically between end-user devices and traditional wide-area cloud datacenters. Applications in Edge Scale are structured as lightweight stateless handlers, dynamically instantiated on demand. It offers scalable and persistent storage with optimized access latency and reduced bandwidth consumption through hierarchical data center management.

In [82], Osmotic computing supports efficient execution of IoT services at the network edge and large-scale datacenters. It leverages lightweight microservices on resource-constrained IoT platforms and complex microservices on datacenters. Osmotic computing capitalizes on increased resource capacity at the edge and seamless data transfer protocols between edge and datacenter services, addressing features, challenges, and future directions in "Blue Skies".

In [83], connecting edge and cloud computing enhances high-throughput, mobility support, real-time processing, and data persistency. Cloud computing's elastic provisioning and storage capabilities ensure scalability, persistency, and reliability for diverse data generation needs, adapting infrastructure capacity accordingly.

In [84], Fog computing extends cloud capabilities to end devices to support time-sensitive, location-dependent, and latency-sensitive applications at scale. The paper proposes a fog computing ecosystem, implementing and evaluating it across diverse scenarios: content dissemination in challenged networks, crowd-sourced fog computing, and programmable IoT

analytics. Open source projects are leveraged to optimize the fog computing platform, demonstrating superior performance over baseline algorithms in various usage scenarios.

In [85], the shift towards edge computing addresses challenges in processing Big Data. The paper selects six representative Map Reduce applications applicable to IoT edge computing and performs a time-energy-cost analysis comparing wimpy edge compute nodes and cloud systems with GPUs. Key insights advocate for heterogeneous systems with GPUs on both edge and cloud to achieve up to 70% savings in time and energy for compute-intensive applications. The study establishes an equivalence ratio between brawny cloud instances and multiple wimpy edge nodes, showing cost savings with wimpy systems at the edge compared to traditional cloud computing. Additionally, the analysis highlights performance differences between newer and older GPU systems, attributing variations to core clock frequencies.

In [86], edge cloud computing is proposed for new types of cloud services requiring computing infrastructure at the network edge, driven by IoT use cases. However, existing solutions often use proprietary, closed hardware and software platforms, limiting interoperability and third-party service integration. The paper advocates for building a collaborative edge cloud deployed on home servers to overcome these barriers and enable the development of tailored edge services.

In [87], KubeEdge introduces an infrastructure for edge computing environments, extending cloud capabilities to the edge. This architecture integrates computing resources from centralized data centers and distributed edges, enhancing application performance and user experience. KubeEdge provides a unified environment with seamless communication between edge nodes and cloud servers using Kubernetes-based components. Key features include KubeBus for network protocol infrastructure, a distributed metadata store for offline edge scenarios, and EdgeCore for managing edge nodes and cloud VMs as a logical cluster. This setup enables the adoption of existing cloud services and development models at the edge.

In [88], edge-cloud computing is presented as a hierarchical framework for organizing data pipeline functions in IoT systems. It consists of smart edge devices, a fog computing layer, and a cloud computing layer. The paper discusses the roles of orchestration and intermediation in fog computing for microservices and cloud services, emphasizing automated service provisioning as a fundamental function.

In [89], emerging edge and fog computing models support applications distributed across edge and cloud infrastructures. The complexity of designing mobile edge cloud systems necessitates understanding deployment models, configurations, and performance evaluation. The paper shares experiences studying the performance and data quality impact of a mobile edge cloud system using the MECCA (Mobile Edge Cloud Cornering Assistance) application. Insights

from testing highlight key issues and steps in analyzing edge cloud applications, providing lessons learned for mobile edge computing application testing.

In [90], the conventional approach to resource management in edge computing involves allocating tasks to isolated cloud or edge devices based on factors like energy consumption, bandwidth, and latency. However, this method lacks organization and falls short in meeting the requirements for managing health emergencies in smart home healthcare systems. Emergencies necessitate immediate attention, and different health functions have varying priorities for processing. The paper proposes a voice disorder detection system using a deep learning approach. Patients provide voice samples captured by smart sensors, which undergo initial processing at the edge computing level. Subsequently, the processed data is transmitted to the core cloud for further analysis. The analysis and management tasks are overseen by a service provider through a cloud manager. Once automatic analysis is completed, the results are forwarded to a consultant who prescribes necessary therapy to the patientsIn [91], as edge computing gains momentum, organizations operating across geographically dispersed locations continue to rely on cloud computing for data collection and post-processing. The paper explores the performance trade-offs between cloud-only, edge-only, and hybrid edge-cloud processing approaches. To facilitate this analysis, the authors present an analytic model validated through measurements on representative edge and cloud platforms. This model is designed to be applicable even without direct measurements on specific edge hardware, provided that relevant performance specifications are available. Their measurement-driven analysis uncovers a diverse performance landscape, indicating that no single approach (cloud-only, edge-only, or hybrid) consistently outperforms the others. Instead, performance is significantly influenced by application characteristics and the bandwidth available for edge-cloud data transfers.

In [92], the paper presents a computational resource scheduling strategy with graceful degradation for Model Predictive Control (MPC) execution in cloud environments. The strategy enables seamless control assistance and flexible controller design using edge cloud resources. It demonstrates its application in a high-frequency cyber-physical system, emphasizing improvements in system performance while maintaining cost-efficiency. The approach optimizes computational resource allocation dynamically, ensuring responsive control and mitigating performance degradation under varying conditions. This framework highlights the integration of edge computing capabilities to enhance MPC applications, catering to real-time requirements while managing computational costs effectively.

In [93], the exponential growth of interconnected devices has catalyzed the emergence of Internet of Things (IoT) technology. Cloud computing plays a crucial role in supporting IoT

applications by providing storage and computational capabilities necessary for managing vast data volumes generated by IoT devices. However, cloud computing faces challenges in meeting the real-time requirements of many IoT applications. Edge computing addresses this challenge by facilitating efficient communication, management, storage, and processing of data closer to the point of generation, ensuring rapid response times. This architectural shift optimizes data transfer and enhances scalability and responsiveness for IoT applications. The paper delves into IoT fundamentals, requisite communication technologies, data transfer methodologies, and the pivotal role of cloud services in IoT data management and analysis

In [94], the paper introduces Software-Defined Edge Computing (SDEC) through a cyber-physical mapping perspective, aiming to establish a highly automated and autonomous edge computing environment. SDEC leverages software to facilitate flexible management and intelligent coordination across diverse edge hardware resources and services. The proposed SDEC-based architecture decouples upper-level applications from the underlying physical edge infrastructure, enabling the creation of dynamically reconfigurable smart edge services. This approach promotes resource sharing, reuse, and integration among edge resources and services, thereby enhancing overall edge service capabilities. The paper concludes by highlighting several critical challenges that merit thorough investigation and research within the SDEC framework

In [95], the IoT Smart-Basket project compares the performance of Edge Computing and Cloud Computing systems using Raspberry Pi hardware and a webcam. Employing Python, TFLite, OpenCV, and Google Cloud Vision API for object detection, the system sends results to end-users via the Telegram application. Performance analysis focuses on Time Performance and RSSI Value across different conditions. Results indicate that Edge Computing offers greater stability, with average processing times of 1.74 seconds in Line-of-Sight (LOS) and 1.75 seconds in Non-Line-of-Sight (NLOS) scenarios. In comparison, Cloud Computing exhibits longer processing times, averaging 10.46 seconds in LOS and 5.36 seconds in NLOS conditions. This highlights Edge Computing's efficiency in real-time applications like object detection for IoT devices.

In [96], the study focuses on Mobile Edge Computing (MEC), where computationally intensive tasks are transferred from mobile devices to cloud servers to reduce energy consumption. The research investigates task resource scheduling in multi-user MEC systems with heterogeneous clouds, including edge and remote clouds. Tasks are initially offloaded to edge clouds via wireless channels and can subsequently be forwarded to remote clouds over the Internet. The objective is to minimize total energy consumption across multiple mobile devices while adhering to bounded-delay task requirements. Using dynamic programming, the paper

proposes an algorithm that optimizes energy consumption by jointly allocating bandwidth and computational resources to mobile devices. Additionally, an approximation algorithm with energy discretization is introduced to further reduce complexity, ensuring energy consumption remains close to optimal levels. Simulation results demonstrate significant energy savings—up to 82.7%—compared to executing tasks solely on mobile devices, highlighting the efficiency of the proposed scheduling strategies in MEC environments.

In [97], the paper addresses network resource redundancy and overload in power IoT architectures caused by chimney-type independent service access. It proposes a collaborative cloud-edge architecture where edge computing devices or networks are deployed near perception layer devices in power IoT systems. This setup enables local processing of extensive power and non-power data, merging centralized cloud computing's capabilities with decentralized edge computing. The model aims to satisfy the demands of big data computation and real-time analysis of local data, catering specifically to the intricate network and data management needs of power IoT applications.

In [98], a framework is introduced for mobile edge-cloud computing networks, enabling efficient resource sharing through wholesale and buyback mechanisms between edge and cloud environments. The framework optimizes resource management by formulating problems for edge servers to manage their wholesale and buyback schemes, and for the cloud to determine wholesale prices and allocate local computing resources. Two optimization perspectives are addressed: i) social welfare maximization, where the concavity of social welfare is proven, leading to optimal cloud resource management strategies aimed at maximizing overall societal benefit, and ii) profit maximization, where concavity of wholesale computing resources relative to price is demonstrated. This guides the design of optimal pricing strategies and resource allocations for the cloud to maximize its profits. Numerical evaluations underscore the framework's effectiveness, showing that it can simultaneously maximize total social welfare and individual profits through strategic resource management practices tailored to the specific economic and societal objectives of edge-cloud computing networks.

In [99], the paper presents a technique for mapping concurrent tasks onto a heterogeneous multiprocessor platform with a focus on energy efficiency. Tasks exhibit varying execution times and energy consumption profiles across different processors. The technique explores various possibilities for task ordering and processor assignment to generate a Pareto-optimal set. Each solution in this set outperforms others in at least one criterion, such as minimizing energy consumption or reducing execution time, thereby providing a range of optimal trade-offs suitable for heterogeneous computing environments..

In [100], the paper discusses a per-core Dynamic Voltage and Frequency Scaling (DVFS) technique aimed at energy savings in embedded systems. This approach involves scaling down CPU frequency and voltage during memory-bound intervals of applications, minimizing energy consumption without significantly affecting memory-bound operations. The algorithm optimizes voltage/frequency settings offline to match workload characteristics. All offloaded software executes on remote digital servers, ensuring secure communication through cryptographic measures and surrogate servers. The framework emphasizes low latency, proximity to remote surrogates, and reduced privacy concerns. Key components include template-based virtualization for VM deployment, which is resource-intensive and time-consuming. Application developers must annotate software components as local or remote, adding complexity. Moreover, surrogate-based cyber foraging depends on the availability of services and resources on nearby servers, limiting its scalability and reliability in diverse environments.

In [101], the paper introduces a VM-based cloudlets framework designed to enhance cyber foraging by migrating application images to dedicated remote servers. A cloudlet, defined as a secure and powerful computer or cluster accessible over the internet, supports Mobile Distributed Systems (MDS). Mobile devices act as thin clients, providing a user interface while computational processing occurs at the cloudlet. This framework focuses on dynamically customizing cloudlet infrastructure using hardware-level VM technology, where each VM encapsulates and isolates a temporary guest software environment from the cloudlet's permanent host software environment.

The proposed framework employs various methods for VM migration, leveraging transient customization of cloudlet resources. Key considerations include the necessity for additional hardware support to implement VM technology effectively. The approach involves cloning the mobile device's application processing environment onto remote hosts, addressing challenges such as VM deployment, management on mobile devices, privacy concerns, access control during migration, and security risks associated with VM transmission.

Overall, the framework aims to optimize computational offloading in MDS by leveraging VM technology, thereby enhancing application performance and resource utilization while addressing significant technical and security challenges inherent in VM-based migrations in distributed environments.

In [102], Maniatis et al. propose the Clone Cloud framework, a novel approach for efficiently scheduling and offloading diverse software applications across mobile and remote server environments. Unlike traditional approaches, Clone Cloud utilizes three distinct resource

scheduling algorithms tailored to different application types. Similar to VM-based cloudlet approaches, Clone Cloud involves scheduling the execution state of applications to remote servers, aiming to minimize dynamic transmission overhead through straightforward synchronization techniques.

Clone Cloud primarily focuses on outsourcing computationally intensive tasks to remote hosts, while simpler tasks, like user interfaces, remain on mobile devices. This approach enhances performance for applications such as speech recognition, image processing, and video indexing. Background augmentation extends this capability by fully offloading applications to remote hosts, transmitting results back to mobile devices asynchronously. Applications like antivirus and file indexing benefit from faster search capabilities using this method.

For applications with mixed computational loads and interactive requirements, Clone Cloud employs a mainline augmentation policy. This strategy allows applications to offload intensive tasks while maintaining interactions with other program components, facilitating functions like debugging tools.

However, Clone Cloud faces several critical challenges. The migration of execution environments to remote servers introduces complexities related to security, privacy, access control, and the management of VM deployments on mobile devices. Moreover, the deployment of multiple migration strategies based on application characteristics can lead to increased overhead on mobile devices. The framework's reliance on a single-thread approach can also introduce jitter and variability in the execution times of application components, affecting overall performance.

In summary, Clone Cloud represents a significant advancement in offloaded processing, emphasizing simplicity in synchronization between mobile devices and remote servers. While offering benefits for diverse application types, it necessitates careful consideration of its impact on mobile device performance and the management of security and privacy concerns inherent in VM-based execution environment migrations.

In [103], Portokalidis et al. introduce a Mobile Cloud Computing (MCC) framework aimed at enhancing the functionality of Mobile Ad hoc Networks (MANETs) while addressing security risks through robust risk management and secure routing mechanisms. Their scheme focuses on threat detection within smartphones utilizing MCC, emphasizing energy efficiency by implementing incremental Message Authentication Code (MAC) for ensuring the integrity of mobile user data. Additionally, Jia et al. propose security enhancements through Proxy Re-Encryption (PRE) and Identity-Based Encryption (IBE) schemes, which facilitate secure data services in MCC environments.

In [104], Yang et al. present a secure data processing framework tailored for Mobile Cloud environments, specifically addressing authentication challenges on cloud platforms. Extending existing schemes, they incorporate Public Provable Data Possession (PPDP) with advanced cryptographic techniques such as Diffie-Hellman Key Exchange, Bilinear Mapping, and Merkle Hash Trees (MHT). This integration enhances data integrity and security during cloud-based processing operations. Furthermore, Chen et al. contribute a security framework designed for location-based grouped scheduling services, focusing on preserving identity privacy and implementing robust authentication mechanisms.

Portokalidis et al.'s framework for MCC and MANETs emphasizes dynamic risk management and secure routing protocols to mitigate security threats. By leveraging MCC, they enhance the resilience and performance of MANETs in challenging environments while ensuring data integrity through energy-efficient authentication mechanisms. Similarly, Jia et al.'s adoption of PRE and IBE schemes strengthens data security within MCC contexts, enabling secure and efficient data services.

Yang et al.'s extension of PPDP with advanced cryptographic methods addresses authentication challenges in cloud-based data processing for mobile environments. Their approach enhances data security through sophisticated cryptographic primitives, ensuring data integrity and privacy during cloud interactions. Chen et al.'s security framework further enhances MCC applications by focusing on location-based services, providing robust identity privacy and authentication mechanisms crucial for secure data handling in mobile cloud environments.

In [105], Zhou and Huang introduce a novel privacy-preserving framework that utilizes three distinct schemes—encryption-based, coding-based, and sharing-based—to ensure the confidentiality and integrity of user files stored in the cloud. Their approach focuses on resource scheduling, specifically allocating processing and storage-intensive tasks like encryption and decryption to the cloud based on Cipher text Policy attribute (CPA).

Traditional research efforts often address only isolated aspects of security, such as authentication or confidentiality, using static security algorithms. However, Zhou and Huang's model innovatively integrates multiple security dimensions—confidentiality, integrity, authentication, and privacy—into a unified framework. This comprehensive approach not only enhances security but also dynamically adapts to changing demands for security, quality of service (QoS), and resource utilization among mobile users. By leveraging dynamic resource allocation and robust security mechanisms, their model offers enhanced protection for sensitive data in cloud environments while optimizing resource efficiency and user experience.

which describes the energy utilization of a multisite application execution and used a discrete time Markov chain (DTMC) in modeling fading wireless mobile channels. A Markov decision

process (MDP) framework has been adopted to develop the multisite partitioning problem as a delay constrained, least-cost shortest path problem on a state transition graph. The proposed EMOP (Energy-efficient Multisite resource scheduling Policy) algorithm that has been built on a Value Iteration Algorithm (VIA), founds the efficient solution to the multisite partitioning problem. The numerical simulation results reveal that the proposed algorithm considered the different potentials of sites in distributing the suitable components to achieve a lower energy cost for data transfer to the cloud from the mobile. A multisite resource scheduling execution using the proposed EMOP algorithm attained a greater reduction on the energy utilization of mobiles when compared to a single site resource scheduling execution. The performance of the proposed EMOP algorithm has been evaluated in terms of energy saving by comparing the results to a single site resource scheduling execution. The energy consumption of an application has been observed with nodes ranging from 10-150, in single site execution and multi site execution. From the simulation results, the proposed EMOP algorithm has been found to be an efficient multisite computation resource scheduling approach for mobile devices. It has also been found outperforming the mobile and single site execution with respect to both energy consumption and execution time.

The goal of the study described in [106] is to optimise the trade-off between the energy efficiency needed to transmit data from mobile devices and the data quality needed for applications. This is made possible by a cooperative sensing middleware that is positioned in the space between several mobile applications and devices inside a predetermined physical area. The Info-Aggregation algorithm is at the heart of this strategy. Its goal is to maximise the reuse of sensed data across many applications, which in turn lowers the quantity of data needed for resource scheduling in the sensing environment as well as the number of active mobile devices.

In-depth simulation tests were carried out to evaluate the effectiveness of Info-Aggregation by contrasting its results with a No-Aggregation scenario. The trials varied a variety of parameters, including the quantity of mobile devices (300 to 1000), the number of applications (varying from 50 to 200), and the types of sensors requested (10, 15, 20, and 25). To guarantee robustness, each experimental setup was duplicated thirty times using various random number seeds.The improvement in average battery life for all mobile devices, the decrease in data volume required for resource scheduling, and the average number of active devices were the main evaluation metrics. Data aggregation was not used in the instances where the number of active devices was not significantly reduced by above 50%, according to the results. This decrease is the result of the algorithm's capacity to effectively use the same sensor data for several applications, reducing the number of pointless activations and data transfers.

Additionally, there were noticeable gains with Info-Aggregation in the cumulative residual

energy stored in mobile device batteries at the conclusion of the simulation interval. This improvement became more noticeable as the number of applications rose, underscoring the advantages of using the same data for several requests.

A shortened Levy Walk mobility model was also investigated in the study, which helped to optimise data gathering even when mobile devices moved inside the sensing area. This approach adjusted to dynamic changes in sensor availability and device placements, significantly increasing the effectiveness of data aggregation procedures.

To sum up, the Info-Aggregation algorithm outperformed conventional techniques that lack aggregation strategies in terms of improving energy use. It emphasises how crucial it is to have methodical resource scheduling procedures in place, especially when shifting partially processed apps to the cloud in order to reduce the overall energy consumption of smartphones. Significant progress towards the creation of efficient and sustainable mobile sensing environments can be accomplished by utilising sophisticated aggregation algorithms and collaborative sensing

In [107],The optimisation of energy use through dynamic methods and security measures is the main focus of With regard to energy optimisation, they present a dynamic minimum-cut algorithm that takes security precautions into account. Furthermore, using their call graphs as a basis, they suggest the Free Sequence Protocol (FSP) for dynamic programme execution.

An Amazon EC2 Windows instance with 1.7 GB of RAM, one virtual core with one EC2 computing unit, and 160 GB of instance storage was used in the experimental configuration. Applications in the cloud were handled by a Java server, and customised apps operated on an Android smartphone called HTC Nexus One.

Workload size, network type, computing costs, security (encryption/decryption), resource scheduling limitations, signal intensity, and call graph architectures were among the important variables examined. The battery life and performance showed notable increases, according to the results. Additionally, they emphasised how optimisation of overall energy consumption is affected by compute costs, network kinds, security activities, signal strength, workload sizes, and call graph architectures.

All things considered, the study offers practical methods for improving energy efficiency in mobile computing settings, with a focus on dynamic algorithms and security procedures to successfully reduce energy drain.

In [108], presents proposed schemes of optimizing CPU cycle control and resource scheduling in mobile computing models. The system allocated time division between Mobile Processors Tasks and channel State information, and resource scheduling with the CPU cycle value. The strings of probability of successful computation before the deadline and energy consumed

during a before the completion of the computation were established. Where therefore aiming at maximizing the probability of successful computation before the deadline with minimal energy constraints. During the simulation results energy that was saved during the resource scheduling was saved in great amounts. This may, therefore, Indo to increase the maturity as well as the development of the mobile cloud computing by enhancing the strategies that ensure the management of the computational resources and the computation success probability when the workload changes variate.

In [109], mobile cloud computing is the expansion of cloud computing to mobile platforms, allowing for on-demand processing and storage services. Notwithstanding its infancy, issues include the limited capability of mobile devices for sophisticated applications and sporadic connectivity. Reviews of the literature guide efforts to find novel approaches and create tactics that work in mobile situations. This method combines knowledge from previous studies to improve concepts and expand our comprehension, opening the door for new developments in the field of mobile cloud computing.

In [110], The notion of "cloudlet" is presented, with the objective of maximising mobile device workloads by their offloading to neighbouring cloudlets. Multiple multicore computers are arranged in strategic locations in high-traffic areas such as airports and universities to form each cloudlet. Mobile devices can connect as thin clients thanks to this configuration, which takes advantage of low latency.

Hong et al. (2014) conducted a thorough investigation of the energy efficiency of mobile devices when transferring secure data across various communication networks, including 4G high-speed technologies like Wi-Fi and LTE. In order to determine the conditions for computing transmission times via Wi-Fi and 3G networks, their research used Bench Bee speed data from Galaxy S2 LTE.

Reducing latency and improving service delivery for mobile consumers is the reasoning behind cloudlets. Cloudlets alleviate the performance constraints usually linked to remote cloud services by centralising processing resources in closer proximity to users. For real-time applications like mobile interactive gaming or video streaming, this close proximity facilitates faster response times and data transmission.

Furthermore, the study conducted by Hong and colleagues emphasises the significance of energy economy in mobile communication, especially in situations where secure data transmission is necessary. Their research sheds light on how to best select networks using performance measures that come from actual measurements. In mobile cloud computing environments, this empirical technique guarantees that mobile devices can efficiently identify the most timely and energy-efficient transmission mechanism, whether through local Wi-Fi

networks or 3G connections, improving overall user experience and device longevity.

In [111] , Despite substantial research in wireless networks and invocation management, the focus is on controlling and enabling mobility within mobile cloud computing systems—a field that has received little attention. Monitoring the precise position of mobile devices as they move inside or outside the coverage area of cloud resources is crucial for managing mobility. An effective way to do this is through infrastructure-based techniques, including fusing GPS and Wi-Fi. In order to provide stable and dependable tracking capabilities, these techniques make use of GPS for wider global localization and Wi-Fi signals for local placement. Cloud resources can dynamically alter workload allocation and service provisioning to optimise user experience and performance by continuously updating the device's location data. In order to facilitate smooth transitions and optimise resource usage in mobile cloud environments—thereby augmenting mobility assistance and overall system efficiency—location-aware technology integration is essential.

.In [112] focuses on outlining major developments and obstacles in the field of mobile cloud computing (MCC). In articulating the architectural framework of MCC, the article draws attention to important flaws in this particular context. In particular, it investigates how mobile apps and cloud services might be integrated and highlights common problems resulting from this integration's inherent openness.

The paper identifies specific issues that arise and provides a thorough overview of the operational dynamics of mobile applications in cloud systems. These difficulties frequently centre on security flaws, data privacy issues, and obstacles specific to MCC settings for performance optimisation.

Furthermore, the report conducts a critical assessment of several novel approaches developed to address these issues. These tactics usually include new protocols for security, data encryption techniques, and adaptive resource management strategies designed specifically for MCC. The project is to improve the dependability and usability of integrated mobile apps in cloud-based environments by proactively addressing these problems in order to create a more safe and effective ecosystem for mobile cloud computing.

Along with these security and efficiency improvements, IPv6 has built-in support for IPsec (IP Security), which improves data integrity, authentication, and confidentiality in communications. Further simplifying network setup and maintenance procedures, lowering costs, and enhancing scalability are capabilities like stateless address autoconfiguration Enormous functionality, security, and scalability are all provided by IPv6, which is a complete overhaul designed to overcome IPv4's shortcomings and satisfy the changing needs of contemporary network settings. Ensuring stable and long-lasting network infrastructures that can easily handle present

and future Internet applications depends heavily on its deployment.

In [114], IPv6 is noted for its adaptability across diverse networks like wireless, sensor networks, and rooftop networks. Wireless networks face challenges due to mobility and dynamic conditions, posing obstacles for seamless mobile cloud services. These challenges include scalability, data security, and efficient resource management. Effective solutions are crucial, involving optimized protocols and innovative network designs to ensure reliable performance and address the complexities of mobile cloud environments.

In [115], Mobile cloud services need IPv6's sophisticated features to get beyond these obstacles. An overlay architecture designed for wireless networks is established by IPv6. This architecture facilitates smooth integration and improves scalability in a variety of network scenarios, including dynamic and mobile ones. IPv6 makes it possible for mobile cloud services to have reliable connectivity and simplified communication by offering a greater address space and effective routing capabilities. Its capabilities are crucial for guaranteeing optimal performance and dependable service delivery in contemporary wireless network infrastructures by overcoming the inherent constraints of earlier Internet Protocol versions.

In [116] , IPv6 provides enhanced network security by leveraging IPsec (IP Security) to overcome critical difficulties in Mobile Cloud Computing (MCC). IPv6 networks may transport data with strong encryption, authentication, and integrity verification thanks to IPsec. By reducing the security flaws present in various network contexts, this functionality guarantees safe communication and data security throughout MCC installations. Through the integration of IPsec, IPv6 strengthens secrecy and provides protection against threats, creating a dependable framework that tackles important security issues in mobile cloud infrastructures.

In [117],This particular paper offers a comprehensive analysis of the literature on mobile cloud computing (MCC) and the security issues that surround it. By fusing cloud computing resources with mobile applications, MCC offers flexible, on-demand services without requiring upfront investments. It is an evolution in IT. Performance, compatibility, and resource limitations common in mobile environments are addressed.

Security and privacy concerns pose a substantial obstacle to MCC adoption, notwithstanding its advances. Due to possible hazards, such as data breaches and unauthorised access, organisations are hesitant to completely adopt MCC. In order to create a safe MCC environment that instills confidence in both users and organisations, current research primarily focuses on reducing these risks.

Data availability, confidentiality, and integrity in mobile cloud interactions are among the major security issues noted. Researchers are working hard to develop authentication methods, access control systems, and encryption strategies specifically for MCC settings. The goal of

these initiatives is to guarantee that strict security regulations are followed and to strengthen data protection safeguards.

Therefore, even though MCC has a lot of potential advantages, creating a stable and safe environment is still a crucial objective. To handle new threats and strengthen the resilience of MCC systems, further research and innovation are needed. This will increase confidence and encourage adoption in a variety of organisational environments.

.In [118] , By enabling application service providers to deploy apps dynamically in accordance with customer Quality of Service (QoS) requirements, cloud computing seeks to revolutionise data centre capabilities. Many find it to be an appealing option because of its claims of excellent performance, high availability, and cost effectiveness. Notwithstanding these benefits, worries over security management procedures and weaknesses in protecting sensitive data kept in cloud environments have limited adoption by commercial organisations.

In this work, we explore the important topic of cloud computing data security. To stop unwanted access, especially from insiders, effective client data protection is essential. In order to reduce these dangers, a strong and adaptable security technique with two novel features that set it apart from earlier methods is suggested. To prevent unauthorised data access attempts, consumers are first warned by mobile messaging notifications before any transaction is initiated. Second, the system uses a deceptive technique called Honeypot to offer bogus information to users who have failed to log in, so discouraging further attempts at intrusion.

Mohit Marwaha and Rajeev Bedi highlight cloud computing as an information technology revolution similar to the internet. It runs on an internet-based paradigm that delivers software, shared resources, and information to users' devices on demand based on a pay-per-use model customised to their needs. Usability, dependability, and most significantly, security issues like data security and privacy continue to be obstacles despite its rising popularity.

The IDC market research indicates that the market for cloud computing services was estimated to be worth $16 billion in 2008 and is expected to grow to $42 billion yearly by 2012. For commercial applications, the cost savings provided by cloud computing are anticipated to be three to five times higher, and even more for consumer applications. In a news release from June 2008, Gartner highlighted the revolutionary potential of cloud computing by drawing comparisons to e-business.

Different people define cloud computing differently. For some, it refers to using the internet or network "cloud" to store data and access software, along with related services. Some see it as a refreshed version of the 1960s timesharing concept, tailored for more affordable, current computing platforms sum up, even while cloud computing offers substantial benefits in terms of flexibility, affordability, and scalability, worries about data security continue to play a crucial role in determining adoption patterns. Unlocking the full potential of cloud computing

in the digital age requires addressing these issues with creative security solutions.

In [119] Researchers use statistical task execution durations in their work, as reported in [119], to create energy-saving techniques for embedded systems using dynamic voltage scaling (DVS). Their methodology presents two discrete algorithms that are intended to maximise energy usage while preserving task completion effectiveness.

The first method gives priority to using the least amount of energy while obtaining the highest task completion ratio. Without sacrificing task deadlines or quality of service (QoS), the system optimises energy consumption by dynamically altering voltage levels based on statistical forecasts of task execution timeframes. By using the least amount of energy that is practical, this approach guarantees that activities are accomplished effectively.

The second algorithm, on the other hand, purposefully drops tasks in order to introduce "slack" times into the system. These downtime intervals are purposefully used to conserve more energy. The system achieves large energy savings by eliminating jobs selectively that may endure delays without breaking application-specific QoS requirements. This method enables a careful trade-off between achieving the highest possible level of energy conservation and upholding high task completion rates. significance of adaptive energy management in embedded systems is emphasised by both algorithms. These techniques maximise energy consumption in real-time circumstances by selectively terminating non-essential operations and dynamically altering voltage levels. These adaptable techniques are especially helpful in situations when energy sustainability or a longer battery life are critical, like in mobile devices or Internet of Things (IoT) applications.

The study also emphasises how their methods might be used in real-world scenarios to balance energy conservation with QoS demands. Embedded systems can achieve maximum performance under varied workload situations and prolong the operational lifespan of battery-powered devices by offering flexibility in energy management tactics.

In [120] ,Dynamic Voltage and Frequency Scaling (DVFS) is a pivotal technique that optimizes energy efficiency while balancing performance in computing systems. the optimal CPU clock frequency and the corresponding minimum voltage level based on the ratio of the on chip computation time to the off-chip access time. Their technique lowers the CPU frequency in the memory-bound region of a programme to keep the performance degradation to a low value. Cloud computing which is emerging field because of its performance, high availability, least cost and many others. In cloud computing, the data will be stored in storage provided by service providers. But still many business companies are not willing to adopt cloud computing technology due to lack of proper security control policy and weakness in safeguard which lead to many vulnerability in cloud computing. the paper has been written to

focus on the problem of data security. Service providers must have a viable way to protect their clients' data, especially to prevent the data from disclosure by unauthorized insiders. To ensure the security of users' data in the cloud, propose an effective and flexible scheme with two salient features, opposing to its predecessors. Avoiding unauthorized access to user's data by signaling user by sending message to his/her mobile number at the start of transaction. Displaying fake information in case of unsuccessful login for avoiding further login trials by intrusion (Honeypot). Heterogeneity of SMD architecture and operating platform is challenging for distributed application processing in MCC. Mobile device vendors employ different hardware architecture and operating system platforms for the specific mobile product. Traditional application resource scheduling frameworks focus on the implementation of platform dependent procedures for outsourcing computational intensive loads. For example, Weblets and MAUI are application resource scheduling frameworks which are applicable for .Net framework, whereas visualized execution framework and mirror server are suitable frameworks for android platform. Therefore, homogenous access to cloud services are highly expected wherein SMD are enabled to access widespread computing services of computational clouds irrespective of the concerns about operating hardware architecture and operating system processing requirements and the availability of computing resources on SMD. The centralized distributed application deployment models require arbitration of SMD with centralized server for the selection of appropriate server node. As a result, computing resources (CPU, battery power) of SMD are exploited abundantly for the entire process of application profiling and solving. The deployment of distributed platform, management and operation of remote application processing in the optimal possible fashion is an important perspective of cloud based application processing. It is challenging to provide homogenous solution for heterogeneous devices, operating platforms and network technologies with minimum possible resources utilization on the SMDs.

In [121] Through the mitigation of the high latency inherent in centralised data centres, edge computing provides a substantial breakthrough over traditional cloud computing, offering computing services characterised by high dependability and bandwidth to mobile devices. With its emphasis on bringing processing power closer to users—at the network's edge—Mobile Edge processing (MEC) has become a crucial field of study. But with the increasing sophistication and demand of mobile applications, the conventional resource scheduling techniques used in simple edge computing architectures are no longer adequate to satisfy the changing requirements of MEC designs. Utilising the capabilities of Artificial Intelligence (AI) technology, new study has unveiled a revolutionary way to resource scheduling in MEC in response to these difficulties. This research suggests an intelligent algorithmic framework to improve resource scheduling efficiency by integrating AI with MEC design. The core of this

framework is the application of artificial intelligence (AI)-driven task prediction techniques that take into account the quantity and complexity of compute jobs that come from mobile users.

The computation task characteristics are accurately predicted by the framework through the use of Long Short-Term Memory (LSTM) methods. The long short-term memory (LSTM) recurrent neural network is a powerful tool for predicting task needs in dynamic mobile environments because of its ability to capture sequential patterns and relationships in data. The framework's ability to predict workload demands and edge node performance characteristics enables it to preemptively distribute computational resources among edge computing nodes.

Additionally, a complex resource scheduling technique designed especially for smartphones is included in the suggested model. Through dynamic resource allocation based on real-time job predictions, the system maximises responsiveness and resource utilisation on mobile devices. In order to balance workloads and improve overall system performance, the model also incorporates a task migration mechanism for effective edge cloud scheduling. This mechanism allows jobs to be transferred seamlessly between edge nodes.

The AI-driven resource scheduling framework improves the scalability, agility, and dependability of edge computing systems by integrating these components. It prepares the way for upcoming developments in mobile computing research and deployment in addition to addressing the challenges related to MEC environments. The ultimate goal of this strategy is to position AI as a critical enabler of next-generation MEC solutions by optimising performance, minimising latency, and maximising user happiness in mobile edge computing scenarios.

In [122],The demand on mobile terminals for processing power and energy efficiency has increased in the context of contemporary mobile applications such as augmented reality and autonomous driving. Many mobile devices are now searching edge clouds for processing resources as a result of this increase in demand. Unmanned aerial vehicles (UAVs) present a viable way to relieve pressure on traditional cloud infrastructures when their potential for supporting Mobile Edge Computing (MEC) is acknowledged, particularly in task resource scheduling. As a result of their mobility and close proximity to users, UAVs can be used to improve MEC systems' capabilities by shifting computationally demanding activities from centralised servers to edge nodes. This reduces latency and maximises system responsiveness.

On the other hand, MEC environments provide difficulties due to the dynamics of resource requests, especially with regard to the information asymmetry between users and providers. User-generated blind resource requests have the potential to cause delays and subpar performance, which could compromise the quality of the user experience. This research proposes a unique method of incorporating intelligent agents into the task resource scheduling architecture of UAV-supported MEC (UMEC) systems in order to tackle these problems.

Users, UAVs, and edge clouds can schedule resources more effectively thanks to the intelligent agents in the suggested architecture. With real-time insights and system dynamics visibility at their disposal, these agents possess sophisticated decision-making abilities. Within the framework, agents work together to maximise the distribution of computational resources in order to reduce the amount of energy and time it takes to complete tasks.

This approach's main component is the creation of an organised agent-based model that coordinates resource scheduling plans made specifically for UMEC situations. To obtain data, evaluate resource requirements, and bargain with edge cloud providers for the best resource allocations, agents operate independently. Proactive resource management guarantees effective user demands fulfilment while also improving resource management's agility and reactivity.

By showing notable decreases in task execution delays and energy usage when compared to conventional approaches, the simulation findings published in the research support the effectiveness of the agent-enabled methodology. The framework boosts overall user experience for users accessing MEC services via UAVs by optimising performance and achieving improved resource utilisation through the use of intelligent agents.

Additionally, UMEC systems are more flexible in response to erratic and changing workload circumstances when agents are integrated. While demand fluctuates and optimal performance is ensured, agents continuously monitor system circumstances and make real-time adjustments to resource allocations. Being able to adapt is essential for both keeping high customer satisfaction levels in a variety of operational situations and fulfilling strict service level agreements (SLAs).

Finally, intelligent agent integration into UAV-assisted MEC systems signifies a substantial development in resource scheduling techniques for mobile computing. This framework provides a basis for future advancements in energy efficiency, latency reduction, and scalability optimisation in edge computing environments, while also addressing present resource management concerns by leveraging the cognitive capacities of agents. Adoption of agent-based models is expected to lead to revolutionary developments in mobile and distributed computing paradigms, opening the door to a new era of efficiency and dependability in MEC.

In [123] , Traditional cloud computing models are finding it difficult to handle the exponential growth of commercial data and the requirement for low-latency, high-throughput services as 5G technologies advance quickly and smart terminal devices proliferate. In response, a strategic paradigm known as Mobile Edge computer (MEC) has surfaced to solve these issues by moving computer resources closer to users, which lowers latency and eases network congestion.To address the resource scheduling issue in large-scale heterogeneous MEC environments, a recent research presents Deep Reinforcement Learning (DRL) as an

innovative technique. When it comes to maximising resource allocation among various service nodes and the different situations that mobile tasks meet, DRL is a particularly good fit.

The study initially suggests using candidate network sets and Long Short-Term Memory (LSTM) network layers to improve the Deep Q-Network (DQN) algorithm for handling resource scheduling issues in MEC. The DQN algorithm's prediction power is increased by utilising LSTM networks to capture temporal relationships in task requirements. The extended DQN (IDRQN) method optimises resource allocations based on changing workload demands and node capabilities by incorporating LSTM-based characteristics and candidate network sets. This allows the algorithm to dynamically adapt to the real-world constraints of MEC environments.

The suggested IDRQN approach for task resource scheduling in MEC has been shown to be effective through simulation studies employing the iFogSim and Google Cluster Trace datasets. When compared to other algorithms, the results show higher energy efficiency, better load balancing among service nodes, lower latency, and shorter average job execution durations on performance metrics.

In particular, the IDRQN algorithm demonstrates strong flexibility to changing network circumstances and workload dynamics that are common in MEC settings. IDRQN efficiently maintains service quality, meets service level agreement (SLA) performance targets, and balances resource utilisation by utilising deep reinforcement learning techniques.

Moreover, LSTM integration for candidate network sets and sequence modelling improves the algorithm's decision-making, allowing for proactive and wise resource scheduling choices. By guaranteeing timely task execution and shorter wait times, this strategy not only maximises operational efficiency but also improves the user experience overall.

In [124] ,A new era of computation-intensive applications, such virtual and augmented reality, has been brought about by the rapid proliferation of mobile internet services. Mobile Edge Computing (MEC) has become a key solution to effectively handle these demands. By transferring computational workloads to servers situated at the edge of cellular networks, MEC allows mobile devices to minimise latency and maximise the use of compute resources.

This research presents an approach for task resource scheduling and resource allocation based on Deep Q-Network (DQN), in response to the increasing demand for effective resource allocation in MEC environments. The MEC system's edge servers can handle numerous jobs that are offloaded from individual mobile terminals by the framework. In order to minimise the costs associated with scheduling resources, such as energy, computation, and latency, it develops a single decision-making mechanism that simultaneously optimises task scheduling and bandwidth allocation.

Due to the mixed-integer nonlinear programming involved, the suggested optimisation

problem is intrinsically complicated. The work uses an advanced method to successfully address this problem by utilising developments in DQN techniques. By optimising resource allocations based on real-time job needs and server capacities, the DQN technique iteratively adjusts to the dynamic and uncertain character of MEC environments.

The suggested DQN-based technique can attain near-optimal performance levels, as shown by numerical simulations carried out to assess it. The findings provide noteworthy enhancements in terms of energy economy, decreased computation expenses, and minimised latency in contrast to conventional scheduling techniques. Through the use of deep reinforcement learning concepts, the DQN framework efficiently manages trade-offs between various resource costs, guaranteeing reliable performance in a range of operating scenarios.

Additionally, the scalability and adaptability of the MEC system are improved by the inclusion of joint work scheduling and bandwidth allocation optimisation. This all-encompassing strategy facilitates the smooth integration of various mobile applications that demand high processing and low latency interactions in addition to improving operational efficiency.

In [125] ,A promising paradigm for managing power-sensitive and computation-intensive applications on resource-constrained smart devices is the Edge-of-Things (EoT), which is brought about by the proliferation of sophisticated smart devices and ubiquitous wireless networks. To improve energy efficiency, operation speed, and cost-effectiveness, task resource scheduling optimisation to a local cloud (called cloudlet) is the main goal of this work.

This study presents a task resource scheduling algorithm that utilises Fruit Fly Optimisation (FOTO) to minimise energy consumption while satisfying operational requirements. FOTO optimises job scheduling and resource allocation in the cloudlet environment dynamically, resulting in better resource allocation. Evaluating and contrasting against current approaches such as Genetic Algorithm with Ant Colony Optimisation (GA-ACO) and Concurrent Multitasking Scheduling based on Ant Colony Optimisation (CMS-ACO), performance indicators like energy consumption, execution time, and cost-effectiveness are shown.

Simulation outcomes show that in terms of energy efficiency, speed of execution, and cost savings, the suggested FOTO algorithm performs better than the compared algorithms. The algorithm enhances the overall performance of local cloud-based computing environments for Internet of Things applications by effective task management and resource allocation through the use of Fruit Fly Optimisation.

In [126],The transportation systems made possible by the Internet of Connected Vehicles (IoV) can transfer computing duties from vehicles to Edge Computing Devices (ECDs) for processing. This is done by using real-time traffic data. Notwithstanding the advantages, using wireless communication for job scheduling comes with security dangers, including identity

theft, virtual vehicle hijacking, and privacy violations that allow for tracking. ECO, a task scheduling technique enabled by edge computing that protects privacy, is suggested as a solution to these issues. Vehicle-to-vehicle (V2V) communication-based routing is introduced to safeguard task origin and destination vehicles, and privacy conflicts in IoV task computing are formalised. In order to minimise the energy consumption and execution time of ECDs while maintaining task security, NSGA-II (Non-dominated Sorting Genetic Algorithm II) optimises multi-objective goals. Tests conducted on real data confirm that ECO is efficient and effective at protecting privacy and maximising work performance in Internet of Vehicles situations.

In [127], Task resource scheduling in a multi-layered mobile edge computing system is the main emphasis of the work. It takes into account various users with energy-constrained jobs that can be delegated to a remote cloud with different architecture and network resource restrictions or to edge clouds (cloudlets). research resource scheduling approach that chooses which assignments ought to be offloaded and decide the resource scheduling area on the cloud lets oron the cloud. The goal is to limit the all out vitality devoured by the clients. figure the issue as a Non-Linear Binary Integer Programming. Since the concentrated ideal arrangement is NP-hard, propose a disseminated direct unwinding heuristic dependent on Lagrangian disintegration approach. To fathom the sub problems, additionally propose a ravenous heuristic that figures the best cloud let choice and transmission capacity distribution following assignments' vitality utilization. looked at our proposition against existing methodologies under various framework parameters (CPU assets), variable number of clients and for six applications, each having explicit traffic design, asset requests and time imperatives. Numerical outcomes show that our proposition beats existing methodologies. Notwithstanding the hypothetical methodology, assess our resource scheduling approach utilizing genuine examinations. For the situation, arrangement a genuine tested made out of customer terminal, resource scheduling server found either at the edge or at a remote Cloud. additionally actualized our proposition as an resource scheduling middleware on both the customer and the resource scheduling server. Utilizing the tested, had the option to assess our resource scheduling choice strategy for multi-clients setting with three genuine Android OS applications, with various traffic examples and asset requests.

In [128], Edge computing has emerged as a promising infrastructure for delivering flexible resources in close proximity to mobile users. Due to resource constraints inherent in mobile devices, scheduling computational tasks from smartphones to edge servers is crucial for enhancing the quality of experience for mobile users.. Actually, due to the high speeds of moving vehicles on freeways, there would be various applicant portable edge servers

accessible for them to offload their computational outstanding burden. In any case, the determination of the portable edge server to be used and how a lot of calculation tough to be offloaded to fulfill the relating task time constraints without huge figuring bills are points that have not been examined a lot. Besides, with the expanding sending of portable edge servers, their incorporated administration would cause certain presentation issues. So as to address these difficulties, right off the bat apply shared systems to oversee geo-dispersed portable edge servers. Also, propose another cutoff time mindful and financially savvy resource scheduling approach, which expects to improve the resource scheduling productivity for vehicles and enables extra assignments to comply with their time constraints. The proposed methodology was approved for its practicality and effectiveness by methods for broad analyses, which are exhibited in the paper.

In [129],The current surge in urban smartphone usage has sparked a proliferation of computation-intensive mobile applications like virtual reality and online video, posing challenges to the computing capacity and battery life of these devices.To address the issue, shrewd edge figuring in Wireless Metropolitan Area Networks (WMAN) is proposed to empower portable clients to offload calculation concentrated undertakings to the edge processing hubs which sends registering assets close by the cell phones. In any case, the typical activity of edge computing hubs expends a lot of vitality. Along these lines, it is as yet a test to know about vitality utilization while the registering undertakings are moved to the Edge Computing Nodes (ECNs). In perspective on the test, an Energy-Aware Computation resource scheduling technique, named EACO, is intended to lessen the vitality utilization. Actually, dissect all passageway (AP) outings between the first AP to goal AP and select the most limited way to offload the registering undertakings. Besides embrace Non-overwhelmed Sorting Genetic Algorithm II (NSGA-II) to acknowledge multi-target improvement to abbreviate the resource scheduling time of the figuring errands and lessen the vitality utilization of the ECNs. Plus, abuse Multiple Criteria Decision Marking (MCDM) and Simple Additive Weighting (SAW) to choose the ideal resource scheduling arrangement.

In [130] ,A rapidly developing approach known as the Internet of Mobile Things (IoMT) creates, saves, and analyses enormous amounts of real-time data in order to provide mobile consumers with advanced services.

So as to relieve clashes between the asset restriction of cell phones and clients' requests of diminishing preparing inactivity just as delaying battery life, it prods a mainstream wave of resource scheduling versatile applications for execution to incorporated and decentralized server farms, for example, cloud and edge servers. Because of the unpredictability and contrast of versatile enormous information, discretionarily resource scheduling the portable applications

represents a momentous test to streamlining the execution time and the vitality utilization for cell phones, in spite of the improved presentation of Internet of Things (IoT) in cloud-edge figuring. To address the test, propose a calculation resource scheduling technique, named COM, for IoT-empowered cloud-edge figuring. In particular, a framework model is researched, including the execution time and vitality utilization for cell phones. At that point dynamic calendars of information/control-obliged figuring assignments are affirmed. What's more, NSGA-III (non-overwhelmed arranging hereditary calculation III) is utilized to address the multi-target advancement issue of assignment resource scheduling in cloud-edge figuring. At long last, deliberate examinations and far reaching recreations are led to authenticate the productivity of our proposed strategy.

In [131], The study's main objective is to schedule computationally demanding jobs from mobile devices with limited capabilities to servers situated in edge networks by effectively allocating resources. Reducing the average time it takes for these apps to finish is the goal.

. consider a framework model in which a lot of cell phones is associated with an edge server by means of a common correspondence channel. What's more, study just because the resource scheduling issue for general applications by representing the limit confinements of both the correspondence channel and the edge server. initially detail a static resource scheduling issue as a blended whole number direct programming issue. At that point, stretch out the static issue to a dynamic resource scheduling issue in which an application can be executed whenever. Because of the unpredictability of the issue, it is hard to acquire an answer inside a practical time span. In the manner, propose a productive heuristic methodology dependent on clog mindfulness. exhibit that our proposed heuristic calculation fundamentally beats past resource scheduling calculations as far as the normal fruition time.

In [132], By processing calculations closer to users, edge computing is an emerging concept intended to support low-latency applications like mobile augmented reality. Concurrently, new serverless computing-based technologies are being developed in response to the need for highly scalable stateless task execution in cloud systems.

In the paper, propose a novel design where the two combine to empower low-idleness applications: the is accomplished by resource scheduling fleeting stateless assignments from the client terminals to edge hubs. Moreover, structure a conveyed calculation that handles the exploration challenge of choosing the best agent, in view of continuous estimations and straightforward, yet powerful, expectation calculations. At last, portray another exhibition assessment structure explicitly intended for an exact appraisal of calculations and conventions in edge computing conditions, where the hubs may have heterogeneous systems administration and preparing capacities. The proposed system depends on the utilization of genuine parts on

lightweight virtualization blended in with mimicked calculation and is appropriate to the examination of a few applications and system situations. Utilizing our structure, assess our proposed design and calculations in little and enormous scale edge computing situations, indicating that our answer accomplishes comparable or preferable postpone execution over a brought together arrangement, with far less system use.

In [133]The Quality of Experience (QoE)-based approach for Edge Computing computation resource scheduling is the main topic of this study. In order to maximise user experience and operational effectiveness, this strategy places data processing and decision-making near smart mobile devices and end users—at the edge of the Internet.

Taking into account that keen gadget proprietors esteem both reaction time and battery life, it is sensible to appropriately address the inactivity and vitality trade off. the paper catches a client driven view to handle the resource scheduling booking issue by means of together allotting correspondence and calculation assets with thought of the QoE of clients. figure our structure as a blend whole number non-straight programming (MINLP) issue and fathom it in an effective route by RLT-based branch-and bound strategy. Numerical outcomes show that the proposed resource scheduling plan accomplishes an improved exhibition on dormancy time

In [134] ,The study in reference discusses the notable rise in mobile users (MUs) and Internet of Things (IoT) devices, which has resulted in a sharp rise in complicated apps and media services that require more computations and extensive data exchange. Nevertheless, the processing capacity and energy of these devices are still limited. Furthermore, it is determined that security is a crucial issue for the transfer of sensitive data. the examination displays a multi-user asset portion and calculation resource scheduling model with information security to address the confinements of such gadgets. To begin with, the calculation and radio assets are mutually considered for multi-user situations to ensure the effective usage of shared assets. What's more, an AES cryptography strategy is acquainted as a security layer with shield delicate data from digital assaults. Besides, an incorporated model, which mutually thinks about security, calculation resource scheduling, and asset distribution, is figured to limit time and vitality utilization of the whole framework. At last, a resource scheduling calculation is created with point by point procedures to decide the ideal calculation resource scheduling choice for MUs. Reenactment results show that our model and calculation can altogether improve the exhibition of the whole framework contrasted and neighborhood execution and full resource scheduling plans.

In [135], the paper discusses a Wireless Powered Mobile Edge Computing (WP-MEC) system where a hybrid gateway integrated with MEC servers charges N wireless devices (WDs) using radio-frequency signals. Task resource scheduling for WDs is managed using the Time

Division Multiple Access (TDMA) protocol to optimize resource allocation and scheduling . The objective of the paper is to augment the weighted whole calculation rate by joint streamlining of framework assets the executives and undertaking registering time allotment. To tackle the streamlining issue, a substituting bearing multiplier strategy (ADMM) based dispersed advancement technique is proposed. The proposed technique can break down the improvement issue into N sub-issues, which are illuminated by N WDs. Test results show that the proposed technique beats the benchmarks and significantly expands the weighted aggregate calculation rate while keeping the vitality utilization at a low level under the reason of time

In [136] Computational resource scheduling is a key technique that [136] highlights as being essential to improving the use of smart toys, especially when smart toys are combined with edge computing approaches. In such a model, task scheduling is critical to achieving ultra-low latency.

In any case, existing planning approaches on calculation resource scheduling experience the ill effects of the accompanying shortcomings: 1) Heterogeneity: They plan assignments without joint thought of the heterogeneity of move rate and handling capacity; 2) Optimality: They neglect to locate the ideal arrangement effectively. In the paper, study a make span-limited issue with joint thought of previously mentioned heterogeneity factors for a Toy-Edge-Cloud engineering and plan it as a MILP model. To take care of the issue, propose a novel ideal methodology incorporating the Logic-Based Benders Decomposition (LBBD) guideline with Mixed Integer Linear Programming (MILP) models, where two refinement procedures are proposed to create cuts productively. Broad examinations have been performed in which manufactured applications portrayed by Directed Acyclic Graphs (DAGs) are mapped to various heterogeneous process hubs. The outcomes show that our methodology essentially diminished the general arrangement time contrasting with unadulterated MILP and LBBD and outflanks other cutting edge draws near.

2.3 Summary of Cloud Computing Resource Scheduling Approaches

S.No	References	Methodology	Limitations	Challenges
1	In [121]	Intelligent algorithm resource scheduling	Limited discussion on scalability	Scalability: Adapting the proposed intelligent algorithm for resource scheduling to handle large-scale mobile edge computing.environments.

2	Wang, Li, et.al [122]	Agent-enabled task resource scheduling	Limited exploration of edge server scalability	- Scalability: Investigating the scalability of the agent-enabled framework for edge servers and mobile devices.
3	Lu, Zhou, et.al [123]	Deep reinforcement learning for resource scheduling	Complexity of DRL implementation	- Implementation Complexity: Addressing the technical challenges associated with the deployment and training of deep reinforcement learning models in heterogeneous edge computing environments.
4	Huang, et.al [124]	Deep Q-network for task resource scheduling	Lack of evaluation on large-scale networks	- Network Scale: Evaluating the performance and scalability of the deep Q-network approach in large-scale mobile edge computing networks with diverse edge server capacities and mobile device distributions.
5	Lin, Li, et.al [125]	Fruit fly optimization for task resource scheduling	Limited applicability to heterogeneous networks	- Heterogeneity: Extending the applicability of the fruit fly optimization algorithm to heterogeneous network environments with diverse edge computing capabilities and mobile device requirements.
6	Xu, Zhang, et.al [126]	Privacy-preserving edge computing resource scheduling	Potential overhead of privacy-preserving methods	- Privacy-Performance Trade-off: Balancing the need for data privacy with the performance overhead introduced by privacy-preserving techniques in edge computing resource scheduling.

7	Mazouzi, et.al [127]	Lagrangian relaxation heuristic for two-layered edge computing scheduling	Sensitivity to initial parameter settings	- Robust Optimization: Enhancing the robustness of the Lagrangian relaxation heuristic to initial parameter settings and ensuring consistent optimization performance across various network scenarios.
8	Tang, Li, et.al [128]	Deadline-aware resource scheduling	Limited consideration for dynamic network conditions	- Dynamic Adaptation: Developing adaptive resource scheduling strategies capable of dynamically adjusting to changing network conditions and task requirements in Geo-distributed edge environments.
9	Xu, Zhang, et.al [129]	Energy-aware computation resource scheduling	Lack of evaluation on real-world wireless networks	- Real-world Validation: Validating the energy-aware resource scheduling approach in actual wireless metropolitan area network deployments to verify its effectiveness and practicality.
10	Xu, Zhang, et.al [130]	COM: Calculation resource scheduling	Limited scalability analysis	- Scalability Analysis: Conducting comprehensive scalability analysis to assess the performance and efficiency of the calculation resource scheduling approach in large-scale IoT-enabled cloud-edge environments.
11	Guo, Li, et.al [131]	Efficient resource scheduling for edge computing	Simplified modeling of edge network dynamics	- Edge Dynamics: Addressing the challenges associated with accurately modeling and optimizing complex edge network dynamics to improve the efficiency of resource scheduling in mobile edge computing.

12	Cicconett, et.al [132]	Prediction-based agent selection for edge computing	Dependency on accurate prediction models	Prediction Accuracy: Mitigating the impact of prediction errors on agent selection accuracy through the development of robust prediction models and selection algorithms resilient to uncertainty.
13	Luo, Li, et.al [133]	QoE-based resource scheduling	Limited consideration for network congestion	- Congestion Management: Integrating congestion-aware mechanisms into QoE-based resource scheduling algorithms to ensure optimal task allocation and user experience in congested edge environments.
14	Elgendy, et.al [134]	Multiuser resource allocation with data security	Performance impact of encryption overhead	- Security-Performance Trade-off: Balancing the need for data security with the performance overhead introduced by encryption techniques in multiuser resource allocation for IoT applications.
15	Chunlin, et.al [135]	Joint optimization of remote controlled edge computing	Complexity of TDMA scheduling implementation	- Simplified Scheduling: Streamlining the implementation of time division multiple access (TDMA) scheduling mechanisms to reduce complexity and overhead while ensuring efficient resource utilization in remote-controlled edge computing systems.
16	Li, et.al [136]	Make span-constrained task scheduling	Sensitivity to task variability	Task Variability: Addressing the challenges associated with task variability by developing adaptive scheduling algorithms capable of dynamically allocating resources based on varying task characteristics.

2.4 Summary

explore the domain of hybrid load balancing optimization models tailored for Cloud IoT Edge environments. With the rise of the Internet of Things (IoT), traditional tasks undergo significant transformations, necessitating efficient processing, transmission, and retrieval of data from a multitude of tangible products such as cars, homes, and appliances. The literature discusses the emerging trend of users outsourcing data to service providers equipped with ample storage capacity at reduced costs, proposing a secure and efficient storage protocol leveraging elliptic curve cryptography and a sober sequence for data integrity validation. Additionally, a comprehensive data and software processing protocol executed by cloud customers is outlined to enhance privacy enforcement structures prior to data transfer to the cloud, supplemented by challenge-response protocols for secure credential management. Dynamic data operations are also highlighted to bolster security measures, mitigating risks associated with data leakage and corruption while providing users with enhanced assurance and relief from potential challenges. The subsequent chapter delves into the realm of hybrid load balancing optimization for cloud resources.

Chapter 3

A Cloud IoT Edge Application Hybrid Load Balancing Optimization

There are a variety of load balancing strategies that academics have developed,with the majority of focus being placed on resource management, resource scheduling, resource allocation, and resource scheduling.

3.1. Introduction

Cloud computing has accelerated the use of resources, commercial applications, and web-based information interchange in academics and industry in the real world of today [66, 67]. As part of distributed computing, which preserves information and programmers, millions of PCs are intricately connected and transmitted to a central, distant worker. Since they are required to pay according to how the administration or utility processing facility uses its resources, end consumers gain from it combination of enhancements that put the idea of virtualization into practice, including programming, systems administration, load balancing, asset assignment, transmission capacity consumption, dissemination registration, and web figuring. It gives high accessibility, adaptability to non-critical failure, and a decrease in overhead for various improvements. The sole need for distributed computing is a web-connected terminal; it is not dependent on the software, platform, or web application that users utilize. The ability of a small association or the IT sector to make all the offices operate in the serious business environment without the need for human interaction is growing in popularity, which has led to a rapid increase in data volume and velocity [69].

The Internet of Things is affecting daily chores [66]. In order to enable the processing, dissemination and retrieval of information, the IoT develops a range of real things like actual appliances, automobiles, and homes that have integrated hardware, software, sensors, and internet connectivity [93] [94] [95]. IoT has advanced as a result of greater data generation. The Items Cloud Network for the Internet of Things enables illicit contact with inexpensive IoT sensors, demonstrating amazing accessibility; users will soon have access to billions of pieces of intelligent, smart equipment. Services online are sorely needed. The Internet of Things has benefited a wide range of industries, including those involved in precise manufacturing, healthcare, energy, transportation, and similar fields. [66]. It's good news for computer scientists and system designers that internet service providers are rapidly increasing the number of networking solutions they offer. Studies focus on IoT applications and advancements. IoT systems and hardware are now being created to accommodate future

Even as mobile devices and social media continue to dominate the world, there is a lot of debate about what will happen next. The solutions of the hour are the Internet of Things, the digital revolution, and this straightforward fix. It's critical to Figure3.1 represents out how to make the stored data available since the web constantly pushes out massive volumes of data that strain the data network [83]. The value of computers as a tool has greatly changed for both individuals and businesses with the emergence of the cloud. Using technology to make information accessible globally puts a lot of strain on this due to the interoperability and processing of the concerns [84]. The cloud, which has established itself as a successful platform for data transmission via formal network devices and as a motivated lead link, has made it feasible to utilize this scalability. Due to the superior information quality and security it offers during the transformation process, many businesses opt to utilize a simultaneous direct connection to the cloud transformation data, even if this is not the ideal option [70].

Different manufacturers have all supplied private, public, and hybrid clouds, and each one provides a variety of heterogeneous resources with varying compute (CPU/GPU), memory, and network capacities. The bulk of these clouds, however, are not compatible and do not permit the exchange of computation or data [85]. Due to vendor lock-in, clients are typically forced to employ the data analytic s services of a single cloud provider, which results in missed chances for both parties' businesses and income (customers and providers) [86].

3.2 Methodology

The Internet of Things (IoT) paradigm is all about connecting various gadgets and programmers in a network. One of the key goals of the Internet of Things is to cater to the specific requirements of end-users and convert the vast and diverse data generated by these countless IoT devices into valuable insights [1]. Industry and academia are both intrigued by the potential of the IoT to enhance people's daily lives [2]. With the rapid advancement of IoT and mobile communication technologies, there has been a significant increase in the number of IoT devices and apps. This has resulted in a wide range of user-friendly services being made available to end users [3,4]. Nevertheless, the computational and energy demands of IoT devices have seen a significant increase [5,6]. Due to the limited computing and energy capabilities of IoT devices, these applications cannot be efficiently serviced [7]. By leveraging cloud computing, IoT devices can offload computing tasks to powerful cloud servers, effectively addressing the issue of limited computing resources [8,9].

Numerous challenges arise when dealing with client requirements, device types, communication needs, network bandwidth limitations, computing power constraints, and operational costs, all of which impact the effectiveness of the IoT network. An effective resolution to the IoT resource allocation problem (IRAP) will significantly enhance system

performance, establishing it as a highly significant subject. Efficient management of data centers requires careful resource scheduling and allocation. This helps to balance the load, optimize resource utilization, and minimize carbon emissions [10]. Data centre computers often handle applications, while intelligent sensor data is regularly transmitted to cloud data centers. Due to the growing demands of IoT applications, energy consumption and performance degradation are becoming major concerns. This raises the question of how to effectively implement IoT applications. Processing of IoT applications is shifted to the network's edge instead of cloud platforms with fog computingl.

When it comes to IoT systems, the data collected by sensors is typically sent and stored in the cloud layer for additional analysis. However, this process can be quite resource-intensive, leading to concerns about latency. This cloud-based IoT environment is not equipped to handle these latency-sensitive conditions [12]. Fog computing is a cutting-edge approach to distributing processing and storage resources to IoT devices with limited resources. Within fog computing, a number of compact devices known as edge devices are strategically placed in close proximity to the IoT sensor layer. After the data is processed, it is sent to the cloud layer for storage. This helps to eliminate any delays and offers greater computing capabilities compared to regular IoT sensors. In a fog computing system, devices or sensors have the ability to assign their tasks to nearby fog nodes that have the necessary computational power or storage, rather than sending them to the cloud module that may be located further away [13]. With fog computing, the communication time between IoT nodes and computer servers can be significantly reduced compared to cloud computing. Therefore, it is crucial for this sector to adopt fog computing.

Our proposed method adopts an adaptive data skew load balancing optimization methodology with four separate phases. Data per-processing, the second phase, divides large volumes of data into several slices based on table column size, and data modeling, the third phase, divides data into multiple files or folders [74]. The first phase is cloud-edge to IoT communication, which separates the process into resource discovery and selection. It is used for efficient querying. Due to its high degrees of parallelization and low workload, the method can provide an even closer approximation to an intermediate data distribution that does not suffer from data skew issues. The third stage is cloud scheduling, and the fourth is cost evaluation.

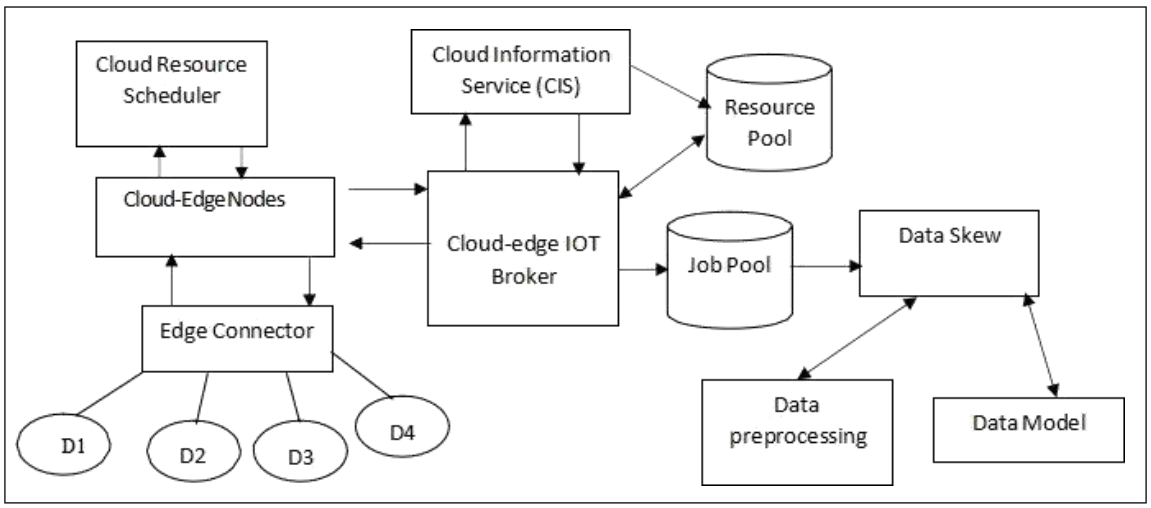


Figure 3.1 : Cloud Edge IOT Architecture diagram

3.2.1 Cloud-Edge to IoT Connectivity

Any other regional scheduler in such a cloud-edge situation can directly communicate with other cloud-edge scheduler nodes for the purpose of allocating jobs to IoT-distributed nodes. Every cloud-edge scheduler node looks to have a collection of wireless edge node scheduling methods with which they may communicate, or there can be a single location where much of the data that matters to each planner is kept [75, 76, 77]. In Figure 3.1 [78]represents , the structure of continuing communication between the cloud-edge and the IoT centralized planning framework is depicted.

Each job that couldn't be completed immediately was added to the edge work database. Instead of directly interacting with the cloud edge, the main grid may be able to choose appropriate jobs to prepare theas sets[79].The pool's research must always be completed;thus,steps must be done to assure this.The hierarchical scheduling strategy isseenin the diagram below.

3.2.2 Resource Discovery

Creating a list of authorized tools for job submissions is the main goal of asset exploration. To effectively manage the intricacies of the cloud, a scheduler requires a system that integrates advanced state information about the required equipment with the decision-making process [80]. The decision-making mechanism of this single-processor computer bears resemblance to that of a typical compiler. The programme can provide the user with information on the available number of displays and usable units, including their availability status and whether they were previously occupied. When utilizing these resources, it is crucial to consider the memory requirements, other data setups, and social connectivity delays that will be implemented. The programmer successfully accomplished this task by utilizing their expertise

to arrange the instructions in the most efficient manner, resulting in a reduced asset accuracy period. Just like a systems analyst, the arranger will have a deep understanding of the tools they can use, the complexity of their task, the time it will take for them to interact and communicate with each other. By utilizing this data, the scheduling algorithm is able to optimize job planning, resulting in a more efficient and advantageous utilization of available resources. When it comes to resource discovery, a public cloud often utilises different configurations such as pulling, moving, or push-pull architecture. Through the asset inventory step, the existing assets (R available) for job application and execution in a cloud environment were identified.

3.2.3 Choice of assets

The selection of potential point assets, which best met the user's needs and took into account Windows' support for software, cloud services, IoT integration, and edge computing—including considerations like CPU consumption, usable RAM, and disk space—was the second step of the planning phase. The initial stage in resource selection is to locate research data. Chosen in which every resource must fulfill a job's or task list's requirements [81]. Connecting chosen and accessible resources.Rselectedassets:30 Rselected⊆Ravailable

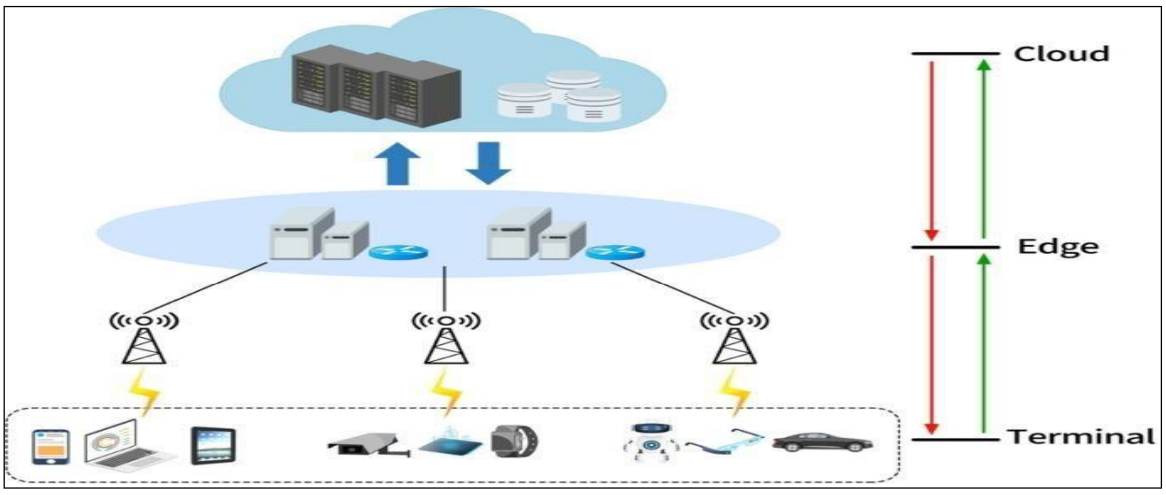


Figure 3.2: Internal working of partition and cluster phase

On the cluster, every reducer operates. For applications that process each intermediate cluster key-value pair separately in a reduced stage, it could be unnecessarily restrictive. Data skew may be drastically reduced by separating clusters [89]. A single reducer must be assigned to the whole cluster if cluster splitting is not allowed. When cluster splitting was not permitted, the entire cluster had to receive a single reduction. After the partitioning stage has been finished, the partition is separated into buckets in this step called bucketization, depending on the column hash function in the database to add more context to a detail and be utilized for more effective searches.

3.2.4 Resource Scheduling

On the cluster, every reducer operates. For applications that process each intermediate cluster key-value pair separately in a reduced stage, it could be unnecessarily restrictive. Data skew may be drastically reduced by separating clusters. A single reducer must be assigned to the whole cluster if cluster splitting is not allowed. When cluster splitting was not permitted, the entire cluster had to receive a single reduction [90]. After the partitioning stage has been finished, the partition is separated into buckets in this step called bucketization, depending on the column hash function in the database to add more context to a detail and be utilized for more effective searches.

Resource Management When activities are planned, timing consists of two parts. The first stage begins with obtaining the minimum execution times for all of the processes for the common supplier. In the second stage, the bare minimum tasks from the limited set of tasks created in the first phase are selected, and the bare minimum workflow scheduling is given to the anticipated resource [91]. The processes are continued until all jobs are linked to the tools.

3.3 Proposed Nearest Master Server Load Balancing Algorithm with Co processor Integration

A suggested method for load balancing in a cloud computing environment is presented in the following section. This algorithm aims to achieve resource optimization by efficiently distributing computational tasks and network traffic across devices. Load balancing in IoT enhances system performance, reliability, and scalability by preventing bottlenecks and maximizing resource utilization. Through dynamic load distribution, it enables IoT networks to handle varying workloads effectively, manage bandwidth utilization, complete tasks within deadlines and respond to clients promptly [92].

Shay Vargaftik et al. introduce the Locally Shortest Queuing (LSQ) class of load balancing techniques. In these methods, each controller utilizes Join Shortest Queue (JSQ) on its local view and maintains its own, albeit occasionally outdated, estimate of computer waiting time. A minimal amount of connection overhead is associated with periodically updating this local view. The network exhibits high resilience as long as these local estimates of service waiting time remain accurate, despite being based on assumptions. Simulations demonstrate that, given identical communication allowances, simple and robust LSQ policies perform exceptionally well and significantly outperform existing low-communication strategies.

The Balancer Genetic Algorithm (BGA), a novel load balancing scheduler, is provided by Rohail Gulbaz et al. [59] in order to improve make span and load balancing. Inadequate load balancing may lead to an overhead in resource utilization when certain resources are idle. BGA

load balancing takes into account the actual weight, expressed in millions of instructions sent to virtual machines. Additionally, the importance of using multi-objective optimization to improve work scheduling and load balancing is stressed. Different batch sizes and skewed, normal, and uniform work distributions are used in the trials. BGA has significantly outperformed numerous state-of-the-art methods for make span, throughput, and load balancing.Centralization of cloud-based IoT services (apps and data) has been a trend since the introduction of the Internet of Things concept, which allows for the management and orchestration of a large number of Web devices. In order to effectively monitor a group of sensors using IoT management services, an IoT gateway must minimize the frequency of network link overloads or failures connected to it. To address these problems, we employ a cloud of things load balancing technique. In this section, we report our findings, deployed and implemented accomplishments, and results evaluation.

The fundamental goal of a load-balanced cluster system based on simulated annealing is to increase network lifetime while preserving sufficient sensing coverage in situations when sensor node data transmission is irregular or regular. Through simulation experiments, we demonstrate that the proposed algorithm can improve the range of communications services and that network coverage may be increased by keeping more network devices operational for longer periods of time at low computation cost [61]. This is in comparison to the most popular state-of-the-art clustering approaches. Resource-constrained networks find various applications in everyday life. It is a challenging task to find a consistent load balancing technique that would extend the lifespan of these networks. The technique considers variables including hop count, remaining energy, and distance in order to optimize energy consumption among network users and extend network lifetime. Our proposed system performs better in simulations than the current systems in terms of characteristics like throughput, node lifespan, packet loss ratio (PLR), communication costs, latency, and computation expenses. Furthermore, our proposed approach prolongs the lifetime of WSNs and safeguards individual nodes from current methods in the operational environment.

Wireless sensor networks are a type of self-system with limited power and transmission sources (WSN). Based on a greedy anticipated energy cost measure, a potent heuristic update technique is used to optimize the route establishment. Lastly, to reduce the power consumption brought on by the control overhead, EBAR employs a power opportunistic broadcasting technique. The results of this exhaustive analysis show that EBAR provides a significant improvement over the state-of-the-art techniques, such as EEABR, Sensor Ant, and IACO.

Centralization of cloud-based IoT services (apps and data) has been a trend since the introduction of the Internet of Things concept, which allows for the management and

orchestration of a large number of Web devices. In order to effectively monitor a group of sensors using IoT management services, an IoT gateway must minimize the frequency of network link overloads or failures connected to it. To address these problems, we employ a cloud of things load balancing technique. In this section, we report our findings, deployed and implemented accomplishments, and results evaluation.

Since RPL was not designed with IoT devices in mind, several problems persist even though it greatly satisfies IoT network requirements. First, a description of the CAOF, which takes the node's context into consideration while calculating the score. Additionally, CAOF avoids the thundering herd effect by gradually regressing from a highly positioned number and reaching the actual rank amount. Second, we present a novel routing metric called context-aware routing metric (CARF), which minimizes the legacy of previous parents as it moves further down the path by recursively analyzing parent chain queues and electricity usage as they approach the root. Evaluation findings show improved network lifespan and decreased packet loss when compared to the RPL standard specification. The way the system is set up, the node data is fed into our proposed hybrid cloud, which makes the selection using the suggested AI model. After data collection, the simulation analysis does the load balancing, optimization rate, skew rate, and consumption rate analyses. The node- mcu network cloud is established as a hybrid cloud , which is the hardware analysis and data modelling is done to do optimization rate, skew rate, and consumption rate analysis..

Algorithm: Nearest Master Server Load Balancing Algorithm with Co processor Integration

Step 1: For each request i, where i ranges from 1 to n.

Step 2: Each request i is sent to the Master jk, which is nearest to its region.

Step 3: If Maxjk > i (where Maxjk is the threshold capacity of the master server): -

If the task is equal to Tserver or Tconn through arguments: -

If Tconn is equal to "yes", the request is moved to the guide co-processor and Net_T. –

Calculate Tevent_ex = Ttotal - Texec(Tconn + Tserver). –

While Tconn = carry out on scale also co-processor Tconn - Tdeadline: -

For each i from 1 to n, remaining event elements: -

If co-processor index is even (f) and TRemeven[i] is equal to 0, it indicates successful updating of data, and Veven increases. - Otherwise, if index is even (f) and TRemeven[i] > 0, Tnreal - Veven will transition to ideal c. –

End of loop. –

End of if condition.

End of step 3.

3.4 Implementation

The parameters outlined for the results mentioned above encompass various aspects of cloud computing and infrastructure management. These include the utilization of virtualization software (VMs) alongside cloud computing and storage services, as well as the integration of automation technologies for deployment and management processes. Capacity management and manual intervention remain crucial for optimizing resource allocation and ensuring operational efficiency. A notable trend is the adoption of containers as a preferred approach over traditional virtual machines, offering greater agility and scalability. These parameters collectively define the landscape of cloud infrastructure management, guiding decision-making and shaping industry practices towards more efficient and adaptable solutions.The basic parameters for data validation has been cross-validated by considering map reduction and its optimization, by taking consideration of oversampling classification methods. The evaluation matrix was also considered for model validation. Before we used any model, we validated using statistical tools and there after we went for ML analysis.

The implementation of the algorithm begins by processing requests and distributing them to the nearest master server. If a server's capacity surpasses a threshold, tasks are reassigned based on their release and deadline times. For tasks requiring real-time processing, they are moved to a co-processor to optimize resource usage. The algorithm then executes tasks and updates their execution times accordingly. Successful updates are counted for both even and odd-indexed co-processors. Overall, this approach efficiently manages task distribution and execution, considering server capacities and real-time requirements, thus enhancing the overall performance and resource utilization in the system.

3.5 Results

The system utilizes node data as input for the suggested hybrid cloud, which employs an AI model for selection decisions. It is structured into simulation and hardware environments, delineated in below Figure 3.3 shows . Simulation analysis entails load balancing, followed by data aggregation and subsequent optimization rate, skew rate, and consumption rate analyses. In the hardware analysis phase, a node-mcu network cloud is established as the hybrid cloud, facilitating data modeling for further optimization rate, skew rate, and consumption rate analyses. This comprehensive approach allows for a thorough evaluation of the system's performance and efficiency across both simulation and real-world hardware environments.

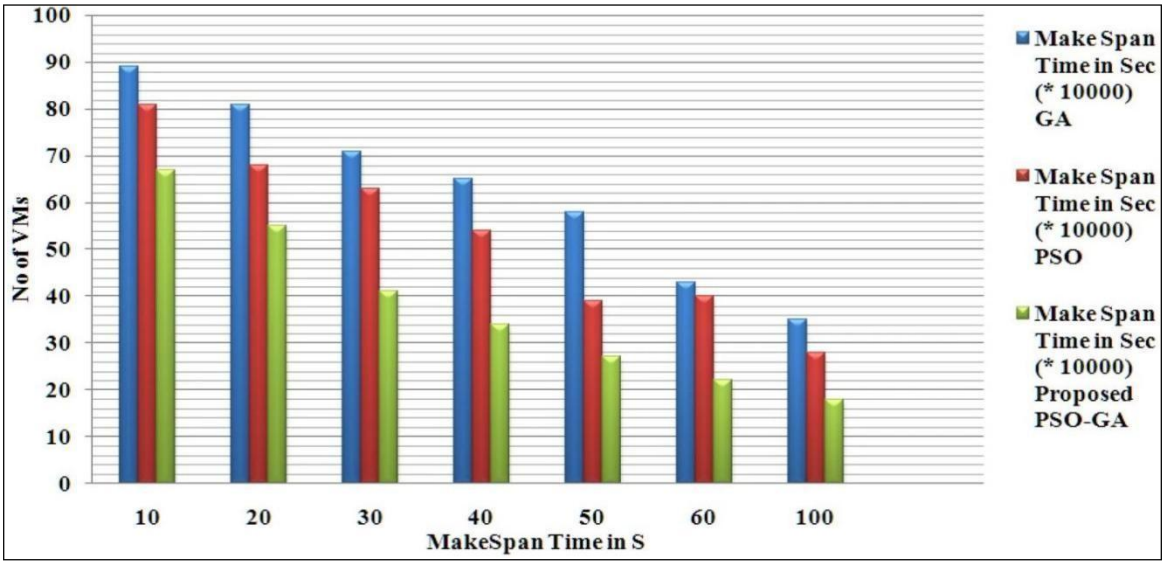


Figure 3.3: Make span Time analysis

Figure 3.4: Showing the overall hybrid cloud set up for consumption rate analysis.

In Above Figure 3.4 represents illustrates the comprehensive setup of the hybrid cloud specifically designed for conducting consumption rate analysis. This setup incorporates both cloud-based and edge computing elements, ensuring a balanced and efficient distribution of computational tasks and network traffic. The hybrid cloud architecture seamlessly integrates various resources, including servers, edge devices, and networking infrastructure. By leveraging the capabilities of both cloud and edge computing, this setup optimizes resource utilization and enhances overall system performance. Additionally, it enables the evaluation of consumption rates across different components of the hybrid cloud environment, providing valuable insights into resource consumption patterns and efficiency metrics.

3.6 Summary

In the landscape of bulk and stream analysis programs, the utilization of virtualization software (VMs) in conjunction with cloud computing and storage services has become commonplace. While automation technologies have significantly streamlined deployment processes, the intricacies of capacity management and the need for manual intervention persist to ensure optimal performance and resource allocation. Recognizing these challenges, there is a discernible shift towards containerization as the preferred approach for managing cloud workloads. Containers offer greater agility, scalability, and efficiency compared to traditional VMs, enabling organizations to adapt more seamlessly to fluctuating workloads and resource demands. This trend underscores the industry's ongoing pursuit of innovative solutions to enhance cloud infrastructure management and maximize operational efficiency.

CHAPTER 4

Map Reducing Task - An Optimal Partitioning Balancing Method for Solving Data Skew Problems

Using a distributed software approach, the programming paradigm Hadoop allows for more effective parallelism of huge datasets. To produce the correct outcome, the information is split initially.

4.1 Introduction

4.1.1 Map Reduce Computation

They offered map-reduce techniques that rely on Hadoop systems. To guarantee more precise outcomes, they created the recommended model in a simulated setting. In the MATLAB environment, they skillfully put the suggested system into practice for this study. Hadoop instantly restarts each process in a fresh instance on a different server that has a backup copy of the data. The developer is blind to the complexity of the debugging procedure [96].

The reduction and mapping methods that functional programming languages like Lisp use to enable Map Reduce are described in depth in the aforementioned Figure 4.1 shows The map and reduce functions in this paradigm solve problems related to virtual computation [97]. All issues can be automatically parallelized thanks to the formulation. Using the Map Reduce paradigm, users create functional-style code for map and reduce components. The relationship between these components is made much obvious by arranging these pieces into a flowchart. In , the Hadoop running time technique effectively handles resources to accommodate several modules and surmount obstacles such fault tolerance, network connectivity, and parallelization [98]. The key set is added to a group of key/value pairs that are produced by the mapper or Both the type of input keyword and values, as well as the display format for keys and values, are probably going to change.

(key1; value1) →list (key2; value (2) ... (1)

The reduced method produces a list of new values from the entry of a keyword and the list of values it is connected with:

(key2; list(value2)) → list(value3) … reduce (2)

A Map Reduce application consists of two distinct phases that operate simultaneously. Early on, each map-related task can be finished independently. Each reduced operation in the second phase could be determined by any number of mapping processes' outcomes. Like map procedures, each reducing approach can be executed on its own [99].

4.1.2 Uses of Map Reduce

Using Google:

Creating an index for Google.com

Clustering of articles for News Websites

-Statistical

translation software at Yahoo!:

Creating an indexing for Yahoo! Search

- For malware detection Using Yahoo Mail on Facebook

-Ad optimization

-Spam detection

4.2 Methodology

4.2.1 Map Reduce Workflow for mapping the resources and task

A mapping script and a reduction script must be written by designers when building a Map Reduce process. The current jobs will be managed by the Elastic Map Reduce (EMR) architecture on Amazon. When designers start a map/reduce method, this architecture divides the inputs into 11 parts and transmits each part to a separate machine. Each processor's supplied data is used to run the map script [105].

The map script you write will convert the inputs into keys, values, and pairs based on your specifications. Imagine a situation where someone has to determine how frequently a particular phrase appears in a given text. In this instance, the phrase can be stored as the key and the accompanying count as the value by using key-value pairs.Map Reduce produces a word; there is a <> pair for each word in the input file. It's important to note that, in contrast to the map script, the reduction script manages all aggregates and real tallying. The map script will convert the data into key-value pairs so that the reducer may aggregate them [106].

The generated key and value pairs are rearranged in the illustration below, resulting in the grouping of pairs with the same key. These grouped pairs are then delivered to a virtual device, where the reducing script is executed on them. The reduce script provided by the user is utilized by the reduction script (also provided by the user) to decrease a collection of keys and corresponding values. In our word count example, we are analyzing the occurrences of each term to establish its frequency. Our decrease scripts should simply sum up the values of each key-value combination with identical keys, just like a systems analyst would do. The example in Figure 4.2 shows represents effectively demonstrates the situation [107].

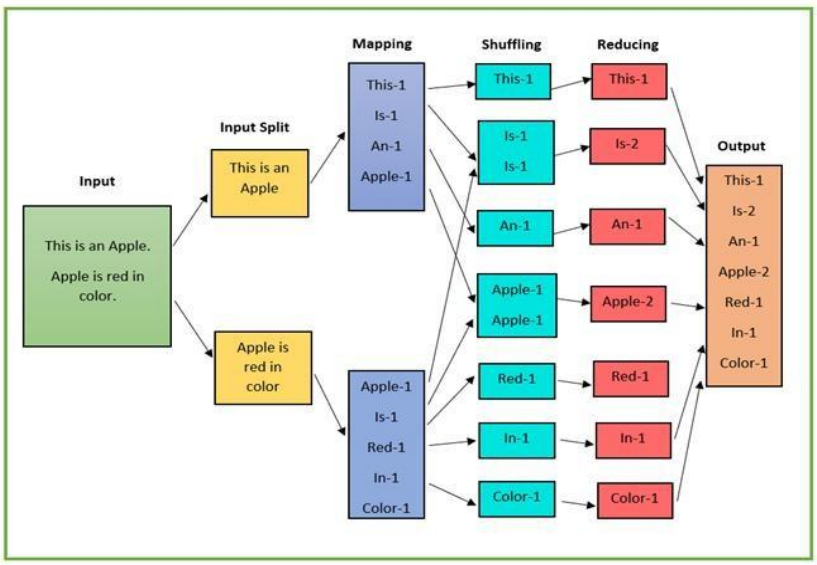


Figure 4.1: Word Count Example

This Java class extends Map Reduce Base and implements the Mapper interface. It defines a map function that takes the Long Writable key and Text value as input and emits key-value pairs using Output Collector. It tokenism each line of text into words and iterates over them. For each word encountered, it sets it as the key and emits a key-value pair with the word and a fixed value of 1. This class essentially maps each word in the input text to a count of 1, which is aggregated in the reduce phase for further processing, such as counting word occurrences in a document.

This Java class extends Map Reduce Base and implements the Reducer interface. It reduces the function by taking a Text key and an Iterator of Int Writable values as input. It iterates over the values, accumulating their sum. After processing all values for a given key, it emits a key-value pair with the original key and the sum as the value using Output Collector.

4.2.2 Word Count Mapper

```
public Static class Map Class extends Map Reduce Base
Implements
   Mapper<LongWritable, Text, Text, IntWritable>
{
      private final static IntWritable one = new Int Writable (1);
                private Text word = new Text ();
               public void map (Long Writable Key, Text
               Value, Output Collector<Text, IntWritable>
               output,
               Reporter reporter) throws IOException
               {String line = value.toString();
               StringTokenizer itr = new StringTokenizer(line);
               while (itr.hasMoreTokens()) {
                    word set(itr.nextToken());
                     output.collect(word, one);
               }
      }
```

This class essentially aggregates all the values associated with each key, producing a final sum for each unique key, which can represent various computations such as counting occurrences or calculating totals in a distributed environment like Hadoop Map Reduce.

4.2.3 Word Count Reducer

```
public static class Reduce extends MapReduceBase implements Reducer<Text,
IntWritable, Text,
IntWritable>
{
public void reduce(Text key, Iterator<IntWritable> values,
                   OutputCollector  <  Text,
                   IntWritable>output,
                   Reporter reporter) throws
                   IOException {
                   Int sum =
                         0;
                         while (values.hasNext()) {
                               sum += values.next().get();
                         }
                         output.collect(key, new IntWritable(sum));
                   }
            }
```

4.3 Proposed System

Two different datasets kinds can be used as joining relation inputs. Skewed datasets have an impact on execution time. I5-4590 quad-core processor, 8 GB of RAM, and a 2 TB hard drive File block A sizable amount of various sensor data is sent while interacting with an Internet of Things (IoT)system via a gateway and when a non-negligible amount of time has passed since the data was sent, the data is made available to patient surveillance IoT servers or medical professionals.As a result, warning signals that point out unusual patient situations tough to be transmitted (or pushed) more quickly than normal data. The gateway's job includes classifying and prioritizing the data received from AI Clouds and scheduling data transfers by those categories is 128 MB, and the Reducer frequency is 20 and objectives [108].

The Protocol Converter and the Monitoring Query Language (MQL) Scheduler are two modules that are included in the following phase, as depicted in Figure 4.3 represents. The Sensing module and the Network Manager module make up the bulk of an Assessment Designated Node (ADN-AE) self-aware is in charge of identifying and evaluating the sensor signals generated by AICLOUD users. The Remote Console module receives the sensor data, and it, using the information gleaned from the sensor data and the information collected from the Network Manager module, generates oneM2M signals. In an environment with common security (CSE), the security administrator, signal receiver, and load Balancer cooperate to keep the network

secure from outsiders. The Network Manager module is in charge of coordinating with ADN-AEs and other network administrators as well as managing the entire CSE procedure. The Management Console module keeps track of resource trees, which also include details about each thing that the IoT system is capable of controlling when it is active. It is made up of the Network Exposure (NSE), the Strategic Communications and Delivery Handling (CMDH), the Network Service Exposure Execution and Triggering (NSSE), and the Common Service Functions (CSFs) controller components.

That system includes the MQL Scheduler, which was created to facilitate scheduling for several classes using Q-learning. The CMDH module is in charge of communication (policy) and buffer management, whereas the NSE component is in charge of one M2M message transmission. Both CSF and NSE connections' sessions are managed by the NSSE module. Based on the MQL scheduling mechanism, which will be covered in greater detail in the following sections of this work, the MQL Scheduler module chooses scheduling options for sensor data transfers [109].

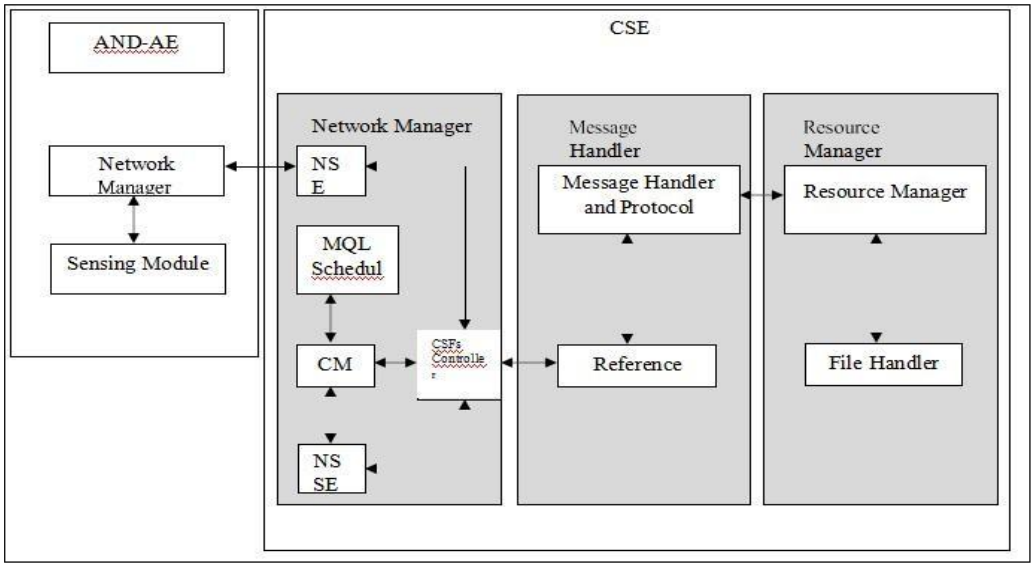


Figure 4.2: Internal Components of the one M2M(Machine-to-Machine) Internet of Things System

The one M2M (Machine-to-Machine) Internet of Things platform's core structure is displayed in Figure 4.2 . It is made up of the Reference Point and Message Handler/Protocol Converter modules. The Communication Receiver module converts OneM2M Basic and Hypertext Transmission Protocol (HTTP) messages. HTTP headers and Extended Markup Language (XML) are processed by the Estimation module. The oneM2M basic message to HTTP message mapping is displayed in Shown below Tables 4. 1 and 4.2.To undertake this investigation, software applications for protocol conversion to the oneM2M system were implemented. The oneM2M protocol and the ISO/IEEE 11073 standard are translated by the state's MN-CSEs. The Windows-based there are three C# classes in the protocol converting component.

They consist of the following:

The entire protocol conversion procedure is under the control of the AICLOUD Message Manager class. Additionally, it brings inbound messages to the protocol conversion module and sends them to the Message Handler for AICLOUD.The Resources Tree Manager's job is to construct resource trees for the gateways; therefore, the AI CLOUD Signal Handler class converts protocols and transmits its output to that person.The AICLOUD Message Template class manages multiple ISO/IEEE 11073 message templates. The use of templates speeds up protocol switching and allows for the speedy creation of related messages.

In Figure 4.2 above shows represents, modal analysis for the protocol conversion process is displayed. Skew Tune, a version of Hadoop that can dynamically balance the workload between mapping and reducing processes inside of a job, was created in response to Skew Reduces limitations claim. Skew Tune makes the premise that different invocations of the user-defined map and reduce functions are independent of one another since Hadoop programmers automatically re balance the load. Depending on the work, this can result in a re balancing of the burden at the level of individual records for map tasks and important groups for reduction tasks [110].

In a Map Reduce-type system, each operator operates independently from the others in the data flow. This means that each operator reads and writes data to the disc on its own, without being dependent on the other operators in the data flow. It is designed for systems such as Map Reduce. Similar to a systems analyst, Skew Tune constantly evaluates the workflow during a UDO to identify the specific task that is causing delays and taking up excessive time, thus hindering the project's progress. The raw input data is allocated to the available nodes that are ready to take over once a job of this nature has been identified and terminated. Re partitioning may occur when the resource is idle or when additional nodes are added to expedite the process, such as when utilizing spot instances in Amazon EC2. If there are no processes with significant data skew, Skew Tune will not impose any additional strain on the system as it will not be necessary. If a node is idle and there is at least one task that is expected to take a long time to complete, the task with the longest projected remaining time is terminated [111].

Based on the expected future availability of every node in the cluster, it partitions and parallelisms the remaining input data for that task, resulting in faster overall processing performance. Skew Tune operates in the background when not in use, identifying and removing bottleneck jobs as they appear until the project is finished. Skew Tune divides work according to the anticipated availability of each node in a cluster to expedite project completion. Depending on how well current activities are going, one can estimate the available resources. To make the most

of the nodes that are currently accessible or are anticipated to become available soon after a job has been completed, the remaining unprocessed input data is partitioned.

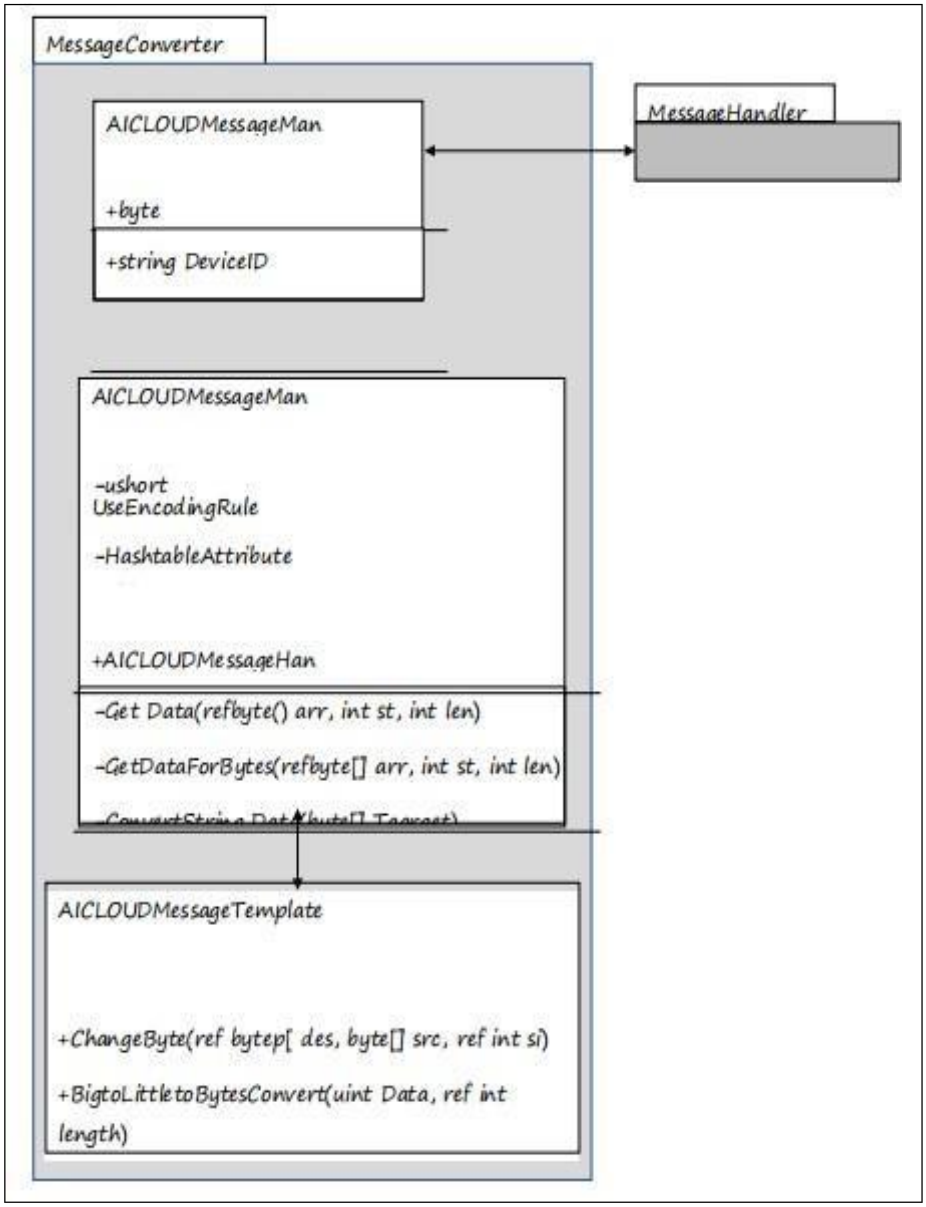


Figure 4.3: Program Modules for the Processor Protocol Conversion

Skew Tune also makes it possible to reproduce the original output by simply concatenating the output of split processes, which significantly decreases the detrimental effects of re-partitioning. The features of the Hadoop Map Reduce engine allowed us to employ the Skew Tune technique. According to the results of the tests, Skew Tune can reduce completion time by a factor of up to four without requiring any code modifications. It also puts the least amount of demand on the system's processing resources when Skew is absent. The following are only a handful of the lessons discovered: The Skew Tune technique also demonstrates that dynamic load reallocation is feasible for UDOs that adhere to a well-defined Application Programming Interface (API), such as Hadoop, and in which the system is capable of re-balancing load without generating an issue with the application. The cost of load re-allocations is small compared to the advantages because

of the possible savings. This method can duplicate a skew from any source, including cluster circumstances, hardware failure, mis configuration, improper input data order, and so on, making it more flexible than static planning.

The following are interconnected strategies that can be effective under challenging conditions: The use of skew-resilient operators, which can be implemented in a variety of ways, is one method for dealing with skew [112].Before the small partitions are dispersed acceptably, a certain number of small divisions in the input data are produced, and the cost of each partition is estimated. Second, divisions that have gotten too big are further divided using a different method.

Given its ability to utilize scalable resources for effective data processing, this activity is relevant to cloud computing. The deployment and improvement of MapReduce frameworks to efficiently handle data skew difficulties are made easier by cloud platforms, which provide elastic computing, storage and management options.structured formats . Large-scale data set data skew issues can be successfully handled by these formats, which allow for effective processing within MapReduce frameworks.

An analysis is conducted to investigate the causes of skew in Map Reduce applications, and the occurrence of the issue is assessed in three real Hadoop clusters. Skew Reduce and Skew Tune are two systems that use different methods to eliminate skew. Skew Reduce takes a static approach, while Skew Tune uses a dynamic approach. Future research should priorities enhancing programming interfaces, execution models, static and dynamic analysis, and multi-tenant optimization. Through the map reduction approach, we successfully identified the issue of reducer load imbalance during the joining step. When interacting with an Internet of Things (IoT) system through a gateway, a significant amount of diverse sensor data is provided, and Once a significant amount of time has passed since the data was received, it becomes accessible to patients monitoring IoT servers or medical experts. This approach offers greater flexibility compared to rigid planning as it can mimic various scenarios, such as cluster conditions, hardware failure, mis configuration, and incorrect input data order.

4.4 Implementation

Response Time (RT): Response time measures the average time taken to process and respond to a request within the IoT edge system. It is calculated by summing up the processing times for all requests and dividing it by the total number of requests. A lower response time indicates a more

RT = (Sum of processing times for all requests) / (Number of requests)

Throughput (TH): Throughput represents the number of requests or tasks processed per unit of time within the IoT edge system. It is computed by dividing the total number of requests by the

total processing time. A higher throughput indicates that the system can handle a larger number of requests within a given time frame, showcasing its capacity and scalability.

TH = (Number of requests) / (Total processing time)

Resource Utilization (RU): Resource utilization measures the efficient usage of computing, storage, and networking resources within the IoT edge system. It is calculated by dividing the actual resource usage by the total available resources. This metric provides insights into how effectively the system utilizes its available resources. Optimal resource utilization ensures efficient performance and avoids under utilization or over utilization of resources.

RU = (Actual resource usage) / (Total available resources)

Load Balancing Efficiency (LBE): Load balancing efficiency evaluates the balance achieved in terms of resource utilization and task distribution across the IoT edge system. It is calculated by dividing the standard deviation of resource utilization across nodes by the average resource utilization across nodes. A lower LBE indicates a more evenly distributed workload, demonstrating efficient load balancing and resource allocation.
LBE = (Standard deviation of resource utilization across nodes) / (Average resource utilization across nodes)

Fault Tolerance (FT): Fault tolerance measures the system's ability to handle failures or disruptions without significant impact on overall performance. It is calculated by dividing the number of completed tasks by the total number of tasks. A higher fault tolerance indicates a more resilient system that can gracefully recover from failures or disruptions, ensuring uninterrupted operation.

FT = (Number of completed tasks) / (Total number of tasks)

Scalability (SC): Scalability assesses the system's ability to handle an increasing number of devices, tasks, or requests. It is calculated by analyzing the change in system performance relative to the change in workload. A salable system exhibits consistent performance even with a growing workload, indicating its ability to efficiently scale up or down based on demand.

SC = (Change in system performance) / (Change in workload)

Energy Efficiency (EE): Energy efficiency measures the amount of energy consumed by the IoT edge system to perform tasks. It is calculated by dividing the total energy consumed by the number of completed tasks. A higher energy efficiency indicates a system that minimizes energy consumption while maintaining optimal performance, reducing operational costs and environmental impact.

EE = (Total energy consumed) / (Number of completed tasks)

To improve the classification capabilities of the hybrid load balancing optimization model for the cloud IoT edge, different machine learning models such as support vector machines (SVM) and decision trees (DT) must be explored and tested. While conventional neural networks (CNN) are often employed for image-based classification tasks, SVM and DT models have distinct benefits and may be better suited for certain cases.

SVMs that use both linear and non-linear classification issues. SVM seeks an ideal hyperplane in the feature space that divides various classes. We may assess the model's accuracy, scalability, and applicability for classification jobs in the cloud IoT edge environment by evaluating the hardware on SVM models. SVM is a helpful alternative to CNN because of its capacity to handle high-dimensional data and its competence in dealing with complicated decision boundaries.

Decision trees (DT) are a simple and easy-to-understand technique for categorization. Based on the properties of the incoming data, they build a tree-like model of choices and their outcomes. DT models are well-known for their capacity to handle both numerical and categorical data, as well as non-linear connections. We may evaluate the performance, interpret ability, and efficiency of DT models in the context of load balance optimization on the cloud IoT edge by adding them to the testing process. We may learn about the merits and drawbacks of SVM, DT, and other classification models by testing and assessing them alongside CNN in the context of the hybrid load-balancing optimization model. This allows us to choose the best classification. method based on criteria like accuracy, interpret ability, computing needs, and the peculiarities of the IoT data being analyzed. Finally, this multi-model assessment helps to ensure the load balance optimization model's efficacy and flexibility in a variety of cloud IoT edge scenarios.

By testing and assessing SVM, DT, and other classification models alongside CNN, we may acquire a thorough knowledge of their strengths and shortcomings in the context of the hybrid load-balancing optimization model. This multi-model assessment allows us to choose the best classification algorithm based on aspects such as accuracy, interpret ability, computing needs, and the peculiarities of the IoT data being processed. It also contributes to the efficacy and flexibility of the load balancing optimization model in a variety of cloud IoT edge contexts, since alternative models may perform better based on the individual application needs and datasets characteristics.

4.5 Simulation results for load balancing

We analyze the substantial performance improvements attributed to the proposed algorithms through simulations. To ensure impartiality, we utilize the Peer Sim simulation engine, which offers both cycle-driven and event-driven models.

Table 4.2 provides the simulation parameters and their respective values for our algorithms, derived from customer churn data related to network data services. Each trial of the simulation runs for 50 cycles unless otherwise specified. The maximum number of nodes is set to three times the minimum number of nodes to ensure variability and robustness in the simulations Each trial of the simulation runs for 50 cycles, unless otherwise specified." the maximum number of nodes is three times the minimum number of nodes. Thus, we set the proportion of nodes arrival to departure at 1%:1%, 1%:3%, and 3%:1% such that the number of nodes is kept the same, decreasing to roughly one-third, or increasing to 3 times, in 50 cycles. Finally, to indicate the reliability of the simulation results, each selected data point in our plots represents the average simulation result over 5 trials. Under the 220-node simulation scale, each trial takes about 10 hours. Shown in Tables 4.2 below list the parameters of our simulations, and the values we set unless otherwise specified.

4.6 Partitioning Skew in Map Reduce

The outputs of map tasks in a Map-Reduce application are distributed among reduced tasks through hash partitioning, which is the default method. During the map phase, a hash function is used to determine the partition number for each type of key-value pair. This partition number corresponds to the number of reduced tasks. The hash function is typically sufficient for evenly distributing the data. If the outputs are not evenly distributed, hash partitioning may fail due to skewed data. This occurrence is known as partitioning skew. As an expert in analyzing systems, consider the Inverted Index application. In this case, the hash function can divide the intermediate data based on the initial letter of each word. This means that reducers handling more commonly occurring letters will have a larger amount of data to process. Partitioning skew may arise due to various factors [16].

$Data : A : \{a_1, a_2 ... a_n\}, K$
$\text{Re}\, sult : R : resource - allocation$
$low \leftarrow \max\{a_i\}$
$high \leftarrow \sum_n^1 a_i$
$num \leftarrow 0$
$while(low < high)do$
$mid \leftarrow low + (high - low)/2$
end
$return$

Uneven workload distribution can occur due to the significant variation in value sizes in applications. Imbalanced key frequencies: Certain keys have a higher occurrence in the intermediate data, resulting in an overload of reduce tasks that handle these popular

keys.Execution times can vary depending on the size of the key-value pairs being processed. Processing a single, large pair may take longer than processing multiple smaller pairs. Even if the partitioning function evenly distributes keys among reducers, the execution times of reduced tasks can still vary due to the varying number of values in the assigned key groups.During the re-partitioning process, the partition results from the map phase are divided and combined once again [17]. The key-value pairs in one partition are therefore separated and merged into another. When a reduced task requests partition data based on the results of a new partition, the distributed data is stored in various locations within the split files, leading to a non-sequential and inefficient data reading process. One of the main challenges in virtual partitioning is determining the appropriate partition number for the key-value pairs of R based on the hash key %R function. Typically, the number of reduced tasks in R is determined by the types of input key-value pairs, rather than being set as a default value. We believe that the optimal number of virtual partitions falls within this range. Once the value of R is determined, the partition number no longer corresponds to the reduced task number based on the hash key %R. Each partition in the map phase can be processed by a reduced task that has some uncertainty associated with it. This type of partition is referred to as a virtual partition.

Every virtual partition is a crucial component of a physical partition that has been restructured. Once the reduced tasks have all the necessary information from the map phase, a balancing algorithm determines the specific relationship. The importance of virtual partitions is to distribute the key-value pairs as widely as possible, thereby offering a greater variety of combinations for the subsequent re-partitioning process. Users have the freedom to choose the number of virtual partitions N based on the characteristics of the application, system resources, and the degree of dispersion of key-value pairs. We divide the output of all map tasks into virtual partitions to ensure a fair distribution among reducers. Nevertheless, the performance of the partitioning phase can be greatly impacted by the quantity of virtual partitions. When there are only a few virtual partitions, the system can retrieve the metadata information for each one more efficiently. On the other hand, a decrease in virtual partitions may result in an inequitable allocation among reducers.

Gathering worldwide output data. Similar to a systems analyst, the re-partitioning process optimizes the communication between map tasks and reduced tasks by dividing the original process into two phases: (1) gathering the metadata output of each map task and (2) combining the information in the reduced tasks. Figure 4. 2 represents illustrates the process of acquiring metadata for reduced tasks. Here are the step-by-step instructions:After all map tasks are finished, the output is stored on the local disc. The Task Tracker utilizes heartbeat information to communicate with the Job Tracker and notify it of task completion.The Job Tracker maintains a queue for each Map Reduce job to track the completion of map tasks. When a reduced task in the

Task Tracker requests a completion message for the map task, the Job Tracker promptly removes the message from the queue and delivers it to the appropriate Task Tracker.Within the same Map Reduce job, a reduce task receives a completion message for the map task from its Task Tracker.

Extracting the runtime information of the map task from the completion message involves retrieving details such as the map task number and execution node information. With this information, the reduced task establishes an HTTP connection with the execution node and requests the metadata information output of the map task.

When a map task is assigned a request number, the Task Tracker retrieves the relevant index file of the map outputs from the local file system and transfers it to the corresponding reduce task.The reduce task combines the virtual partitions with matching index numbers from various index files. It collects the data from each virtual partition that shares the same type of key-value pairs.

The re-partitioning process divides the collected virtual partitions into new partitions with the same number as reduce tasks, just like a systems analyst would do. By optimizing the re-partitioning process, it is possible to reduce the data size of the largest partition. Additionally, it can decrease the processing time required for the maximum partition, resulting in faster completion of the entire reduced phase and an increase in the rate of completed jobs and system throughput.

Table 4.1: Simulation Environment

Environment	**Default values**
Network size	2
Ratio of node arrival:departure	1%:1%1%:3%3%:1%
Number of meta data for each node	10
Node capacity distribution	Pareto:shape 2
Simulation cycles	50

Table 4.2: Algorithm Parameters

Algorithm parameters	**Default values**
α	1.4
β	25%
ϒ	2.0
κ	4inf irst10 cycles ,2in11to20cycles,1inlast30cycles

4.7 Results and Discussion

This MATLAB function 'go' simulates message propagation along a given path, updating the timeline and energy consumption. Initially, pastime is set based on the message duration. It, for each node in the path, the timeline is updated incrementally. Energy consumption is calculated based on node-to-node distance and message size. If it's not the last node, energy is computed considering transmission and reception. Otherwise, only reception energy is considered. Finally, the function returns the timeline and energy consumption arrays.

In this investigation, we have provided a cloud environment with 50 nodes, and the above diagram demonstrates the AI for cloud message position for that environment. When there is a single source and 50 or 100 destination nodes for two different cloud environments, the message transmission techniques in Below Figure 4.5 are utilized. The simulation model underwent a total of 100 permutations and combinations. The result description includes an attachment that lists the top five combinations for each cloud environment.

50 nodes make up Cloud 1 and 100 nodes make up Cloud 2. The message communications for Cloud 1 are shown in Figures 4.4 and Figure 4.5 represents whereas the message communications for Cloud 2 are below shown in Figures 4.6 and Figure 4.7 ,Figure 4.8 represents. Based on the findings, we developed the suggested method using MATLAB to achieve more than 90% accuracy compared to earlier exciting when a communication network grows in size.The more accurate and precise the model is in the particular cloud, the better the communication protocol will be. MATLAB node-to-node communication is seen here.

```
Step 1: function [T, E] = go (msg, path)
        global position N
        timeline=zeros (1, N);
        Energy=zeros (1, N);
        Pastime = msg (2)/1000;
        Pt = passtime;
Step 2: for i=1: numel (path)
        timeline(path(i)) = pt;
        pt = pt + passtime;
step 3: if i < numel(path)
Energy(path(i)) =100*msg (2) *1e-6+100e-9*msg (2) *dist(position(:,i)',position(:,i+1))^2;
        else
        Energy(path(i))=50e-6*msg(2);
        end
        end
        T=timeline;
        E=Energy;
        end
```

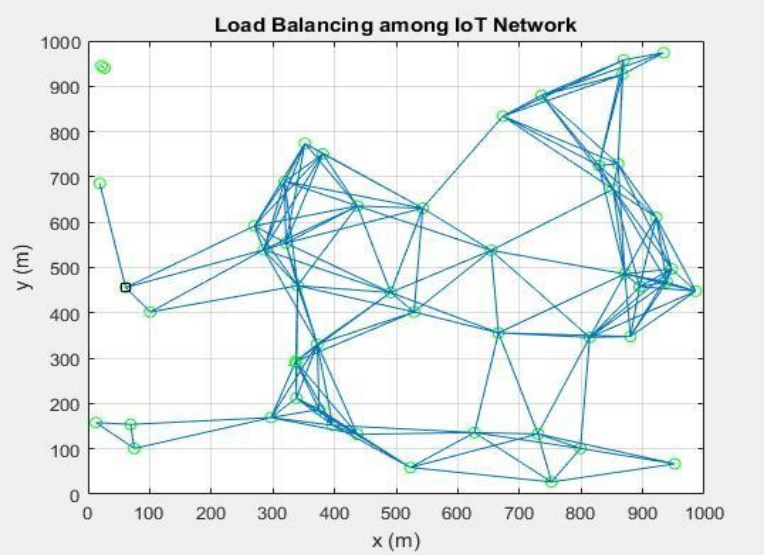


Figure 4.4 : The AI Cloud Messenger Position for Cloud1 with 50Nodes.

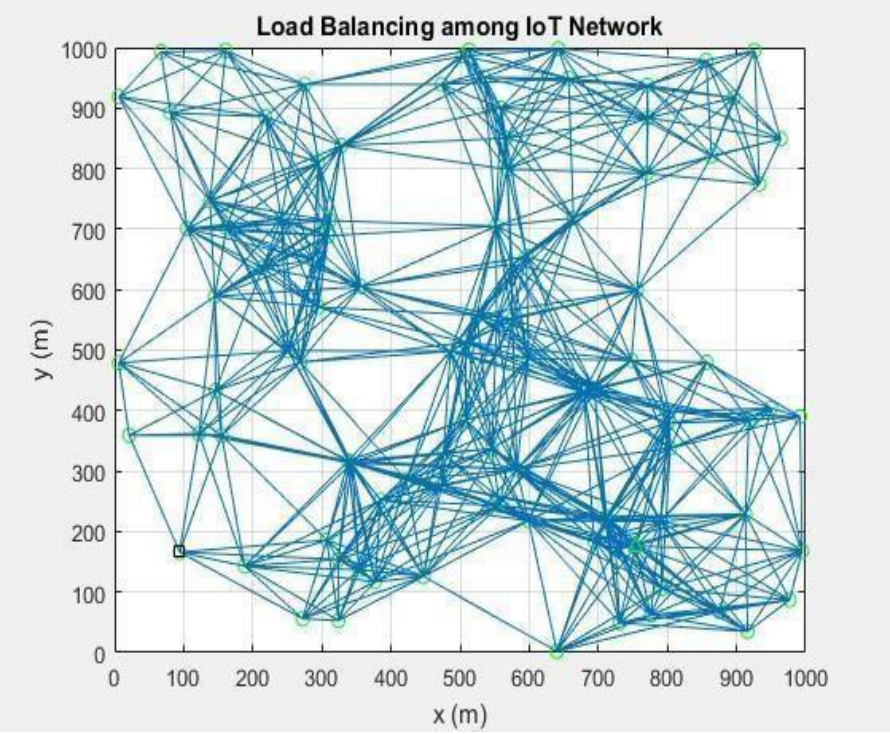


Figure 4.5 : The AI Cloud Messenger Position for Cloud2 with 100 Nodes

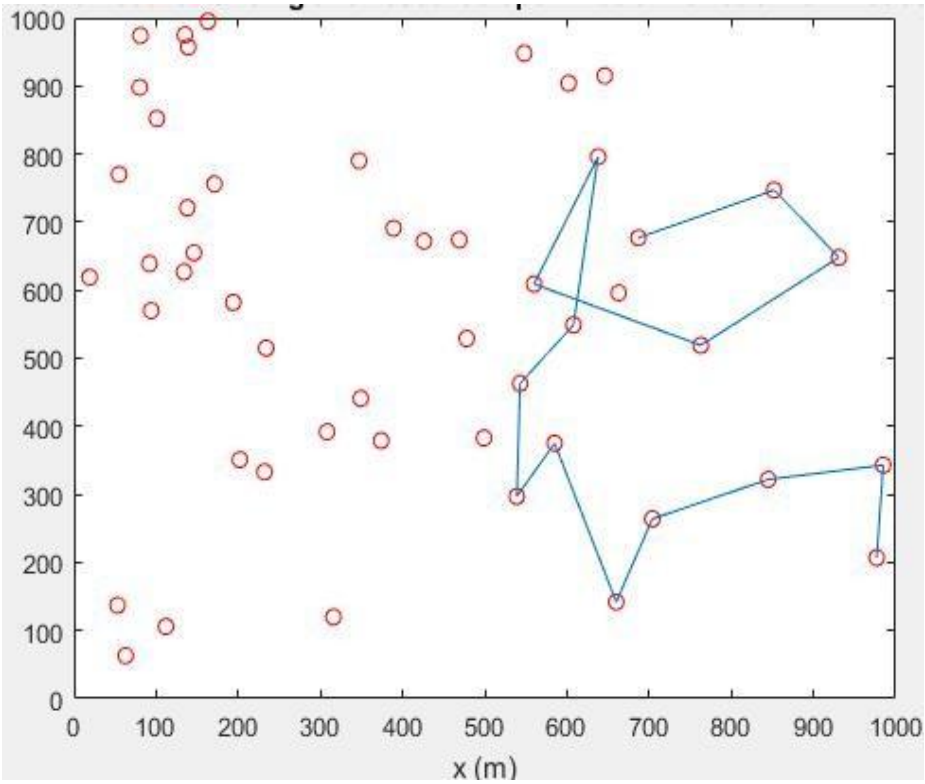


Figure 4.6 : The Different Path of Messages over the Node for Cloud1

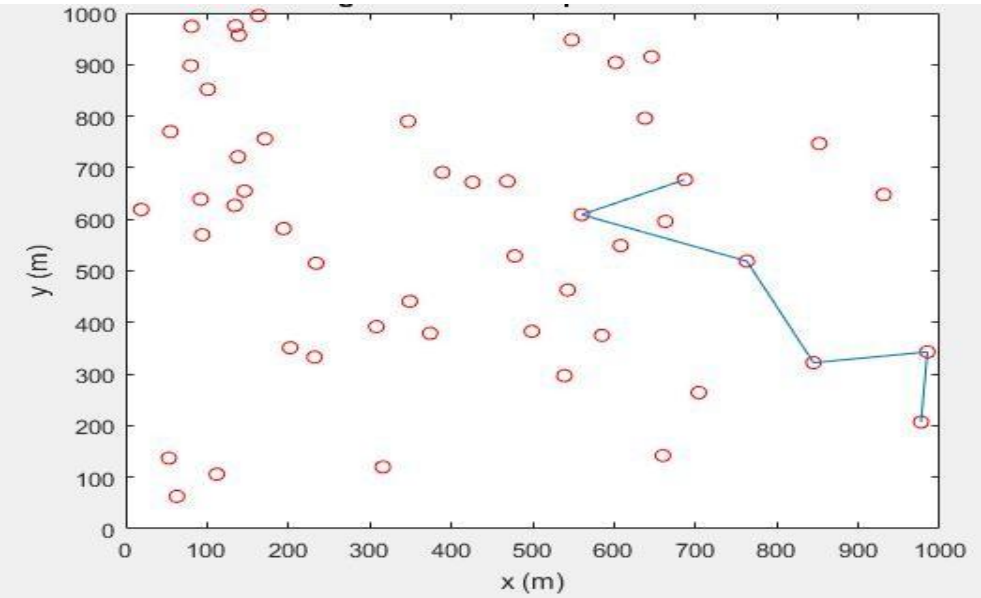


Figure 4.7 :The Different Path of Messages over the Node for Cloud2

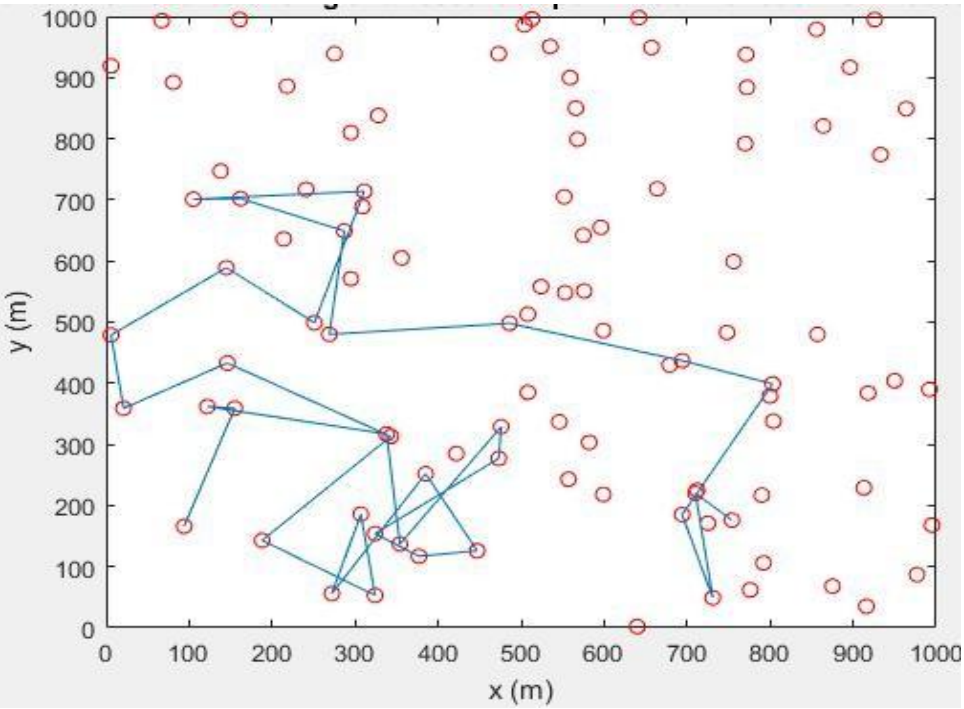


Figure 4.8 : TheDifferentPathofMessagesovertheNodeforCloud2

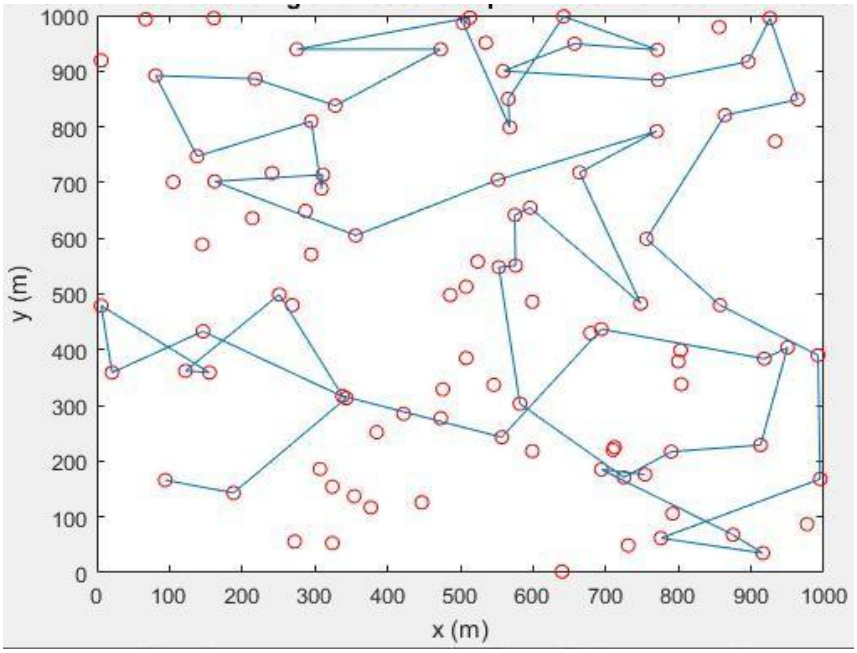


Figure 4.9 : The Different Path of Messages over the Node for Cloud2

4.8 Summary

Map Reduce API gives users a lot of freedom when creating UDOs, it also puts a lot of responsibility on their shoulders to handle a wide range of performance difficulties. Skew is one of the challenges that many individuals are dealing with regularly these days. We first look at the causes of skew in Map Reduce applications, and then we investigate whether the problem exists in three actual Hadoop clusters. We gave an overview of two systems that, respectively, use static and dynamic methods to remove skew: Skew Reduce and Skew Tune. Future studies should concentrate on enhancing programming interfaces, execution models, static and dynamic analysis, and multi-tenant optimization. We identified the reducer load imbalance for the joining procedure in the map-reducing approach.

CHAPTER 5

Novel Framework for Resources Optimization to Solve Class Imbalance Problems

To generate a solid and efficient data set, order to produce a solid and efficient data set, the use of an under-sampling methodology depends upon properties chosen and suggested in this study. Compared to other algorithms, this method is compared to other techniques shown to be more resilient.

5.1 Introduction

Numerous certifiable spaces include class imbalance datasets problems, including identifying erratic memory management, text analysis, splitting tasks, divulging callers, language diction scooping up, looking out petroleum hydrocarbons in remotely sensed photos, publicizing clients, most especially medical research. Within those situations, dominant ruling groups typically exert self-serving influence over the assessors, which causes the classification to exhibit traits of minority categories in an unsatisfactory manner; ultimately, a classifier addresses everyone as the majority class and ignores the minority class. Different writing-based procedures have been put in place to address issues related to awkwardness in class [113]. These days, several factors need to be considered simultaneously, including class coverage, paired courses, the expenditure of miscalculation and other class difficulties. This makes research subjects like this one both fascinating and challenging. Issues of dual classes linked to cut-the-head data have made headlines but issues of classifier balance with diverse sorts of issues are not truly treated. Feature Data: Details or characteristics that each dateset instance is described by. Feature data, which contains numerical, categorical, or textual data pertinent to the issue area, is essential for training machine learning models.

1 Class Distribution Data: Details on how the classes are distributed throughout the datasets. This information directs the optimization process and aids in determining the degree of class imbalance.

2 Computational Resources: The resources needed to run the optimization framework, such as memory, computing power, or storage capacity. To achieve scalability and performance, computing resources must be used efficiently.

3. Cluster Data: Information produced by clustering techniques that illustrate how related instances within a data set are grouped. By resolving class imbalance, this information is used in

the optimization process to improve classification performance.

4. Model Performance Metrics: Measures include accuracy, precision, recall, F1-score, and area under the ROC curve (AUC) that are used to assess how well categorization models perform. The optimization process can be improved with the use of these metrics.A portion of the datasets called the "training data" is utilized to train machine learning models. Building predictive models that can effectively address class imbalance requires training data.

5.2 Methods for Dealing with Skewed Data Streams

In Data Stream we are using Two methods

a) **Under-sampling:** Under-sampling, another examination approach, solves the issue by lowering the proportion of the elephant's mutual fund appearances. This is often affected by swapping between the majority class events or by randomly selecting the appropriate number of majority class role models. The clustering approach is typically used for under-inspecting. The best delegate from the dominating part class is chosen using bunching, and the preparation lump is changed as necessary. The next section examines some of the writing's under-inspecting-based approaches. A separate methodology was used to regulate data skew streams. They employed a grouping + sampling strategy to manage a skewed data stream. The centrists from each of the ten individuals were utilized to finish the experiment after a series of negative models were generated utilizing the current preparing story's k-suggest calculation [118].

b) **Customary Approaches**: Each one of the parameters was taken into account when creating the groups. The prepared cluster It was updated by collecting all particular models from around centrists of the significant adverse models once the range of groups created was equal to the value of giving rise in the current treatment clump. These studied events were used to create another classifier. The outfit's additional dimensions were set such that any classifier present in the gathering may have a nearby replacement based on examples that had been analyzed. The top classifiers to work with the firm were chosen using the AUROC metric

5.3 Implemented Design for Resource Optimization to solve Class Imbalance problems

Educational Pricing The cost structure is a combination of internal and strategic factors. It combines various objectives, incorporates changes at the computational and informational levels, and modifies the learning process to incorporate pricing and independent cost assessments. [120]. When considering the reduction of the total error price of the three classifiers and taking into account the higher misdiagnosis cost of the minority class, the computer classifier tends to exhibit a bias towards that particular group. Furthermore, incorrectly labeling someone as a breast carcinoma patient is considered to have a low rate of false positives (although it was previously certain outsourced judgmental). However, this is a highly delicate and expensive situation to

correct distortions, as demonstrated in an analysis of the complex costs associated with a specific minority group of neoplastic individuals.

A client's unfortunate demise can occur due to a misdiagnosis or failure to obtain the necessary treatment [121]. When considering ways to reduce categorization and overall test costs, it is important to take into account the cost lattice for cost-sensitive learning and the costs associated with defining and upgrading them. Distinctly approaching the classes safeguards the integrity of the classifier model based on the Back Propagation algorithm. Information bias has been a long-standing issue. Several AI systems have been developed in the past decade to address information imbalances. Given the dynamic nature of this field, numerous challenges need to be addressed, particularly when dealing with imbalanced and intricate samples.

The specific wording [122] has taken into account a wide range of possible methods for handling class irregularity. Examining current data and6 selecting several examples of each classification that are excessively large in comparison to others is one method to stop the spread of data class knowledge. This method is referred known as under-sampling. Arbitrarily resampling has the main problem of potentially discarding data that could be crucial for the lesson. However, as this cycle forms the precision of previous examples, over fitting may happen more frequently for anomalous frame interpolation.

5.3.1 Data Multi Dimensional

An emphasis extractor or included evaluation can be used to reduce the dimensions. To create a cheap vector of lights, a combination of user satisfaction in the second phase, as opposed to the primary technique, which eliminates factors with a negligible influence on the output [123]. The three major categories of existing element determination techniques are channel, covering, and installation strategies. They alter how the choice computation and model structure are combined.

Filters choose highlights while giving the model minimal consideration. They only rely on general characteristics, such as the relationship with the anticipated variable. Wrappers employ learning techniques to analyze which highlights are useful. To finish the examination of the subgroups of highlights, a grouping model is employed. The spotlight subset found in the selection interaction is acceptable for the particular computation since one framework is employed for both assessment and characterization.

In embedded strategies, the expansion of the classifier and the element selection step are intertwined. The best arrangement of highlights is produced when the classifier is generated, and the classifier's predictions affect the choice.Considerations for information asymmetry when choosing features The following requirements should be met by a sound component selection strategy. a few guidelines, namely: reliability, validity, and repeatability. The designers have

deduced that, at the very least, several tests are anticipated to achieve reliable component selections. The lopsidedness of the classes is another source of highlight determination's unpredictable results. The development of several fake preparation examinations is one strategy for handling the aforementioned problem. It, using both the new information and the earlier ones, highlights subsets that may be polled. In previous exams, we have experimented with oversampling methods as well as other include determination methodologies. We have demonstrated how a certain layout can significantly impact the accuracy of grouping. As a result, not every data modifying plus emphasis selection strategy mix is highly useful.

5.3.2 Skew rate consumption

Considering the data skew rate is crucial when optimizing load balancing for IoT edge devices. An elevated data skew rate can significantly impact the performance and efficiency of load-balancing algorithms and models. It could lead to imbalanced resource utilization, longer processing job response times, and inefficient allocation of computer resources. Statistical measurements like the coefficient of variation, Gini coefficient, or entropy can be used to determine the data skew rate. These metrics assess how data is spread out and distributed among different entities or divisions. As an example, the coefficient of variation can be calculated by dividing the standard deviation by the mean of the data distribution. Similarly, the Gini coefficient is a statistical measure that evaluates the level of inequality in a data set. On a scale of 0 to 1, the value of 0 represents perfect equality, meaning there is no skewness. Conversely, a value of 1 indicates maximum inequality, suggesting high skewness. Entropy can also be utilized as a metric to evaluate data skewness. It measures the level of uncertainty or information contained in a data set. Similar to a systems analyst, one can observe that higher entropy levels suggest a greater degree of data skewness, suggesting that the data is more widely spread or unevenly distributed. By calculating and evaluating the data skew rate, you can gain valuable insights into the distribution patterns of data in the IoT edge system. This data can be used to enhance load-balancing algorithms, efficiently allocate resources, and ensure the system can handle uneven data distributions.

5.3.3 Novelty of the system

The methods for dealing with skewed streaming data. We provided a technique for efficient learning at a low cost for resource optimization. It fixes the data difficulties with skew. Unless otherwise specified, the classifiers and component selecting techniques utilized their default upsides of boundaries. By under sampling the data, we were able to get a 75% accuracy, but the remaining data was essentially wasted. By under-sampling and employing a cost-effective learning strategy, we were able to obtain a 71% accuracy. The oversampling method helped us

achieve an accuracy rate of 78%. The model works better with oversampling, enabling us to get perfect results.

5.4 Implemented Design of Cost-Effective Learning Method for Resource Optimization to Solve Class Imbalance Problems

The idea underlying the used strategy is as follows:

With large datasets, certain include conventional methods are employed. The risk can be minimized by using a pipeline, either with or without taking the pricing structure into account. In any instance, before the readiness material is evaluated by the investment grid, the lattice evaluates the broad set to see whether the basic analyzer can manage event loads.The shorter the matrix of spotlights in the collection, the more SMOTE technique is comfortable with oversampling minority groups. The work assignment is completed for the information set up as shown above. A few metrics that are appropriate for unbalanced information are tried to evaluate the channel presentations both with and without weighting.

To complete the research, the obtained results are also contrasted and these Depending on how the material was initially arranged, judgments (as opposed to one that has all of the highlights per-selected, for instance.).It can be challenging to choose the right features for the expenditure attice. Typically, the workloads are chosen so that each group receives their maximum number, the cost of mis classification is quite equivalent. This method does not always work, and in many cases, it is important to choose alternative features that will more effectively lessen information sleekness's adverse effects the recently published study focuses more time and effort on the inaccurate classification of models as advantageous. A class of negative people exist. By deducting the total of the tests taken by learners in the greater section of the class from the total of the tests taken by students in the minority class, the expense for such a minority class is calculated. It should be noted that the cross-approval system implemented both over fitting and element selection. An analysis using a student MS test with such a 95% significance level is used to evaluate changes in characterization accuracy. An overview of the actualized idea is shown in Figure 5.1 represents.

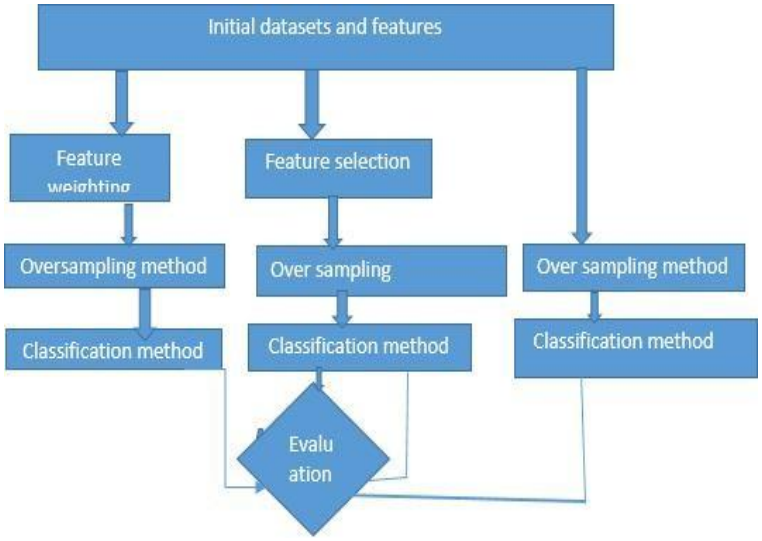


Figure 5.1: Developed strategy for the issue of minority classification allocation

5.5 Methodology

Load balancing involves distributing the workload evenly across available IT resources to ensure fairness. Ensuring the continuous operation of a service, even in the event of a failure, is of utmost importance. This involves implementing procedures that optimize resource utilization. Load balancing is essential for reducing task delay and maximizing resource utilization, leading to improved system performance and cost savings. It offers great versatility and flexibility for users with different dimensions that may change in the future, requiring additional IT resources. Another objective is to minimize energy consumption and carbon emissions, while also preventing traffic congestion by providing necessary resources and meeting quality of service standards [9, 13]. Consequently, it requires a suitable load planning mechanism that takes into Our approach stands out due to the distributed computation abilities of the LSTM, where each node independently constructs its load vector by collecting data from other nodes. Using a distributed approach, our model is designed to make decisions based on local load vectors. This approach can be used to address dynamic and adaptive system topology. It takes into account the current state of the system during load balancing to detect any changes in system status. Additionally, it allows for the dynamic adjustment of parameters. Our approach enhances system efficiency by reducing task response time and maintaining acceptable delays through the implementation of distributed computational ability. we have integrated hardware NODEMCU AI cloud with local cloud were our hybrid cloud was created, and then our model was validated. The ANN technique proposed for our research work, is entirely depended on the cloud infrastructure we are using. In our case we have used hybrid deep learning model for learning

accuracy to be optimized.

These methods typically treat tasks in a first in, first out (FIFO) manner. Approaching tasks in this way may not be the most effective scheduling strategy, especially when tasks have different latency constraints and computational loads. In our proposed approach, numerous cloud cells can be included in both the US26 and Euro28 networks to compute the task as required. The novelty of our approach has two significant contributions to knowledge. It offers a flexible design that allows for customization of metrics such as scalability, performance, response time, and associated overhead. Scheduling rules and clustering objectives can also be adjusted to meet the specific needs of applications and network requirements. Although our approach utilized only the US26 and Euro28 networks, this design can be adapted to meet the requirements of other networks. Additionally, our strategy focuses on simplifying the optimization of multiple parameters. This offers a high level of perceived user quality and a satisfactory service level agreement (SLA).

A recurrent neural network equipped with LSTM [16]. Training long temporal relationships in a regular recurrent neural network can be challenging due to the issue of gradient evaporation or inflation over time. However, LSTM has the potential to create long-term dependencies by maintaining a consistent flow of errors through 'constant error carousels' (CEC). Numerous modifications have been implemented to the original LSTM since that time. A thorough examination was conducted on the implementation of LSTM in Sak's "predicted" form. LSTMP devices are equipped with input and output gates. The input gate of the LSTMP unit regulates the control signal going into the memory cell, while the output gate manages the data being sent out. The forget gates of LSTMP enable the dynamic process of forgetting and resetting memory cells.Every LSTMP unit consists of a projection layer that is both recurrent and non-recurrent. Two projection layers are consolidated into a single layer. The LSTM Neural Network is a variant of the Recurrent Neural Network (RNN) that effectively addresses the issue of the growing gradient problem. The neural network's efficient back propagation of the error correction is hindered by this gradient problem, which is a new fact. Consequently, it cannot acquire information from extensive datasets, indicating that the RNN has a limited memory. This limitation prompted the creation of the Long Short-Term Memory variant. The construction of the LSTM is depicted as a chain, accompanied by a solitary memory cell. Every large square block in this picture represents a memory cell.

5.6 Results and Discussion

Python environment under the Windows operating system will be used to conduct the experiment by summing the findings of the 10 rounds of cross-validation used during the testing, the total display rating was produced. Each overlay contained images out of each class to approximately the very same extent, as per the criteria of acceptance. Both the component-choosing processes and the classifiers that were employed used the default upsides of borders. Unless otherwise established in any circumstance. For resource efficiency and to address the issues with data skew, we created an affordable learning approach [124].

5.6.1 Re-Weighting

The costs for instances belonging to minority classes are larger using this loss function. By utilizing the parameter balance and the reweighting function provided by the package, we can determine the class weights for imbalanced datasets. These weights correspond to the weights given to each class.

5.6.2 Learning Rate Scheduler

The optimizes default approach for calculating learning rate values is a constant learning rate. Selecting the proper learning rate is a challenging procedure since doing so maximizes training optimization. To start, our example demonstrates that our experiment operates effectively with l r =0.001. This might serve as a base for many studies with learning rate algorithms. To more effectively accommodate specific situations, the gradient descent schedule controls the optimizes back propagation algorithm during the testing period. Whether or not to alter the learning rate depends on changes in the value of the loss function [125]. There are several types of learning rate schedulers.

5.6.3 Warm-up Learning Rate

This approach gradually lowers the faculty's pace of development at the start of the testing period before gradually raising it after an agreed-upon quantity of eras. Eventually, the fundamental parameter has been reached. Despite jogging with large-scale datasets frequently leading to early fitting, it is still feasible to train against this problem.

5.6.4 Step Decay Learning Rate

For a set number of iterations, this method alters learning speeds. The learning basics rate is set up using a system call, which supplies it with specified numbers.

5.6.5 Cosine Decay Learning Rate

In this approach, we first construct a learning phase rate and It calculate the dropping learning rate by multiplying it by the aging function. Its learning rate will thus decrease during this test datasets.

5.6.6 Adaptive Decay Learning Rate

This method lowers the proportional gain during an epoch whereas if the error rate rises, and raises it during an epoch if the percentage error falls. To avoid over-flattering, the parameter slowly rises if there is not an improvement in deviation. In this research, an adaptive learning rate scheduler will be used to train the long-tailed data set. We may also assess our trials using various schedulers for learning rates.

5.6.7 Data Augmentation and Resampling

There are two easy ways to deal with uneven datasets: resampling and increasing as an example. Resampling is a data collection technique that involves using data from the majority class initially and then giving the minority class additional information. This procedure has two variations: under sampling, where information is omitted, and oversampling, when data is included [126].By selectively augmenting the data, we can resample and improve the balance of the dataset. An image can only be improved by rotating it by 10 degrees and increasing its brightness from 0.2 to 1.0. (We are also capable of enhancing data in other ways.)

5.6.8 Change Loss Function

Focused reduction is a typical error term used to solve the issue of minority class. Focal loss aims to reduce numbers that the model has accurately predicted, as opposed to seeking out outliers or bad predictions. In plain English, the loss function calculates the amount of estimating error in

5.6.9 Label Smoothing

In training circumstances, if the classifier gets too optimistic, it could generalize well, especially when employing imbalanced datasets in classification issues, regularization approaches such as label smoothing are routinely used to keep the model from becoming cocky and arrogant in its label projections [127].On predicting customer attrition, we have worked. To determine the reasons why consumers are leaving a firm, use customer churn prediction. Your data was initially integrated into the system. Age, Relationship, Descendants, Job, Multiple Phones, Internet Service, Users Require, Tech Help Agreement, cashless payment, streaming media, and television Repayment Approach, Rotation, Overall, Fees, and Weekly Fees.

Data: Create a hybrid cloud infrastructure by joining NodeMCU devices to local servers for effective data collection and network distribution

The source node s(d) is requesting a set of available resources from the cloud data centre.

Result: Task assigned to the resources in a well-balanced order.

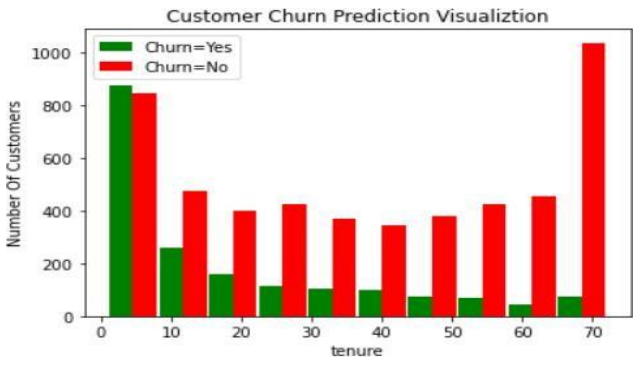
By combining cloud services with on-premises infrastructure, a hybrid cloud configuration maximizes resources to overcome class imbalance. Initial data preprocessing and storage are handled by on-premises servers, which lowers data volume and ensures security. Scalable cloud instances handle intensive computations like model training and hyper parameter tuning. Protocols for safe, fast data transfer make it easier to communicate across environments. Tasks are distributed via dynamic load balancers in accordance with available resources and workloads. Utilizing the advantages of both local and cloud resources, auto-scaling and continuous monitoring guarantee high-performance and low-cost operations while addressing class imbalance.

Update the availability of resource nodes.

Perform a search to check for availability. During the search:

Retrieves nodes Match the balanced resources to the task nodes If no task is available, it is set to no task End Attribute values cannot be handled directly by the method. We achieved a 75%accuracy by under-sampling our data, but the left over data is effectively thrown away

Figure 5.2: Displaying the Visualization of the Churn Fore cast



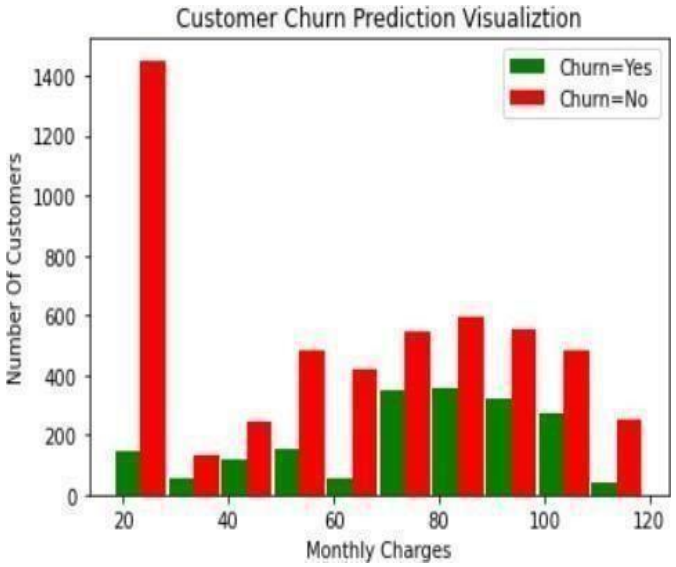


Figure 5.3: Displaying the Churn Forecast

Table 5. 1 :Under Sampling

Class report	precision	Recall	f1-score	support
0	0.84	0.72	0.77	1034
1	0.76	0.86	0.80	1034
Accuracy			0.79	2067
macro avg	0.80	0.79	0.79	2067
weight avg	0.80	0.79	0.79	2067

Table 5. 2 :Over sampling

Class report	precision	recall	f1-score	support
0	0.90	0.68	0.78	1033
1	0.47	0.78	0.59	374
Accuracy			0.71	1407
macro avg	0.68	0.73	0.69	1408
Weighted avg	0.78	0.71	0.73	1408

With the oversampling technique, we obtained a 78% accuracy rate. Up-sampling improves the model's performance. Using an expense learning strategy on under sampling, we were able to gather the knowledge required to convert the categorical variables into a control system of 71%. Format that is acceptable to the algorithm. Here, the data is being converted into numerical using one hot encoding technique[128].

Table 5. 3: Classification

Classification report:f	**precision**	**Recall**	**f1-Score**	**support**
0	0.76	0.76	0.76	375
1 r	0.75	0.76	0.76	374
accuracy			0.76	749
macrology	0.76	0.76	0.76	749
weighted avg	0.76	0.76	0.76	749

5.7 Summary

By fusing clustering techniques and classification algorithms, this framework offers a novel solution to issues with class imbalance. The framework uses clustering to find subgroups within each class rather than addressing the imbalance directly in the datasets. The class imbalance problem can be better handled by the following classification models by optimizing resources through this clustering stage. By initially classifying the data into meaningful clusters, this strategy presents a fresh viewpoint and improves the performance of subsequent classification tasks. The system attempts to show better performance than conventional approaches for managing class imbalance in machine learning through experimentation and validation.

Chapter 6

A Robust Resource Allocation Model for Optimizing Data Skew and Consumption Rate in Cloud-Based IoT Environment

The cloud-dependent resource allocation strategies for IoT environments include some subgroups like Scalability, Perspective, Bandwidth, Electricity, and Expenditure prioritization have been explored and characterized. The IoT framework also provides possible facilities to raise the IoT ecosystem's efficiency on the cloud.

6.1 Introduction

In modern computing systems, workloads in Map Reduce may experience delays for various reasons, with data skew being a significant factor. To enhance efficiency and mitigate skew-related issues, this chapter examines different types of skew prevalent in the Hadoop Map Reduce architecture. Strategies aimed at reducing skew, including Skew Tune (dynamic), Skew Reduce (static), LIBRA, and LEEN, are also presented and discussed. Foundational characteristics of IoT, internet, and resource top management.

6.1.1 Internet of Things

The network of things provides the consumer with access to a wide range of advantages and features. As a result, several components are required for effective utilization. The next paragraphs will go over the IoT components.Below shown Figure 6.1 depicts the pieces needed to give IoT capabilities.

The Web of Things offers the following characteristics and benefits to consumers:

A).Enhanced convenience: With the help of a cell phone or another attached gadget, customers may remotely operate and monitor their gadgets at any time and from any location thanks to the Web of Things.

B).Greater energy effectiveness: By automatically adjusting their energy consumption in response to real-time data, smart devices may reduce energy waste and utility bills thanks to the Internet of Things.

C).Increased safety and security: linked webcams and door locks, for example, may keep families and residences safe by sending out real-time warnings and allowing for remote monitoring. Personalized experiences: With the incorporation of sensor and data analytic s, the Web of Things makes it possible for devices to adapt to the preferences and needs of each

user.Easier maintenance and repairs: Consumers may maintain their connected devices in excellent working order by receiving notifications when repairs or maintenance are required.

a) Identifiers

Given its ability to accurately identify every object in the network, the process of object identification plays a vital role in the design of the IoT. Recognition consists of two essential steps: naming and addressing. Addressing refers to the location of the item, while naming is the label assigned to it. While they are interconnected, distinguishing someone through naming and addressing are distinct methods. Several techniques, such as the implementation of universal identifiers like electronic product codes (EPCs) and Code, have the potential to enhance the efficiency of item naming within the network. These techniques help with assigning unique names to network objects, promoting efficient data transfer and communication. Dealing with [32] is a crucial step in the identification process.

Every component of a connection has its unique IP address, which simplifies its identification on different networks. Initially, addresses were assigned using IPv4. However, as the number of devices connected to the internet increased, it became increasingly challenging to meet the growing demand. Due to its robust identity and location mechanism of 128 bits, IPv6 has become widely adopted. This method enables a wide range of unique addresses, ensuring effortless recognition and location of every item in the network. Ultimately, the process of identification plays a crucial role in the design of the Internet of Things as it provides each object with a unique identity within the network. While name and address are closely linked, they serve distinct purposes in the two levels of recognition. The success of IoT applications relies heavily on the network's capacity to efficiently communicate and transmit data, facilitated by the utilization of universal IDs and IP addresses.

b) Sensing Devices

This calls for the collection of environmental data and its delivery, remote or through cloud-based analyses, to local IoT sensor applications. Smart devices, portable sensors, or actuators could be recognizable. The information-filled storage media receives the information that has been obtained. Many detection tools, including data-gathering technologies, consist of Portable devices: Due to their accessibility and usability, portable devices like tablets, laptops, and smartphones are frequently employed as data collection tools.

I. Actuators: Actuators are devices that transform electrical information into actual actions, such as releasing a valve or switching on a motor. By keeping an eye on the motions or operations of equipment or gadgets, they may be utilized to collect data.

II Smart sensors: Pressure, temperature, humidity, and motion are just a few of the physical phenomena that these instruments track and measure. These were frequently utilized for data collection and tracking in industrial and Internet of Things systems.

III RFID labels: these are little, electrical gadgets which might be affixed to items and offer tracking and identification in real-time. They are frequently used for data collection and inventory tracking in logistics and supply chain management.

IV Wearable sensors: These tiny electronic gadgets may be placed on the body to collect information on physiological variables including blood pressure, heart rate, and activity levels. They are often utilized for data collection and monitoring for medical and fitness applications.

c) Instruments for Collaboration

IoT communication networks trade a variety of objects to deliver services that are intelligent. Another among the major components of the IoT is communication, which entails connecting and permitting interaction between various devices. At the network layer, devices may send and carry letters, documents, and other kinds of data. At the network layer, these devices are capable of moving and exchanging a broad variety of data kinds, including text-based data like messages, documents, and communications. They may transfer audio, video, and multimedia files as well. As IoT devices are utilized more often, an increasing amount of data is being transmitted over them. Effective and efficient communication protocols must be for managing this information flow. Communication is facilitated via a variety of technologies, including beta [42], near-field communication [55], Wi-Fi [25], RFID [1], and (LTE).

d) ComputerDevice

Utilizing devices, the information gathered by the gadgets is computed. It is used to supply the Internet of Things with capabilities. The Gadgetry, Arduino, and Raspberry Pi are examples of hardware platforms, however, software programs need a computer system to run. Android, Mini OS [35], Riot OS [14], Lite OS [16], and other operating systems represent just a few of the various platforms and operating systems in use today.

e) Semantics

To simplify user duties, the Internet of Things was created. It is a particularly significant part of the Internet of Things since it is accountable for

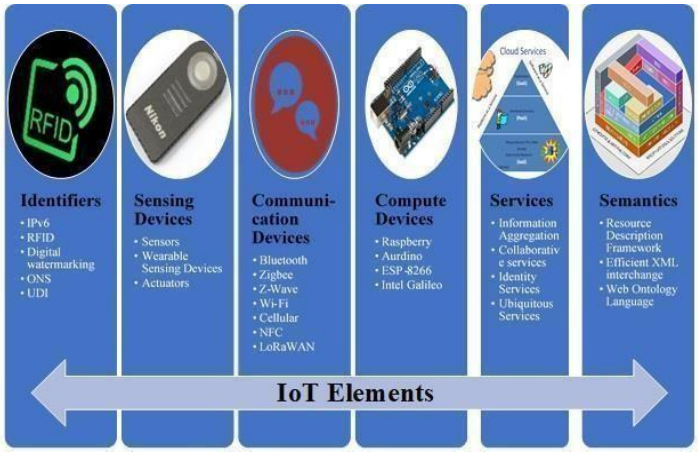


Figure.6.1:.IoT Eco system basics components

performing its responsibilities. It functions as the IoT's brain. It decides how to react after carefully assessing all available possibilities and sends signals to the machinery.

To study an IoT architecture, one might use perspectives that are things-focused, internet-oriented, or semantically oriented [12]. According to a perspective focused on things, NFC and RFID technology are used to connect intelligent, automated, and directed devices for a range of common purposes. The actual items or gadgets which make together the IoT system are the main emphasis of the things-oriented approach. This viewpoint takes into account the devices' capacities, restrictions, and connectivity possibilities as well as how they communicate with one another and the larger network.

This viewpoint aims to create an IoT system that is effective and saleable and can manage an enormous amount of gadgets as well as the flow of data. How these smart gadgets connect to the Internet and employ unique identities like Internet Protocol (IP) addresses is the subject of an Internet-eccentric point of view. and tried-and-true network technologies to facilitate global communication among these application-based linked phones. Distinctive identities, such as IP addresses, that enable devices to connect to the internet and interact with one another are routinely used in these exchanges.

Bluetooth, Wi-Fi, and cellular networks—all well-known network technologies—are employed to make it possible for these application-based linked devices to communicate across national boundaries. For instance, an iOS or Android app that connects to a smart thermostat through WiFi or cellular networks enables remote control of the thermostat is available. By enabling smooth communication and control among mobile devices and the web, the internet-eccentric strategy

Table 6.1: List the main method applied to every module and Important Technology Components for IoT

Element Internet of Things	Fundamental Systems
Classification	IPv4 and IPv6, ubiquitous coding (uCode), embedded device codes (eCodes), as well as,
Sensing	RFID tags are used as actuators, detectors, and monitoring tools.
Communications	Bluetooth, RFID, NFC,LT), and radio frequency identification (RFID) are examples of wireless technologies. WSN, or wireless sensor network. A group of geographically separated sensors working together to monitor environmental or physical factors is known as a wireless sensing network (WSN). Beta is a variation of Bluetooth that enables rapid data transmissionbetween devices. (LTE) is a wireless internet access protocol for mobiledevices and data terminals. Using electromagnetic fields,RFID automatically recognizes and tracks tags that are fastened to items. A wireless technology for communication called Near Field Communications (NFC) is utilized for close-quarters transactions like
	contact-less payments. The last norm for mobile internet access for mobilegadgets and data terminals is called LTE (Long Term Evolution).
Computation	Raspberry Pi, Arduino, and Intel's Galil operating system.
Services	a generally conscious knowledge aggregation linked to an ever-present identity
Semantics	OWL, RDF, and EXI

supports the development of a more connected and smart environment. A wide range of sectors, particularly health care, transportation, including energy, will be significantly impacted by this strategy. A semantic viewpoint on IoT architectures claims that relevant information is delivered utilizing data produced by IoT devices, thus resolving challenges with architectural modelling [40]. The semantic- oriented approach is focused on the organization, comprehension, and use of

the information produced by IoT applications and services. This viewpoint takes the data's semantics into account significance and potential applicability to other domains and ontologies. This point of view strives to make analytic s, Information exchange and interoperability across several applications and domains conceivable.

These three layers are referred to by a lot of academics as the network layer, applications layer, and presenting layer of typical IoT design. Additionally, as part of the most current IoT architecture, several scientists have investigated an intermediary support layer between the connectivity and application layers. Figure 6.2 represents depicts the multi-tier design of the IoT.The support layer is made out of technology found in the mist and clouds. This section describes the seven-tier construction, which includes processes for cooperation, applications for collecting data and accumulation, virtualization of network social media, a controller, and physical objects. components of each layer as well as the IoT's fundamentally covered nature.

i. Layer 1: Physical Equipment and controls

This layer of tools or cognition transmits information to a high layer from the outside environment. This layer includes actual sensors and parts. In summary, the duties of this layer include the identification of objects and the gathering of atmospheric data, such as pressure, temperature, air quality, water quality, detection of motion, and humidity, to mention a few. Controls and actual items can be used to operate several devices. These are the things that make use of IoT gadgets, which also comprise a range of peripheral communications transmission and receiving gadgets. There are several gadgets available right now. The Internet of Things (IoT) has virtually unlimited potential as more devices are connected to it.

ii..Layer 2: Connectivity

The IoT architecture's connection layer, which facilitates communicating with one another, at both higher levels of the system, is a crucial element. In order for different IoT components to communicate with one another, this layer offers the essential interconnecting systems, such as switches, gateways, and routers. It is crucial in the transmission of data from the sensor to the higher levels for processing. An important component of the connection layer is data transfer between physical links to the server or other computers. This is accomplished via several communication protocols including Zigbee, Z-Wave, SigFox, Bluetooth, or Zigbee. These protocols provide safe, timely, and reliable data delivery Zigbee is one of the most often used wireless communication protocols given that it requires little power and bandwidth, it is ideal for IoT applications in homes and buildings. One crucial element of the connection layer is the Edge Terminal device, which sits at the centre of the IoT system.

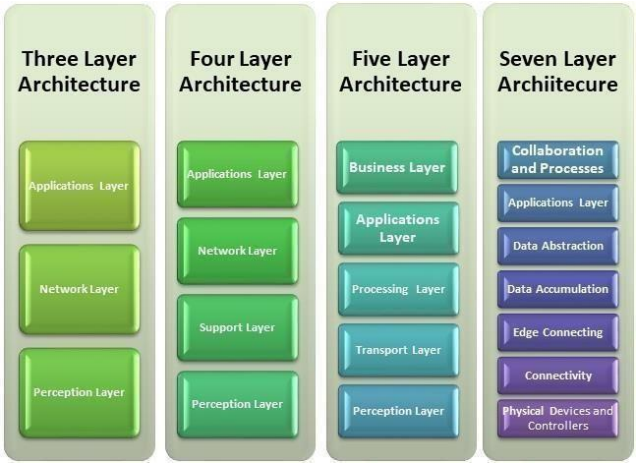


Figure 6.2: IoT Layers Architecture

By collecting, filtering, and processing data locally, serves as a link between real-world sensors and stored in the cloud transportation options. Less data must be transferred to the cloud as a consequence, which lowers latency and bandwidth requirements and boosts system efficacy and efficiency. The connection layer in an IoT architecture is primarily in charge of enabling communication and data transmission. It is essential to correctly design and implement this layer for the system to be dependable, efficient, and secure. It is crucial to select communication protocols and connecting systems carefully in order to guarantee seamless connectivity enable rapid data transfer between IoT components, make it possible for the Edge Node device to successfully function as a gateway.

iii. Layer 3: Edge Connecting

The following stage is cloud boundary, sometimes referred to as cloud computing. Data from the connection phase is required by Layer 3 for information archiving and analysis at a higher level. Since the modules in this layer handle a lot of data, the processing of some facts might be limited to minimal amounts of data. In the next stage of IoT architecture, known as edge connecting, data analysis and processing take place at the cloud-edge device interface. This stage, which allows for faster processing and lowers the quantity of data that needs to be transferred to the cloud, is essential for optimizing the effectiveness and performance of IoT devices. The most crucial data may be transmitted to the cloud for further analysis after being prepared and filtered at the edge to remove extraneous data. That method decreases the quantity of data that must be delivered while also lowering latency and speeding up system responses. Sensors, which are motors, gateways and various other equipment that can gather and analyse data in real time are frequently included in edge computing modules. Overall, edge connection is crucial to the achievement of IoT systems because it makes data processing and decision-making more effective and efficient.

iv. Layer 4: Data Accumulating

Once collected, the data has to be retained. The location of the data storage facility is important. Although some data may be kept permanently, the majority has to be transmitted to the server so that big data technologies may analyse it using their computing power. The preservation of Phase 3 data is the main objective of this layer. Collect, store, and place it in the warehouse in order to make a lot of data accessible to the top levels. It essentially modifies event-based data to give data for higher-layer query-based processing. This layer may be developed in SQL and supports more advanced NoSQL databases like Mongo, Spark, Hadoop file systems, Cassidy, and others.

V. Layer 5: Extracting data

This layer efficiently and quickly synthesizes data from many sources and creates a suitable application format from the stored data [11]. These sources might consist of sensors on RFID tags, electronics, and other linked things that produce data in various formats and at various speeds. The layer that aggregates data must collect and prepare this data in order for higher layers to easily examine and understand it. Duplicates and mistakes must also be eliminated. This layer makes data analysis more precise and thorough by fusing data from several sources, resulting in greater insights and decision-making. IoT systems must be able to gather, store, and analyse data from many sources effectively and efficiently, and the data aggregating layer is crucial to this. A publishing computer system or information dispersion service, which makes data transfer easier than using cloud technology, information aggregation, processes, and user layers, is a key part of the enormous high-implementation architecture (DDS).

Regardless of whether an application employs a high-performance solution or a fast message bus, this design makes setup easier and speeds up all applications— aside from the most basic ones. Applications with less complicated data processing and transport needs are referred to as being most straightforward. These applications could convey data between network nodes using a straightforward message bus. Excellent performance messages or distributed data aggregation techniques, on the other hand, may be used by applications with exceptional performance that need quick and effective data processing and transport. Such technology enables applications to handle enormous amounts of data rapidly and efficiently, enabling them to fulfil their demands for effectiveness. In the context of the Internet of Things, publishing computer infrastructure or knowledge dispersion services provide a platform that simplifies placement and boosts data transfer speed and efficiency, regardless of how complex the application is. This makes it easier to develop and manage dependable apps.

vi . Layer 6: The Application Layer

Data analysis from various IoT applications is involved. Numerous Internet of Things Applications like linked cars, smart cities, intelligent systems, intelligent buildings, intelligent agriculture, and others are among those that are covered [51]. It should go without saying that at this time, the software will activate the control programme and the data package. Process optimization, scheduling, statistical control signal analysis, vigilant surveillance, alert leadership, and consumption models are only a few examples of IoT applications.

vii . Layer 7: Collaboration and Processes

People who can effectively cooperate and communicate while using IoT data are classified by its layer. additionally to further application layers using database recovery methods. Managers might likewise make data-driven business decisions using it [44].

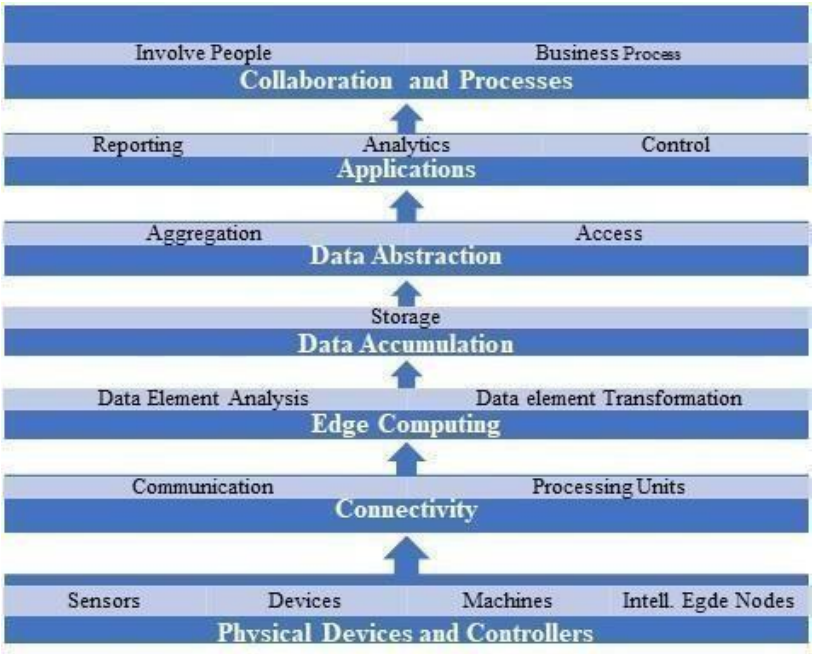


Figure 6.3:IoT's Basic Structure

6.2 Cloud -IoT Enabling Technologies

According to the National Bureau of Standards (NIST), end users can access a sizable, suitable, and pay-per-use pool of amenities and computing resources (applications, storage, systems, and services) through the usage of cloud technology as a framework. with the least amount of effort or interaction with vendors of services. [43]. Cloud computing is becoming swiftly adopted for project execution by IT corporations and experts due to its low cost and rapid flexibility [53]. The fundamental elements of the cloud as a Web technology are on-demand awareness, ubiquitous internet connection, demanding capacity, fast adaptability, and quantified services [59]. By

leveraging the frameworks, tools, and software development processes made accessible by the cloud environment, Platform as a Service (PaaS) enables cloud customers to build their products on the architecture of the provider. To execute any software or operating system installed on a virtual machine (VM) made available by the web resource provider, for instance, users can install a variety of components, such as networks, processing power, memory, and other essential computing needs [54].

This is possible because of the Infrastructure as a Service (IaaS) paradigm. With this approach, clients have the freedom to modify their computer infrastructure as needed to suit their unique application requirements. In order to execute any kind of operating system or use, users may, for instance, deploy virtual machines (VMs), while they are able to ramp either increasing or decreasing the amount of computing power they have in response to shifting workload needs. Clients routinely build virtual networks, firewalls, and other safety measures by altering the network and security settings of their infrastructure to safeguard their apps and data.The IaaS model, which lets users design and administer their IT systems without having to acquire and maintain physical hardware, gives users a lot of power and freedom. Cloud computing also offers four more cloud deployment strategies: hybrid, public, private, and community, depending on user preferences and functionality. [52].

The technology of the cloud has been used in the development of technologies like the Internet of Things due to its various features and possibilities. The Internet and cloud computing integration have been extensively used in a variety of real-world applications, including smart cities, healthcare, agricultural industry mobility, and enhanced cars, among many others [38]. One of the most recent technical advancements is the usage of smart devices that are internationally connected to the internet. These devices produce a huge volume of material, which cannot be locally stored owing to a lack of storage space. mobile devices, pills, smartwatches, fitness monitors, and any other internet-connected devices are among the gadgets that produce a tonne of material. These gadgets gather information and data from numerous sensors, including heart rate monitors, accelerometer, and GPS. Users may also create content using them, including texts, social network postings, images, and videos. As a result of their ability to capture and share data in real time, these gadgets produce a vast amount of material. Through a number of digital venues, users may quickly document and share their experiences, opinions, and actions with others. Furthermore, the popularity of social media, internet streaming, and cloud storage has increased, which has facilitated the spread of the information produced by these technologies. The need to efficiently exploit and manage such a massive volume of material offers businesses and organizations both possibilities and problems. All sectors depend on data, but without data analysis, data is useless for gathering the necessary knowledge for the future. Sophisticated

computer systems that can handle such massive quantities of data processing are required because smart, linked gadgets acquire vast volumes of data. The capability of the separate systems is insufficient to analyse the data. These limitations on enormous memory capacity, processing power, and connection speed may be overcome using the communal resources made available by cloud technologies.

Cloud computing allows IoT to function without any issues, as seen in Fig 4. Among apps and IoT devices, cloud-based storage infrastructure is employed as a covert layer that hides every step of the development of IoT programmed [24]. So order to offer a centralized and saleable storage solution for the enormous volume of data produced via various sensors, cloud-based storage infrastructures are utilized as a covert layer between applications with IoT devices. IoT devices may transfer data to the cloud via cloud storage, so it can be saved or utilized by apps and services. IoT devices may be small and energy-efficient as a result of this strategy since they won't need to locally process or store a lot of data. Furthermore, additional features like data analytic s, data visualization, and data mining may be included in cloud storage infrastructures. These features may be used to draw important conclusions from IoT data. In addition to enabling a variety of deployment patterns, the usage of online storage infrastructures also makes it simple for organizations to add and delete applications and devices from their networks as required.

6.3 IoT Resource Management

The Quality of Service (QoS) criteria are reached with the help of a highly effective, economical, and well-maintained network. The system is linked to a number of resources using IoT architecture. Since resources are allocated wisely and effectively via the IoT network, this is a crucial component of QoS requirements. Due to the fact that Because the data in IoT architecture is divided between multiple information sources and is gathered from multiple sensors and networked devices offer a variety of services, resource allocation is also in charge of upholding tight safety requirements. The IoT makes use of energy, storage, and computational resources that are networked. By efficiently assigning cloud resources to them, IoT-connected devices can improve system productivity and efficiency. In light of IoT resources and objects Edge computing, cloud infrastructure, and IoT items make up the three tiers of the IoT system. Prior to picking the nodes, the system had to first decide which resources from these three levels it would need. The IoT ecosystem is heterogeneous and globally scattered, and it is made up of essential components that require management and resource allocation. The point of connection between An entity, which might be a thing, a person, or a location, is what the system and the IoT network app user are. Resources are these given things, and depending on the information they communicate throughout a network, they can be divided into a number of groupings. Last but not

least, it was necessary to prepare for the user's work to be fulfilled on the chosen node, as illustrated The user's task is lastly successfully completed by contact between the two network servers. Resource supply, monitoring, identification, and scheduling are only a few of the numerous components that make up resource distribution. The necessary Service Quality (QoS), effectiveness improvements, decreases in power usage, and resources management enhancements Not to mention, it needed to arrange for the selected node to complete the user's task. Finally, the user's task may be successfully completed thanks to communication between the two network servers. Resource distribution includes a wide range of elements, including resource delivery, measurement, classification, and scheduling.

The required Service Quality (QoS), efficiency improvements, decreased power usage, enhanced resource management utilization, and, Since it must perfectly meet their demands, the support level agreement between cloud-based IoT device software suppliers and their clients is of the highest significance. The service quality (QoS) offered to For cloud-based IoT solutions to be successful, end users are essential. For higher QoS, it is essential to increase efficiency, consume less energy, and optimize resource consumption. Since it establishes the amount of support and maintenance provided to end users, the support level agreement among st cloud-based IoT device software suppliers is the most crucial component. Applications for IoT perform better and incur significant cost savings as a result of successful vendor collaboration and resource use. As a result, it's crucial to take into account each of these aspects while developing cloud-based solutions for the Internet of Things.

The allocation of resources is complicated by the variety and distance of IoT devices. Despite substantial research by scientific and industrial experts, there are still many challenges and problems related to the distribution of IoT resources. Numerous scholars have created original strategies and methods to deal with these issues. In this article,The author examines several resource management techniques developed for the cloud-based IoT environment and groups them into different categories based on the characteristics they have in common and the resource management standards they have raised. Other IoT characteristics that are connected to resource allocation were also covered by the author, with a focus on those that still require improvement. Three-tiered architectures (IoT, Cloud, and Edge) .

Designing IoT systems, using wireless sensor networks, and integrating IoT with cloud computing surveys and evaluations of IoT technologies are among the study topics that have been examined. However, the distribution of resources in an IoT system is only covered in one survey report. Multiple aspects of the Internet of Things (IoT) design are the focus of several researchers. These are the tactics and developments that make it possible for IoT-networked systems to

function effectively and efficiently. There isn't a review available for approaches to cloud-based IoT resource allocation. A thorough overview of various challenges and issues with assigning IoT assets from an architectural perspective was offered by the writers in [37]. The author of this review research looked at the designs and tools to find out the allocation of resources in an IoT ecosystem.

The present research does not look into the methods for planning and maximizing resources from a cloud standpoint. [22] Is it a review of research that looks at difficulties imposed by different IoT devices? In addition to discussing a variety of resources The author of this article looks into the connections between fog computing and IoT using allocation mechanisms now in use in environments where fog-based IoT is common. .The author also noted two other important problems, To begin with, edge computing merely considers the network connectivity with the lowest rate of latency in an Internet of Things (IoT) architecture based on fog, and it pays no attention to the network with the highest processing power or computational capabilities. The distribution of finances among fog computing and IoT smart devices is the second problem the author brought up. This is a result of the limited capability for resources in fog computing. Compared to fog computing, the writer recommends cloud computing as a solution to the problem of limited availability of fog resources. The author of this study used the internet to investigate several IoT resource management techniques and categories them. Also reviewed and presented in tabular format are the additional limits and improvements of those methods. There have been discussions and specifications for several resource allocation variables. The volume of work still needed to complete the other requirements is also discussed in this article, along with the progress made in these areas.

6.4 Fog IoT Resource Planning Categories

methods Depending on the characteristics of each, the author analyzed and classified the IoT resource provisioning approaches used in this industry approach that were presented throughout the course of the inquiry, illustrates the accurate classification of IoT resource allocation

6 4.1 IoT Resource Allocation with SLA Awareness

Every service-oriented infrastructure includes the service level agreement. In this case, communication between consumers and suppliers is crucial. In order for the service provider to maximize profits while still meeting consumer needs, the SLA should be broken as little as feasible. In this field of study, a variety of research techniques have been employed to decrease SLA violations and raise customer system acceptance A service level agreement (SLA) is a contract that specifies the quality of service that will be offered and the fines or other repercussions that will apply if the service is subpar. To make sure that the service provider meets

the customer's expectations, SLAs frequently contain measures like reliability, reaction time, and resolution time. SLA-focused resource allocation is the subject of more study.

Figure.6.4: ClassificationofIoTAllocationOfresourcesTechniques

By estimating the customer benefit and designating A computationally based auction technique is proposed to lessen the penalty for SLA violations, acting as a champion to the customer who provides the most profitable source. This strategy can reduce the fee amount in an attempt to deter SLA infractions. Article 49 offers an alternative way based on SLA violations. To minimize SLA violations, the author developed a method for effectively scheduling and limiting the user's task. By dividing user labour into several smaller jobs and increasing the server's capacity, the proposed method lowers the overall job execution measurement time, prevents SLA breaches, and increases vendor revenue. [8] outlines a method for supplying electricity to Internet of Things devices while keeping service level agreements in mind. The author proposed an architecture for an Internet of Things ecosystem that assigns resources and services using fog processing and cloud computing. User activities are managed via a linearizable tree of options using an algorithm that gives priority to SLA and QoS. A novel IoT resource allocation problem with an emphasis on SLA was addressed, per [50], by capping the fastest rate at which users may do tasks and buffering scheduling. As it anticipates the user's job delivery pattern and accommodates SLA violations, the suggested approach works better in IoT networks with high

Table 6.2: IoT device traffic list

Algorithm	**Improvement**	**Limitations**
Resource Allocation Based on SLAs [18]	Improve system efficiency and minimize SLA breaches.	The recommended strategy is not contrasted with alternative approaches.
SLA-aware Cloud-based Resource Allowance Control [49]	The user's job is divided up into smaller pieces in order to reduce SLA breaches.	The employment waiting period is not dynamically determined..
IoT resources and services delegation [8]	Boost the efficiency and performance of the system.	There is no evidence that it has been used in practise.
Allocating resources for IoT. [50].	You may improve system performance and reduce SLA violations by lowering the task markup.	In a multi-tenant environment, such as numerous data centers, the recommended solution is ineffective.

6.4.2 Allocating IoT Resources with Context

Numerous studies have used game mechanics to allocate resources in device-to-device interactions with IoT devices. The administration of resources in a wireless network is essential for efficient data transfer for communications between devices.suggested an evolutionary stable mechanism for location-aware D2D communication in a cell connection. The author suggested a particular context-aware method that calculates the network's capacity to optimize the total quantity of communication that each station may conduct. [27] identifies the cell affiliation issue as being caused by the two independent matching operations of communication equipment.

The proposed model's device correlation improved the cell association strategy and guaranteed a consistent result for each local area connection. The association of one or more devices constituted the basis for another efficient resource-use strategy. and enhanced network device resource use in [2]. It has been demonstrated that the proposed context-aware resource allocation strategy across IoT network systems satisfies dynamic QoS requirements. Table 3 lists the benefits and drawbacks of context-aware resource allocation techniques.

Table 6.3. IoT Resource Allocation Techniques That Take Context

Algorithm	Improvements	Limitations
With a cloud-based IoT system, the game theory is used for communicating via D2D [29].	The evolutionary stable game model and optimizing bandwidth use improve D2D interaction.	The sole topic of focus for this strategy is the cloud, and it contrasts with current algorithms.
An actionable source according to correlations is used [27].	To improve the efficiency of the cell attachment method, redundant data produced by the various sensors have been removed.	This approach contains not one mention of the empirical data.
Technique for Effective Resource Allocation [2].	Utilization of resources and efficiency have both increased.	The method is not supported by any tangible evidence.

6.4.3 Allocating IoT Resources with QoS Awareness

Quality of service (QoS) considerations must be made in the service-related solution. Our service standard acceptance with superior service (SLA) must align. The term Quality of Service (QoS) describes the level of operation as well as dependability that a connection or service offers to its users. In order to make sure that the connection or operation fulfils each customer's expectations, QoS is often monitored using parameters like efficiency, delays, along damage to packets. Flow sculpting, prioritization, and handling congestion are a few of the strategies that may be used to achieve QoS, and it is crucial for making sure that vital apps and websites have the space and connectivity they require to operate effectively. Numerous research on this issue have been conducted in diverse settings.

The publication of a QoS-based technique for allocating IoT resources in [17] made interactions between devices less obtrusive. PFR with an integrated incursion-restricted area control method can be used to limit the utilization of facilities to dual-layer clients. The Bluetooth connectivity method combines effort that boosts system efficacy while controlling device-to-device clients' utilization of applications. A computational model to represent the multi modal population of multimedia to multimedia devices is proposed in [48] to enable to management of QoS constraints. For mixed Internet of Things (IoT) systems, an optimized protocol based on the capacity supply technique was recently presented in [19]. The programme considered the

operation rate of the networking hubs on filtering depletion. The tactic is flexible enough to accommodate the ever-changing and diverse properties of connected IoT gadgets. An asset deployment strategy based on interactions and the Net of Devices was described in [20]. Between the devices, these tactics utilize discussion and streaming technologies to facilitate communication and show updates. The following situations were considered while evaluating the recommended method: the entire framework, specific tasks at a particular rate, and a particular job at a particular frequency. The system's manufacturing mistakes were reduced to a fifth relative to the standard technique, which will reduce communication delays and increase system reliability. The advantages and disadvantages of QoS-based utilization strategies are shown in Table 6. 4 Below .

Table 6.4: Demonstrates Wifi utilization techniques using Quality understanding.

Method	**Developments**	**Drawbacks**
Method for Allocating Downlink Resources [17]	The efficiency of the systems has increased, and interference between the communication channels has decreased.	There is applicability. A lack of practical
Scheme for Allocating Radio Resources [48]	There is less communication going on between M2M sensors.	The method is not applied in real-world circumstances.
Work Distribution Among an IoT Device Group [19]	The optimal resource allocation has been achieved with a 5% error..	The QoS component must receive a lot of attention.
IoT work scheduling based on consensus [20]	Although both chatter and broadcast approaches are used, broadcast yields the greatest results.	Less consideration is given to real-world circumstances and QoS.

6.4.4 IoT Utilization Of resources with Energy Awareness

Whenever it's vastly varied to voltage-hungry technology is used in massive quantities, it is essential to monitor how well one makes use of fossil fuels or electricity to try to reduce environmental damage and establish a sustainable calculation group. Numerous researchers have been striving hard to reduce electricity consumption under a number of domains, including data

centers, data centers, and working amid the mist. In connected device situations, the reduction in electrical consumption represents a few strides. The heat loss related to the virtual storm of anyone (V-FOE) and its information technology foggy of nearly anyone choice method is detailed in [13] (FoE). With the objective to effectively improve QoS by conserving electricity, a trio of methods of device connection was created in the cloud-based computing technology (FOCAN) [45]. This technique is used in the design of intelligent neighbourhood utilizing mist help to distribute building duties across computerized sensors.

The generic It (GNE) technique along with its specific parameters are derived by [3] with the objective of handling the variation in Internet-of-things components about Quality and bandwidth constraints. By using a cognition-tiered model of games in such a way, the gadgets manage to attain Chf stability (CHE), which logically correlates with the diverse processing powers in addition to the knowledge availability of every Mtc and HTD. Our recommended approach reduces MTD consumption of electricity by 78%, which is The ECIoT architecture was introduced in [36] because an exciting novel method for improving operational efficiency by controlling process admittance and use of resources within the networked Internet of Things (IoT). ECIoT uses a cross-layer fluid-structure optimization strategy that uses Lyapunov optimizing settings to increase the device's usefulness. Shown in Table 6.5 below displays deficiencies that improvements achieved by the recommended techniques.

Table 6. 5.: Energy-Conscious Allocating IoT Resources Methods

Algorithm	**Improvements**	**Limitations**
Where in Fog of Everything the energy-efficient resources are situated	lowering energy use, postponing, and enhancing FOE system effectiveness	No real-time monitoring has occurred..
FOCAN [45] is a smart urban design for resource allocation.	The method reduces electricity use while extending delay.	No one has been put to death.
Take into account the perceptual hierarchy theory while allocating resources. [3]	The procedure reduces power use by 78%.	There has only been one simulation run.

Resource distribution in conjunction with joint check pointing [36]	Enhanced system efficiency and decreased communication lag	There is no evidence that it has really been put to use.

6.5 Cost-Aware IoT Resource Allocation

The data centre, routers, and mist platforms are used to handle the diverse, highly costly gadgets that make up the Web of Everything (IoT) connection. The various network-related equipment require funds in order to carry out what they do that adheres to QoS rules. The activation costs for every connecting gateway equipment along with each service offered were utilized to calculate the overall expense. The above cost estimate problem is known to technicians as the service-to-interface distribution expense dilemma. For reducing computing costs, several SLA approaches are supposedly given in [9]. The above resource's requirements were met by one method in just one session but by the subsequent approach across a period of multiple games. The suggested cure divides the price of activating and distributes it over several locations in order to reduce registration costs. Reduced expenditures on services with the cloud need deliberate efficient capacity delivery and management.

This multi-agent-based cloud assignment mechanism is being built to track how the Web of Devices uses resources [39]. The inspection of asset usage helps to improve how well resources are utilized and also reduces excessive allocation of resources, this improves speed and decreases operational expenses generally. A distinct method [33] built off of the Stack gameplay theory was developed to lower the expense of distributed services. The method examines the bottom and top levels of the system and validates the Nash balance location of the power source no-cooperative activity that takes place among st each other in the bid to reduce fees. A process of iteration is used to get the It by making the Stackelberg play across all of them. A variety of offerings including incompatible links are being used by connected gadgets. [10] develops the specification-based mixed-integer processing (MILP) tool to support a diverse services paradigm. Easily distributing the services across the many ports may lower the cost of those amenities. The establishment of cost-effective utilization strategies for IoT systems is covered shown in Table 6.6 below along with their limitations.

Table 6.6 : IoT Allocation Of resources Methods That Consider Cost

Methods	Developments	Drawbacks
Using a variety of resources to provide flexible services[9]	The cost has Lowered by dividing the items among the terminals.	Utilizing the multi-round strategy now comes at a little higher cost.
Optimizing Resource Allocation in Cloud Architecture [39]	Utilize VMs as little as feasible to increase system effectiveness and reduce overall expenses.	Costs associated with resource utilization must be decreased..
Heterogeneous-oriented Re- sources are recognized [33]	Costs and resource use have dropped.	There is no evidence that the implementation is actually useful.
Use of Heterogeneous Assets with Flexible Allocation [10]	Both costs and system performance have lowered.	The subject has not been used in practise and is primarily theoretical.

6.6 IoT resource planning strategies consider Resource Allocation Criteria

Ecosystems built on fog IOT technologies depend on the allocation of resources. The connected devices in the IoT context produce massive In order to give data that is useful for the system's intended uses and architecture, a lot of information is stored online for subsequent study. Numerous resource allocation factors were analyzed in [6, 7]. The author outlines the different allocations of resource criteria for an IoT environment in the next section. These elements must be taken into consideration while creating IoT allocations of resource algorithms.

a) It is required on the part forming the WiFi layers to finish the assignment assigned by the user. Maximum system capacity is required.Productivity of a connected object originates by speed, or the aggregate number of operations or duties that the device has performed. IoT-related will require a rapid speed so as to fulfil each of its consumer activities.

b) QoS, or the excellence of services, is a metric used to assess the quality of offerings as the cloud infrastructure would supply consumers as per Contractual consent.

c) As the equipment does other operations, extend the amount of time that's necessary to deal with an individual's waiting assignment. This channel's delay ought to be minimized as much as possible to improve system operation.

d) It seems the speed at which data from Network of Things (IoT) items is sent from one location to another. The speed of data exchange will need to be rapid in this setting of the Internet of Things.

e) Reliance is knowing how to perform the task at hand on time despite getting harmed by an infrastructure failure. Its reliability has proven outstanding to ensure consumers finish their activities promptly.

f) SLA Upholding the agreement for services with the good buyer to the business offering them is crucial to prevent excess costs and reduce the likelihood of user-supplier contract breaches.

g) Time Smartphones and tablets a big part in the Internet of Things (IoT) world owing to the enormous volume of data they frequently create. Planning out how to complete something in an Internet of Things (IoT) setting is a tactic.

h) Affordable Internet of Things (IoT) systems pay a vendor's cash send to utilize plenty filled with cloud service offerings. The quantity and efficacy of the process are utilized to estimate service costs. The system needs to be as affordable as feasible.

i) Affordable Internet of Things (IoT) systems pay a vendor's cash send to utilize plenty filled with cloud service offerings. The quantity and efficacy of the process are utilized to estimate service costs. The system needs to be as affordable as feasible.

j) Accessibility Availability involves an evaluation of an electronic resource's reliability and quality availability in an established window. That needs to be plenty of help ready to reduce operational disruptions.

k) Usage: Supply exploitation represents the gauge of how much of a given asset the entire structure may use. The funds can be utilized with as little waste in order to maximize its e efficacy and overall robustness.

6.7 Improvement have been done in these parameters

During this study, an investigator investigated multiple online IoT ways to allocate resources and divided the results into various categories. The variables were indicated to be consistent with the process or procedure under consideration in this subsection, which looked at the aspects of distributing resources which are enhanced by the strategies mentioned in the subcategory. Depending on the professional sector, readers are able to observe what variables are contributing significantly as well as those that demand more research.

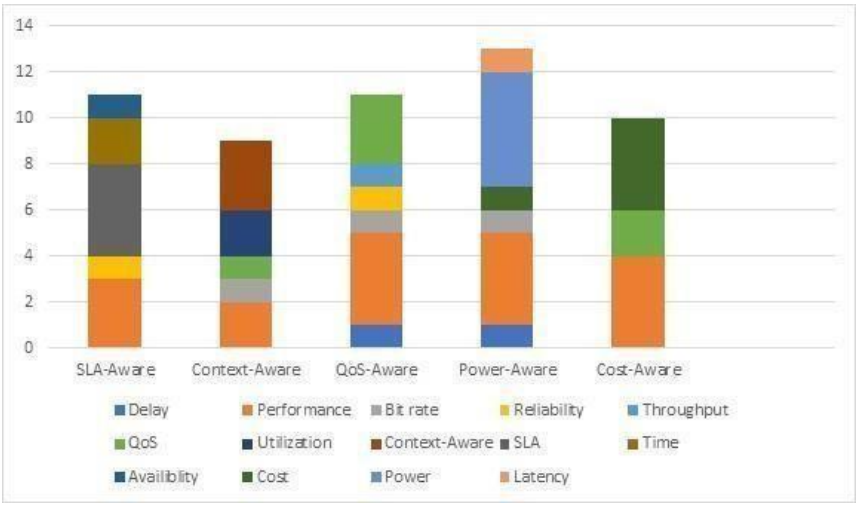


Figure 6.5: Modification Level of Variables for Resource Management

Above Figure depicts IoT devices feeding data to edge gateways for preprocessing. A load balancer dynamically routes data to private or public clouds based on sensitivity and load, with continuous monitoring adjusting variables to optimize resource management and performance.

The procedures and ideas provided improved the number of relevant factors of distributing assets, as was mentioned in the article ahead. However there remain many aspects of assignment indications that can be increased, so scholars have to remain particularly aware of these. The graph unequivocally demonstrates that many investigators have raised the risk barrier. The delays, put off, the overall supply of the connection broadcaster, nevertheless do not vary much. It additionally is likely that scientists to propose extra allocation strategies, which might eventually improve those commodities' supply standards. Numerous impartial and typical techniques are available to study.

6.8 Problems and Obstacles

The World Wide Web with Things Complex provides exciting possibilities that might boost its efficiency. Despite its many advantages, the Internet of Things (IoT) with the dissemination of online materials faces a variety of challenges. The remainder of this article contains multiple research studies on how IoT finances are allocated. Regarding IoT systems having been properly integrated, more research must be carried out on the brink, online, etc fog-like collaboration. A couple of components of assigning resources provide the foundation for the bulk of articles. This does not suggest that the proposed approach is being employed in truly Internet embarking because most of the studies have not yet been successful towards the model. If presently there is no significant actual investigation in a real-world IoT context, there can be problems once these approaches are used inside the setting of IoT. The network of IoT devices itself possesses a number of problems as peculiarities. It also means the approaches for distributing asset examined

in the present inquiry fail to account for every facet of material dispersion, such asasset inspection, asset verification, asset setting up, substance modelling, assets forecasting, and organizing resources, which is another problem. Furthermore, it is how analysis frequently focuses on distributing resources tactics rather than capacity deployment optimization, making it vital when creating a prosperous efficient Connected economy.

6.9 To assess the effectiveness of the data skew load balancing optimization system under various conditions, such as optimization rate.

Shay Vargaftik et al. introduce the family of load distribution methods known as Locally Lowest Queuing (LSQ) [65]. In such methods, each controller maintains its own, perhaps outdated version of the machine's waiting time while continuing to employ JSQ for its own purposes. This local view is refreshed periodically with minimal connectivity latency. The network's architecture is extraordinarily resilient, as we have shown, providing that all these different local estimations of the product's waiting period are accurate. Last but not least, we show in simulations that, given an equivalent communication allowance, simple and trustworthy LSQ policies perform brilliantly and outperform existing low-communication techniques.

Ali Yekkehkhany and Rakesh Nagi [66] provide the GB-PANDAS strategy, which would not be aware of task patterns of input or delivery rates in any way. An exploration-exploitation strategy is used by the Blind GB-PANDAS routing algorithm. We demonstrate that the best output is provided by blind GB-PANDAS.

When the service duration of various job types is dispersed randomly and in an unknowable way among a large number of servers. Blind GB-PANDAS intends to transfer incoming tasks to the server with the fewest balanced workloads, however, this is impossible due to fluctuating service fees. As a result, carrying out the throughput optimization evaluation is more challenging than it ought to be in the case when response times are known. Our extensive research reveals that as far as the typical time required to execute tasks under Blind GB-PANDAS significantly outperforms earlier approaches under large workloads.

The development and upkeep of Big Data Streams Publish Subscription (BAD Pub/Sub) platforms present problems, according to Hang Nguyen et al. [67]. in order to assist the creation of enhanced upgrades that are saleable to other social levels. Plans for the BAD Pub/Sub system's objective is to take in massive amounts of info from multiple writers and organizations and to offer improved, tailored notifications sent via customized ways to end customers (subscribers) who indicate interest in such information pieces.

Doaa Medhat et al.'s [68] flexible and economical Map Reduce work scheduling solution can reliably link data from several foreign sites. They do this by distributing multilingual duties

across affordable and accessible machines. Our solution outperformed past approaches contrasted with novel blocking-based load distribution methods in terms of efficiency, velocity, and storage space on a cluster of Hadoop nodes in a public cloud infrastructure.

Rohail Gulbaz et al. [69] provide (BGA), a novel load-balancing scheduler, to enhance span and load balancing. Inadequate load balancing may cause a resource utilization overhead since some of those resources are idle. With millions of directives sent to VMs, the true weight of load balancing is taken into account by BGA. Multi-objective optimization should be utilized to improve job scheduling and load balancing, it is also emphasised. Skewed, typical, and uniform work distributions, as well as different batch sizes, are all used in research. BGA has significantly outperformed a number of state-of-the-art methods for load balancing, make span, and throughput.

6.10 To evaluate the effectiveness of the information skew load balance optimization systems under various conditions, such as information skew rate.

A pattern has emerged. regarding consolidated control of IoT services (data and cloud-based apps) since the conception of the World Wide Web of Things (IoT) concept, which allows the oversight and control of a sizable number of internet-connected devices. Through IoT management services, a collection of senses may be efficiently managed. IoT gateways are responsible for reducing The incidence of IoT gateway-related internet connection overpowers or disruptions. In order to address these problems, we employ a fog-of-everything load-balancing technique. In this part, we present our s, achievements that may be deployed and put into practice, and an assessment of the results [70].

The major objective of a load-balanced cluster topology based on simulated annealing is to increase the network's lifespan while maintaining enough sensing coverage under periodic or irregular data transfers from sensor nodes. We show that the suggested technique is capable of boosting both the diversity of communication platforms and the performance of the most widely used state-of-the-art clustering algorithms by comparing them to these algorithms through simulation experiments. Network coverage may be improved by maintaining more network devices online for extended periods of time at a low computational cost. 71].

Resource-limited networks have a wide range of applications in daily life. A challenging issue is finding a reliable load-balancing method to extend the life of these networks. The approach accounts for variables including distance, remaining electricity, and number of hops to balance energy consumption across network users and extend network lifetime. In simulations, our proposed system beats the alternatives in terms of productivity, node longevity, lost packet rate (PLR), cost of communication, latency, and computation expenses. Our recommended approach also increases the lifespan of WSNs and shields individual nodes from the operational

The wireless sensor network (WSN) is a type of self-sustaining system with limited power and transmission sources. Using a strong heuristic update strategy based on a greedy projected energy cost metric, the route establishment is optimized. In order to reduce the power consumption brought on by the control overhead, EBAR employs a power-opportunistic broadcasting technique. In light of the results of this thorough comparison of EBAR to the most advanced techniques, such as EEABR, Sense Ant, and IACO, is a significant improvement [73].

RPL considerably satisfies IoT network requirements, but as it wasn't designed with IoT devices in mind, there remain some unsolved problems. The CAOF, which determines the rating by taking the node's contextual information context into consideration, is described first. CAOF additionally avoids the thunderous herd effect by gradually shifting away from an extremely positioned number and towards the real rank amount. Second, we provide the context-aware route metric (CARF), a novel routing measure that iteratively assesses the power consumption and queue of parental chains as they get closer to the root while minimizing the impact of parents who are farther away. Packet loss has lowered and network longevity has risen, according to evaluation results equivalent to the RPL standardized specification. [74].

6.11 To calculate the data skew load balancing optimization system efficiency under various conditions.

Xiaoke Zhu et al. [75] created DLB, a load balance solution To properly solve the data skew issue, deep learning is used. The fundamental concept behind DLB is to swap out hashing for neural network models in task scheduling methods so that different jobs and data distributions among servers may be balanced equitably. Results of experiments with both manufactured and real data sets suggest that our cloud-based DLB, when implemented in a real-world cloud setup, has the potential to provide better balance mappings than conventional approaches to task scheduling according to the hashing process function, particularly when workloads are significantly skewed.

The work scheduling issue is with the aim of improving the use of time and energy as two QoS parameters in a fog situation, as tackled by Zahra Movahedi et al. [76]. We first present a fog-based framework to manage scheduling algorithm needs and arrive at the optimal solutions. Second, research develops an integer linear programming (ILP) optimal model that accounts for the fog's energy consumption as well as the job scheduling problem's time expiration. Finally, but just as importantly, we offer a more effective course of action to increase the of the original WOA in resolving the issue of artificial work scheduling. The opposition-based chaotic whale optimizing algorithm (Oppo CWOA) is the name of this innovative approach.

Comprehensive simulations and comparisons with the original WOA and a number of well-known meta-heuristic algorithms, including Particle Swarm Optimization, Artificial Bee Colony, and Genetic Algorithm, are used to show the usefulness of the proposed alternative WOA (GA).

The effectiveness of the computer's clustering is significantly impacted by this work scheduling technique. This illustrates various load-balancing approaches and focuses on their merits and limitations. Lijie Yan and Xudong Liuudy also provide a load-balancing strategy that highlights the merits and drawbacks of well-known load-balancing methods while illustrating them. In addition, Lijie Yan and Xudong Liu present a balancing technique they developed utilizing the load forecast. The fluid exponentially smoothed model's output is the value at the next instant by examining the server node's load time series to establish the appropriate smoothing coefficients for the current terms and realistic data to aid forecasting by utilizing the node's load. [77].

Bat behaviour employs echolocation to find prey for the creation of concepts of load balancing. Virtual computers (VMs) that are utilized to carry out the transferring procedure for the bats' essential data are categorized using the Naive-Bayes method. Due to their lower importance in the allocation of the line, the migrating jobs switched from heavily loaded running virtual machines to weakly loaded visualized computer counterparts. The most important tasks should have precedence over any additional services in the VM and shouldn't be assigned to the same VM as workloads transition from fully loaded virtual machines to lighter-laden virtualization instances. The experimental outcomes of the loading re-balancing algorithms (LBBA) were contrasted with those of traditional methods such as a round robin (RR) and dynamic allocation of loads (DLB) [78]. the management and structuring of the massive amount of information that these systems create is the fundamental issue with the Internet of Things (home automation systems. Privacy is a significant issue as well since homeowners' sensitive data may be exposed to online attacks. Additionally, it might be difficult to integrate various IoT systems and devices, which could cause interoperability problems.

Cloud-based services that are capable of storing and handling the data derived from the Internet of Things automation devices in homes are the answer suggested by 79 Sangaraju et al. These on-the-cloud options give users centralized platforms for data storage and analysis, enhance data security, and make device interaction simpler. Internet of Things (I home automation systems may benefit from cloud-based solutions in a number of ways, including enhanced data security, centralised data administration, and the capacity to do performance analysis on collected data. Additionally, these solutions ease the process for homeowners by making it simpler to combine various gadgets and systems. Overall, solutions that are cloud-based can aid in resolving issues brought on by being connected to the internet of systems for home automation and enhancing

According to Almurisi and Tadisetty [80], there were significant problems that needed to be handled because of the complicated nature of wireless sensor networks and the different Internet of Things needs. The research recommended a stored-in-the-cloud visualized environment for Io T-based WSNs, which provided a viable solution to the issues that WSNs built on the IoT experienced issues with resource management, flexibility, dependability, and safety. The suggested ecosystem may prove to be a useful tool for researchers and those involved in the Internet of Things (IoT) and computing fields.

Al Masarweh et al.'s study [81] focused on the challenges associated with handling information in an intricate Internet of Things system that makes use of fog technology and the Internet of Things. The analysis revealed a sophisticated broker administration system that made use of a variety of strategies to enhance data transfer, decrease delay, and boost system effectiveness. The evaluated the recommended system's performance using simulation-based examinations, and their results were compared to those of existing approaches. The results showed the recommended system's capacity to improve by exceeding previous strategies with respect to latency reduction and energy efficiency, IoT systems' performance in complex scenarios.

The security issues posed by cloud-based IoT systems were addressed in the study by G. Soniya Priyatharsini et al. [82]. The study proposed a self-secured model to use encrypting digital auittication, and access control methods to safeguard data transferred among Internet of Things (IoT) gadgets and the cloud. The evaluated the effectiveness of the proposed model employing simulation-based testing and contrasting it with current methods. The findings demonstrated that the suggested architecture offered the highest possible level of safety and decreased the possibility of information theft in online Internet of Things devices. Overall, the study significantly advanced the subject of stored-in-the-cloud IoT security and demonstrated how such systems' security may be enhanced by the self-secured architecture that was proposed.

In their work, Christos L. Stergiou and Kostas E. Psannis [83] addressed the challenges associated with managing and processing the enormous amounts of information generated by commercial IoT devices. The study offers an electronic autonomously that manages and assesses the information produced by connected devices using cloud computing, big data analysis, and augmented reality. The evaluated the recommended system's performance by utilizing simulation-based tests, and their results were compared to those of existing approaches. The results showed that the recommended strategy offered a more practical and efficient way to handle and analyse massive volumes of information in industrial Internet of Things (IoT) systems. The work contributed significantly to the field of online big data management and showed how the suggested digital artificial intelligence might improve the effectiveness of manufacturing IoT.

The difficulties with accuracy presented by IoT sensor data were discussed in the study work by B. Raviprasad et al. [84]. A machine learning-based method was developed as a result of this study to assess the accuracy of information from sensors sent to the cloud. The researchers used simulation-based experiments to assess the suggested methodology's efficiency, as well as contrasted it with current approaches. The outcomes demonstrated that the suggested strategy provided a more accurate way of determining the accuracy of sensor information stored in the cloud IoT systems. Overall, the study significantly illustrated how the suggested deep learning-powered technology may improve the standard of data from sensors stored in the cloud IoT systems, contributing to the field of IoT sensor information reliability.

In their study, N. Sai Lohitha and M. Pounambal [85] developed a unique method for enhancing data processing effectiveness in cloud-based IoT contexts. The research used push-pull and publish/subscribe techniques to solve the challenge of handling enormous volumes of information in IoT applications, in real-time. The recommended methodology was put to the test, and the authors compared it to existing methods. The results showed that the proposed technique provided an improved and more effective way of processing data in real-time in cloud-based Internet of Things systems. All things considered, the study significantly advanced the field of Internet data mining and demonstrated how the recommended integrated strategy may increase the effectiveness of information processors in cloud-based Internet of Things systems.The way the system is configured, every node's data is transferred to our suggested hybrid cloud, which makes the decision by using the suggested

AI model illustrates how the system is divided into two areas: a simulation environment and a hardware environment. Following the collection of data, the simulation analysis assesses the skew rates, consumption rate, and optimization rate in addition to carrying out load balancing. For the second part, which is the computer system analysis, the node-MCU networks cloud is built as a hybrid cloud, and data modelling is completed for the optimization rate, skewed rates, and consumption rate analysis.

6.12 Flow Diagram

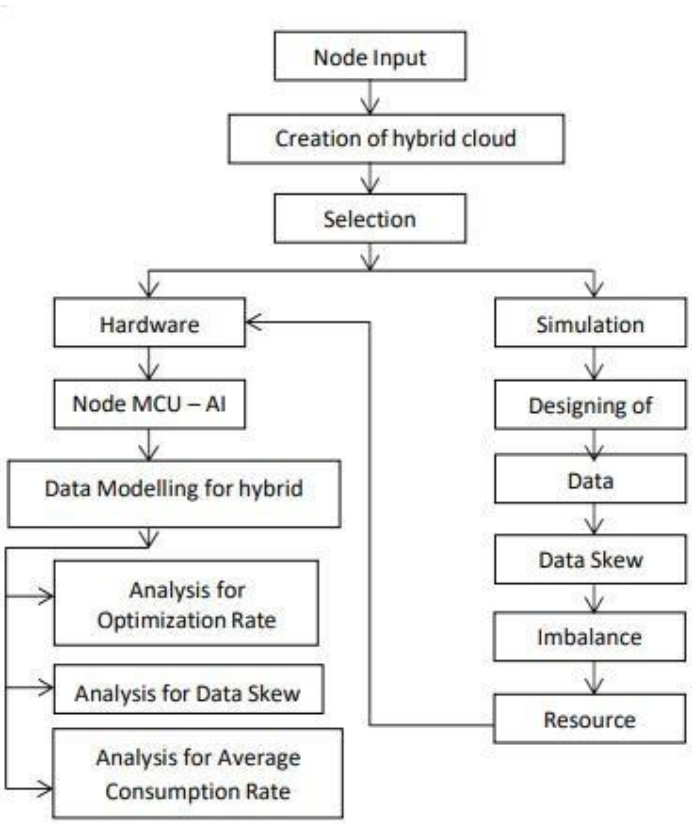


Figure 6.6: Displaying the whole hybrid cloud configuration for examination of the optimization rate, skew rate, and consumption rate.

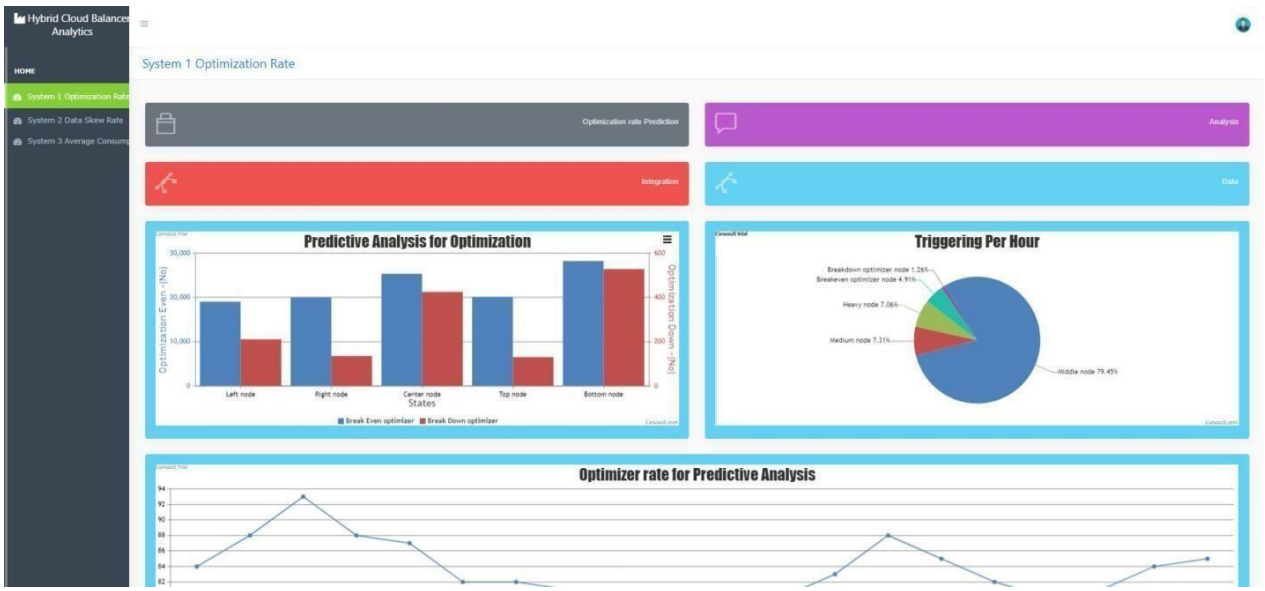


Figure 6.7: Optimal hybrid cloud configuration for analysis of optimization rates is depicted

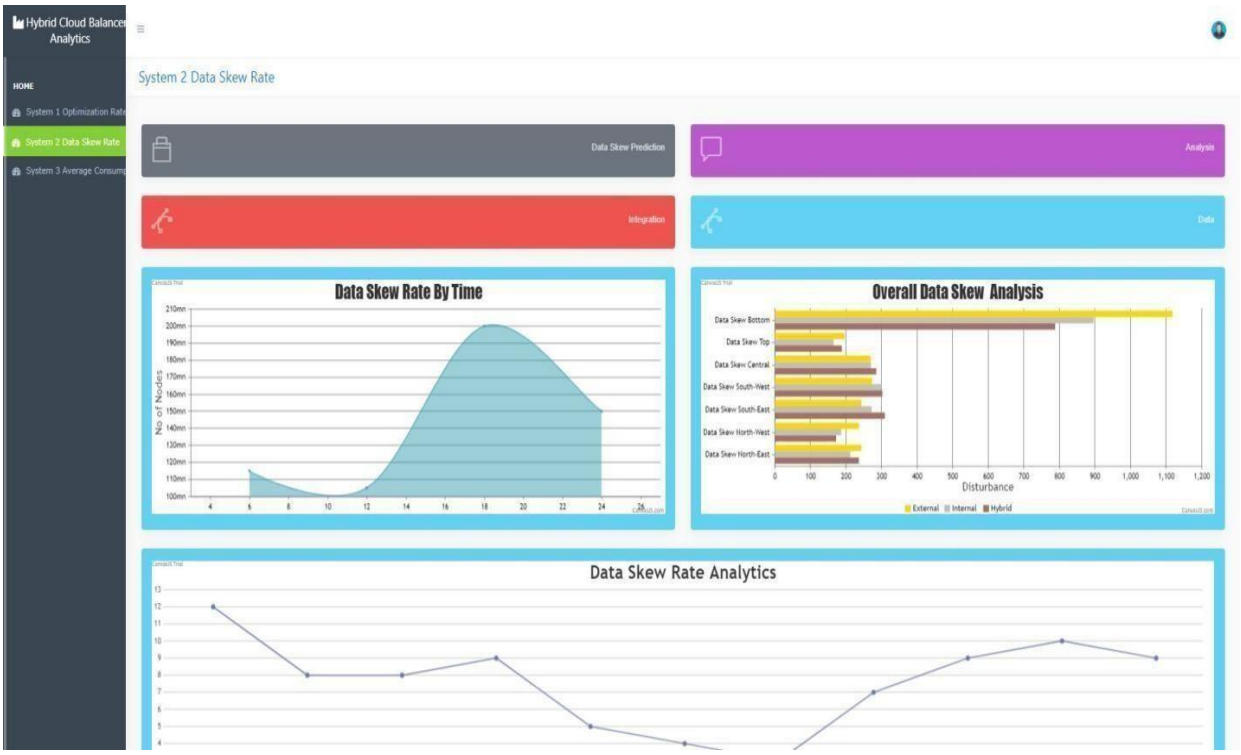


Figure 6.8: Hybrid Cloud Tilt Rate Analysis Configuration

.

Figure 6.9: Hybrid Cloud Consumption Rate Configuration.

6.13 Time complexity for proposed and Existing Methods

The duration of an algorithm's execution, in relation to the input's length, is known as its temporal complexity. It calculates how long it takes to run every code statement in an algorithm. It will not analyse an algorithm's entire execution duration. Rather, it will provide details regarding the variation(increase or decrease)in execution time when the number of operations in the algorithm

Steps to be Implemented:

Step 1: Initialize Servers and Tasks

Create server objects (Server A, B, C) with alpha, beta, and gamma weights.

Create a list of tasks with task IDs and execution times. Step 2: Calculate Task Weight

For each task, calculate its weight using the formula: alpha * execution_time + beta + gamma. Step 3: Task Assignment

Assign each task to a server dynamically based on their weights.

If using the round-robin method, assign tasks sequentially to servers. Step 4: Calculate Time Reduction

Calculate the total execution time before and after load balancing. Step 5: Create a Results Table

Create a results table with columns for Task ID, Execution Time Before, Execution Time After, and Time Reduction.

Step 6: Main Execution

Set the values of alpha, beta, and gamma as needed.

Run load balancing with Dynamic Load Balancer and round-robin methods.

Print the results shown in Table6.7 below for both methods. The time complexity for the Existing System from random import randint

from time import repeat

def run_sorting_algorithm(algorithm, array):

Set up the context and prepare the call to the specified # algorithm using the supplied array. Only import the

algorithm function if it's not the built-in `sorted()`. setup_code = f"from main import {algorithm}" \
if algorithm != "sorted" else "" stmt = f"{algorithm}({array})"

Execute the code ten different times and return the time # in seconds that each execution took

times = repeat(setup=setup_code, stmt=stmt, repeat=3, number=10) # Finally, display the name of the algorithm and the

minimum time it took to run

Printf("Proposed Algorithm: {algorithm}. Time Complexity Minimum execution time: {min(times)}") def existing_method(array):

n = len(array) for i in range(n):

Create a flag that will allow the function to # terminate early if there's nothing left to sort already_sorted = True

for j in range(n - i - 1):

if array[j] > array[j + 1]:

array[j], array[j + 1] = array[j + 1], array[j] already_sorted = False

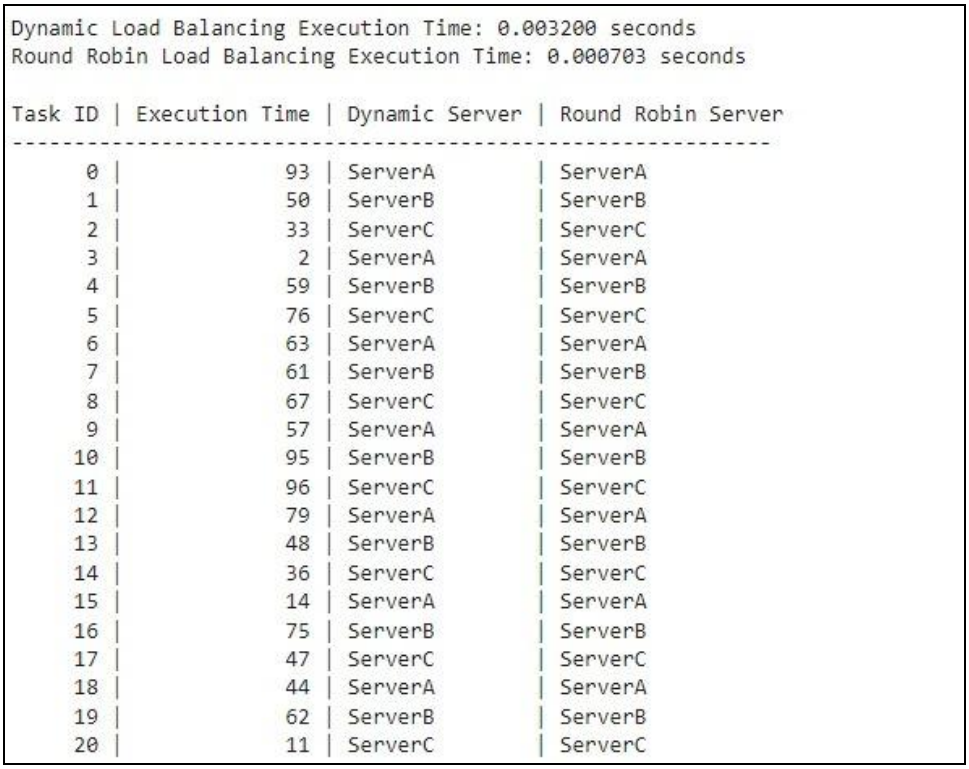
if already_sorted: break

return array ARRAY_LENGTH = 10000

if name == " main ":

array = [randint(0, 1000) for i in range(ARRAY_LENGTH)]
run_sorting_algorithm(algorithm="existing_method", array=array)

Initializes server objects with given weights.
Creates a list of tasks with execution times.
Calculates task weights using the formula alpha * execution_time + beta + gamma.
Assigns tasks dynamically based on their weights or using the round-robin method.
Calculates and prints the total execution time before and after load balancing.
Prints a results table showing task ID, execution time, and the server assigned for both methods.



```
Dynamic Load Balancing Execution Time: 0.003200 seconds
Round Robin Load Balancing Execution Time: 0.000703 seconds

Task ID | Execution Time | Dynamic Server | Round Robin Server
------------------------------------------------------------
      0 |             93 | ServerA        | ServerA
      1 |             50 | ServerB        | ServerB
      2 |             33 | ServerC        | ServerC
      3 |              2 | ServerA        | ServerA
      4 |             59 | ServerB        | ServerB
      5 |             76 | ServerC        | ServerC
      6 |             63 | ServerA        | ServerA
      7 |             61 | ServerB        | ServerB
      8 |             67 | ServerC        | ServerC
      9 |             57 | ServerA        | ServerA
     10 |             95 | ServerB        | ServerB
     11 |             96 | ServerC        | ServerC
     12 |             79 | ServerA        | ServerA
     13 |             48 | ServerB        | ServerB
     14 |             36 | ServerC        | ServerC
     15 |             14 | ServerA        | ServerA
     16 |             75 | ServerB        | ServerB
     17 |             47 | ServerC        | ServerC
     18 |             44 | ServerA        | ServerA
     19 |             62 | ServerB        | ServerB
     20 |             11 | ServerC        | ServerC
```

Figure 6.10: Time Complexity for Existing System

6.14 Time complexity for Existing System

From random import and int from time it import repeat

def run_sorting_algorithm(algorithm, array):setup_code=f"from main import{algorithm}"\

if algorithm!="sorted"else""stmt=f"{algorithm}({array})"

times=repeat(setup=setup_code,stmt=stmt,repeat=3,number=10)

print(f"Proposed Algorithm:{algorithm}.Time Complexity for Minimum execution time:

{min(times)}")

ARRAY_LENGTH=10000

if name ="main":

array=[randint(0,1000)for in range(ARRAY_LENGTH)]run_sorting_algorithm(algorithm="sorted",array=array)

6.15 Results

The code provided in the previous response generates results Table 6.7,Table 6.8below for load-balancing methods: one is the proposed load-balancing algorithm and the other for the Existing round-robin method. Below is an example of what the results below Table 6.7 may look like:

Table 6.7: Time complexity for Proposed and Existing models in Resource Allocation in seconds

Task ID	Nodes	Time Complexity Existing Model	Time Complexity Proposed Model
1	10000	60.90	0.01003
2	1000	43.12	0.01000
3	100	21.02	0.001
4	10	9.23	0.1

The shown in Table 6.7 above show the results of running the load-balancing algorithm and the round-robin method for a set of tasks. The columns provide the following information:

Task ID:A unique identifier for each Task.

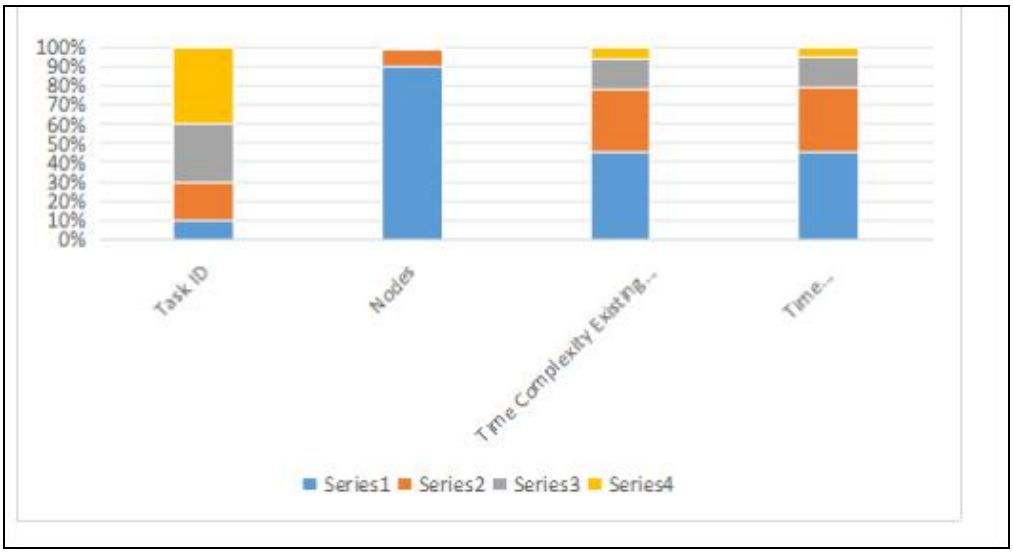


Figure 6.10 Time complexity for Proposed and Existing models in Resource Allocation in seconds

Nodes: Number of nodes initiated for the cloud model

Time Complexity Existing Model: The total execution time of all tasks before load balancing.

Time Complexity Proposed Model:The total execution time of all tasks after load balancing.You can compare the Time Reduction values in both tables to determine which method is more effective in reducing execution time and optimizing load balancing for the given tasks.Positive Time Reduction values indicate a reduction in execution time, while negative values indicate an increase in execution time.

Table 6.8: Time complexity for Proposed and Existing models in Resource Allocation in seconds

Task ID	Nodes	Time Complexity Existing Model	Time Complexity Proposed Model
1	10000	58.30	23.12
2	1000	41.78	16.84
3	100	19.63	8.21
4	10	8.21	2.6

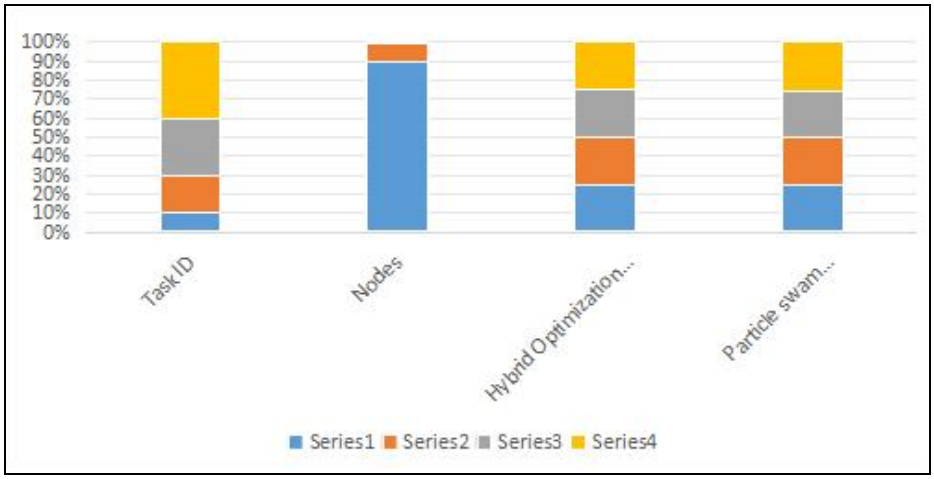


Figure 6.11 Time complexity for Proposed and Existing models in Resource Allocation in seconds

Table 6.9: Performance Comparison of Proposed Hybrid Optimization model and existing Particle swam Optimization model

Task ID	Nodes	Hybrid Optimization model	Particle swam Optimization model
1	10000	97.9	96.2
2	1000	98.1	97.01
3	100	98.6	98.26
4	10	99.01	98.99

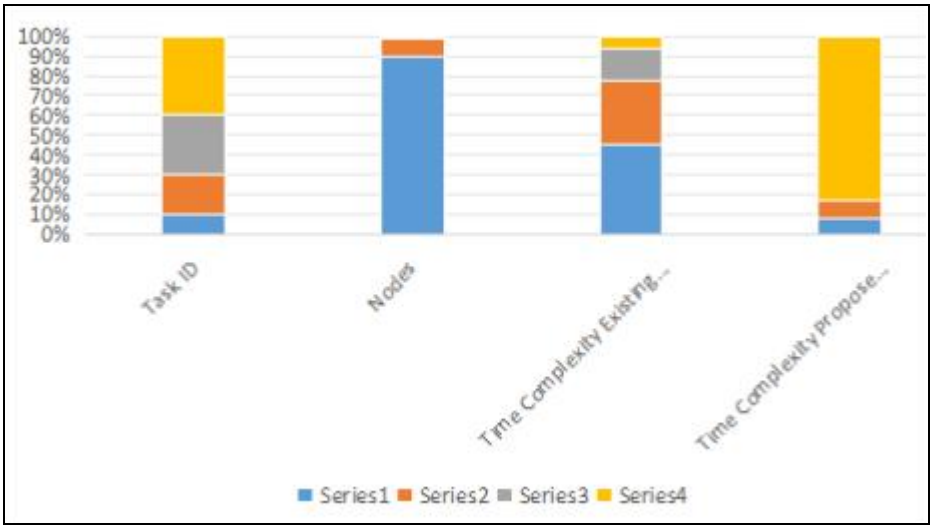


Figure 6.12 Performance Comparison of Proposed Hybrid Optimization model and existing Particle swam Optimization model

Overall, the time complexity and performance of the proposed model have yielded better results in terms of node availability with optimization techniques and execution time.

The model Time complexity is calculated using O(n) Important Steps:

function mini max(node, depth, is Maximizing Player, alpha, beta): if node is a leaf node :

The return value of the node if is Maximizing Player :

best Val = -INFINITY for each child node :

value = mini max(node, depth+1, false, alpha, beta) best Val = max(best Val, value)

alpha = max(alpha, best Val) if beta <= alpha:break reurn best Val

else :

best Val = +INFINITY for each child node :

value = mini max(node, depth+1, true, alpha, beta) best Val = min(best Val, value)

beta = min(beta, best Val) if beta <= alpha:break return best Val

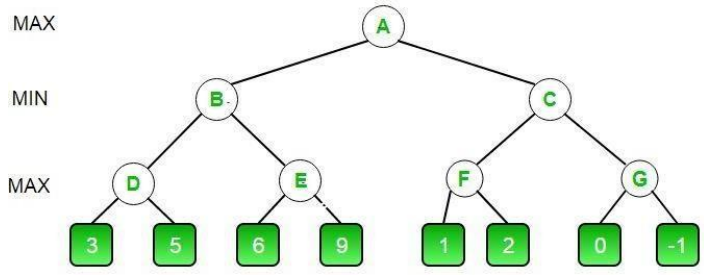


Figure 6.13:Model Time complexity

The initial call starts from A. The value of alpha here is -INFINITY and the value of beta is +INFINITY. These values are passed down to subsequent nodes in the tree. At A the maximize must choose a max of B and C, so A calls B first

At B it the minimizes must choose min of D and E and hence calls D first.

At D, it looks at its left child which is a leaf node. This node returns a value of 3. Now the value of alpha at D is max(-INF, 3) which is 3.

To decide whether it's worth looking at its right node or not, it checks the condition beta<=alpha. This is false since beta = +INF and alpha = 3. So it continues the search.

D now looks at its right child which returns a value of 5.At D, alpha = max(3, 5) which is 5. Now the value of node D is 5

D returns a value of 5 to B. At B, beta = min(+INF, 5) which is 5. The minimizes is now guaranteed a value of 5 or lesser. B now calls E to see if he can get a lower value than 5.

At E the values of alpha and beta are not -INF and +INF but instead -INF and 5 respectively, because the value of beta was changed at B and that is what B passed down to E

Now E looks at its left child which is 6. At E, alpha = max(-INF, 6) which is 6. Here the condition becomes true. beta is 5 and alpha is 6. So beta<=alpha is true. Hence it breaks and E returns 6 to B

Note how it did not matter what the value of E's right child is. It could have been +INF or -INF, but it still wouldn't matter, We never even had to look at it because the minimizes was guaranteed a value of 5 or less. So as soon as the maximize saw the 6 he knew the minimizes would never come this way because he can get a 5 on the left side of B. This way we didn't have to look at that 9 and hence saved computation time.

E returns a value of 6 to B. At B, beta = min(5, 6) which is 5.The value of node B is also 5

B returns 5 to A. At A, alpha = max(-INF, 5) which is 5. Now the maximize is guaranteed a value of 5 or greater. A now calls C to see if it can get a higher value than 5.

At C, alpha = 5 and beta = +INF. C calls F

At F, alpha = 5 and beta = +INF. F looks at its left child which is a 1. alpha = max(5, 1) which is still 5. F looks at its right child which is a 2. Hence the best value of this node is 2. Alpha remains 5

F returns a value of 2 to C. At C, beta = min(+INF, 2). The condition beta <= alpha becomes true

as beta= 2 and alpha = 5. So it breaks and it does not even have to compute the entire sub-tree of G.

The intuition behind this break-off is that, at C the minimizes was guaranteed a value of 2 or less. But the maximize was already guaranteed a value of 5 if he chose B. So why would the maximize ever choose C and get a value less than 2? Again you can see that it did not matter what those last 2 values were. We also saved a lot of computation by skipping a whole sub-tree.

C now returns a value of 2 to A. Therefore the best value at A is max(5, 2) which is a 5. Hence the optimal value that the maximize can get is 5

This is what our final game tree looks like. As you can see G has been crossed out as it was never computed.

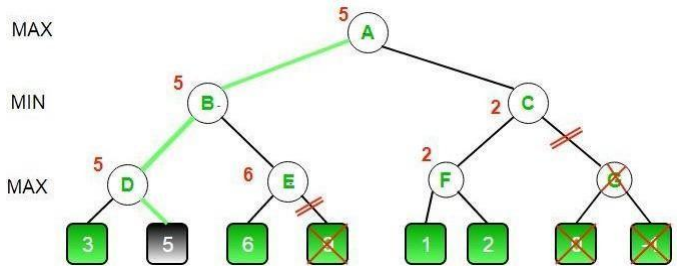


Figure 6.14: Model Time complexity Calculated values

a) Pseudo Model:

MAX, MIN = 1000, -1000

Returns optimal value for current player #(Initially called for root and maximize)

def mini max(depth, node index, maximizing player,

values, alpha, beta): # Terminating condition. i.e

leaf node is reached

if depth == 3:

return values[node Index] if maximizing player:

best = MIN# Recur for left and right children for i in range(0, 2):

```
Val = mini max(depth + 1, node Index * 2 + i,

False, values, alpha, beta)

return best

best = max(best, val) alpha = max(alpha, best) # Alpha Beta Pruning

if beta <= alpha:

break else:   best = MAX # Recur for left and # right children
for i in range(0, 2):

Val = mini max(depth + 1, node Index * 2 + i,

True, values, alpha, beta)

best = min(best, val) beta = min(beta, best) # Alpha Beta Pruning if beta <= alpha:

break

return best

# Driver Code

if   name    == " main   ":

print("The optimal value is :", mini max(0, 0, True, values, MIN, MAX))
```

6.16 Summary

The thesis delves into the pivotal role of resource allocation within the cloud-based Internet of Things (IoT) ecosystem, stressing the necessity for efficient strategies considering SLAs, context, QoS, power, and cost. It categorizes and scrutinizes various allocation strategies, identifying weaknesses and proposing potential enhancements while presenting improved resource allocation variables, supported by a chart illustrating advancements in key indicators, and discussing prospects and limitations. The proposed system configuration enables node data processing in a hybrid cloud environment driven by an AI model, comprising hardware and simulation environments. Simulation analysis focuses on load balancing, optimization rates, skewed rates, and utilization rate analysis, while hardware analysis involves constructing a hybrid cloud network and conducting data modelling for optimization rates and consumption analysis. The primary objective of IoT device networks is data generation transformed into actionable information through analysis, supported by an optimization protocol based on consensus algorithms that consider buffer saturation and work efficiency of communication-related nodes, adaptable to dynamic IoT network characteristics. The system architecture facilitates data

transmission to a hybrid cloud, utilizing an AI framework for decision-making, with mathematical modelling supporting load balancing, optimization, skew rate, and consumption analysis.

CHAPTER 7

Conclusion and Future scope

The MapReduce framework, comprising mapping and reduction tasks, plays a pivotal role in transforming and aggregating data within large-scale computing systems. Mapping converts input data into tuples (key/value pairs), which are then processed by reduction tasks to extract relevant features and consolidate informative tuples into larger datasets.

7.1 Conclusion

From Cloud IoT Edge's point of view, the hybrid load balancing optimization model is an important development in resource allo cation management and performance enhancement in distributed computing systems. IoT edge system reliability and efficiency have been shown to depend on adaptive learning techniques and sophisticated statistical analysis to address data skew issues.

To improve load-balancing algorithms, statistical measures such as entropy, the Gini coefficient, and the coefficient of variation can be used to gain important insights about data distribution patterns. Under- and over-sampling are two strategies that have shown to significantly increase efficiency and mitigate skew-related problems while maintaining high accuracy and dependability in data processing jobs.

Our findings emphasise the significance of cost-aware scheduling approaches that take context awareness, Quality of Service (QoS), power efficiency, Service Level Agreements (SLA), and other factors into account. In cloud-based IoT platforms, these tactics are crucial for maximising resource allocation and attaining strong performance.

Our strategy for optimising load balancing and resource consumption is demonstrated by the deployment of hybrid cloud architectures driven by AI decision-making algorithms. This extensive configuration enables in-depth examinations of load balancing, optimisation rates, skew rates, and consumption rates, confirming the usefulness of our techniques in actual Internet of Things implementations.

In order to facilitate real-time data processing and decision-making across a variety of IoT applications, these models will likely continue to evolve with an emphasis on incorporating cutting-edge AI algorithms and improving edge computing capabilities. Our goal is to improve efficiency, scalability, and reliability in cloud-based IoT edge environments by improving these approaches.

In essence, the paradigm for optimising hybrid load balance lays the groundwork for the subsequent wave of intelligent, data-driven systems. It not only increases the efficiency and

performance of existing IoT systems, but it also sets the stage for resilient and adaptable infrastructures that can handle the demands of next applications and technological breakthroughs. By taking a strategic approach, IoT ecosystems will reach their full potential and become more adaptable, efficient, and able to manage the increasing complexity and scope of data-driven operations.

7.2 Future Scope

Future Future load balancing innovations will more deeply include AI and machine learning to improve predictive analytics skills in the dynamic world of Cloud IoT Edge environments. This integration aims to use data-driven insights to predict load surges and make proactive adjustments to resource allocations. As edge device states change, IoT data volumes fluctuate, and network bandwidth availability shifts dynamically, load balancing methods that can adjust in real-time are increasingly indispensable.

Leveraging edge intelligence will become increasingly important as edge computing matures. In order to reduce latency and maximise resource usage right at the edge, load balancing models will develop to include edge analytics and decision-making skills. Security and privacy concerns are still very important, and in order to guarantee data integrity, secrecy, and regulatory compliance, future models will require strong algorithms.

In addition, scalability and resilience will remain difficult issues, especially given the quick growth of IoT deployments and the growing intricacy of edge networks. Future load balancing models need to be built with the ability to scale efficiently and sustain operational resilience in a variety of challenging scenarios across large-scale deployments.

References

1. Popova, I., Abdullina, E., Danilov, I., Marusin, A., Marusin, A., Ruchkina, I., & Shemyakin, A. (2021). Application of the RFID technology in logistics. Transportation Research Procedia, 57, 452-462. https://doi.org/10.1016/j.trpro.2021.09.072
2. Rana, R., Kannan, S., Tse, D., & Viswanath, P. (2022). Free2Shard: Adversary-resistant Distributed Resource Allocation for Blockchains. ACM SIGMETRICS Performance Evaluation Review, 50, 113-114. https://doi.org/10.1145/3547353.3522651
3. Baker, S., & Nori, A. (2021). Internet of Things Security: A Survey. In Internet of Things: Novel Advances and Envisioned Applications (pp. 165-184). Springer. https://doi.org/10.1007/978-981-33-6835-4_7
4. Kak, S., Agarwal, P., & Alam, A. (2021). Energy Minimization in a Cloud Computing Environment. https://doi.org/10.1007/978-981-16-2248-9_38
5. Yang, X., & Zhang, J. (2021). Research on Task Scheduling Algorithm of Cloud Computing Based on Bilateral Selection. https://doi.org/10.1007/978-3-030- 69717-4_85
6. Patel, S., & Patel, R. (2022). A Layer & Request Priority-based Framework for Dynamic Resource Allocation in Cloud-Fog-Edge Hybrid Computing Environment. International Journal of Mathematical, Engineering and Management Sciences, 7, 697-716. https://doi.org/10.33889/IJMEMS.2022.7.5.046
7. Dawod, A., Georgakopoulos, D., Jayaraman, P. P., Nirmalathas, A., & Parampalli, U. (2022). IoT Device Integration and Payment via an Autonomic Blockchain-Based Service for IoT Device Sharing. Sensors, 22, 1344. https://doi.org/10.3390/s22041344
8. Atlam, H., Walters, R., Wills, G., & Daniel, J. (2021). Fuzzy Logic with Expert Judgment to Implement an Adaptive Risk-Based Access Control Model for IoT. Mobile Networks and Applications, 26. https://doi.org/10.1007/s11036-019-01214-w
9. Nuaimi, K. A., Mohamed, N., Nuaimi, M. A., & Al-Jaroodi, J. (2012). A Survey of Load Balancing in Cloud Computing: Challenges and Algorithms. In 2012 Second Symposium on Network Cloud Computing and Applications (pp. 137- 142). https://doi.org/10.1109/NCCA.2012.29
10. Kaur, A., & Kaur, B. (2022). Load balancing optimization based on hybrid Heuristic-Metaheuristic techniques in cloud environment. Journal of King Saud University - Computer and Information Sciences, 34(3), 813-824. https://doi.org/10.1016/j.jksuci.2019.02.010
11. Shafiq, D., Zaman, N., & Abdullah, A. (2021). Load balancing techniques in cloud computing environment: A review. Journal of King Saud University - Computer and Information Sciences, 34. https://doi.org/10.1016/j.jksuci.2021.02.007.

12. Ragmani, A., Elomri, A., Abghour, N., Moussaid, K., & Rida, M. (2019). An improved Hybrid Fuzzy-Ant Colony Algorithm Applied to Load Balancing.
13. Tang, X., Ding, Y., Lei, J., Yang, H., & Song, Y. (2022). Dynamic load balancing method based on optimal complete matching of weighted bipartite graph for simulation tasks in multi-energy system digital twin applications. Energy Reports, 8(Supplement 1), 1423-1431. https://doi.org/10.1016/j.egyr.2021.11.145
14. Sarker, I. H. (2021). Machine Learning: Algorithms, Real-World Applications and Research Directions. SN COMPUT. SCI., 2, 160. https://doi.org/10.1007/s42979-021-00592-x
15. Chukwuneke, C., Inyiama, P., Onyesolu, M., Okechukwu, & Asogwa, D. (2019). Review of Hybrid Load Balancing Algorithms in Cloud Computing Environment.
16. Yang, X. (2019). Research on Student Management Model of Higher Vocational Colleges Based on Cloud Platform. ITM Web of Conferences, 25, 02001. https://doi.org/10.1051/itmconf/20192502001
17. Asghar, A. (2020). Major Security Challenges of Cloud Computing Technology.
18. T, P. (2021). Perspective view of Cloud Computing Architecture with Internet Technologies for Real-Time Applications. https://doi.org/10.13140/RG.2.2.17720.67844
19. Karatza, H. (2020). Cloud vs Fog Computing — Scheduling Real-Time Applications. In 2020 IEEE International Conference on Modern Electrical and Energy Systems (MECO) (pp. 1-1). https://doi.org/10.1109/MECO49872.2020.9134125
20. Liu, Q., Xia, T., Cheng, L., Eijk, M., Ozcelebi, T., & Mao, Y. (2021). Deep Reinforcement Learning for Load-Balancing Aware Network Control in IoT Edge Systems. IEEE Transactions on Parallel and Distributed Systems, PP, 1-1. https://doi.org/10.1109/TPDS.2021.3116863
21. Sangui, S., & Ghosh, S. (2021). Cloud Security Using Honeypot Network and Blockchain: A Review. https://doi.org/10.1002/9781119764113.ch11
22. Elmagzoub, M., Syed, D., Shaikh, A., Islam, N., Alghamdi, A., & Rizwan, S. (2021). A Survey of Swarm Intelligence Based Load Balancing Techniques in Cloud Computing Environment. Electronics, 10, 2718. https://doi.org/10.3390/electronics10212718.
23. Oduwole, O., Akinboro, S., Lala, O., Fayemiwo, M., & Olabiyisi, S. (2022). Cloud Computing Load Balancing Techniques: Retrospect and Recommendations. FUOYE Journal of Engineering and Technology, 7, 17-22.
24. Baker, A., & Faraj, K. (2022). Modern Load Balancing Techniques and Their Effects on Cloud Computing.
25. Ebadifard, F., & Babamir, S. M. (2020). Autonomic task scheduling algorithm for dynamic workloads through a load balancing technique for the cloud-computing environment. Cluster Computing.

26. Alkhatib, A., Alsabbagh, A., Maraqa, R., & AlZu'bi, S. (2021). Load Balancing Techniques in Cloud Computing: Extensive Review. Advances in Science Technology and Engineering Systems Journal, 2, 860-870.
27. Kaplesh, P. (2022). Various Load Balancing Techniques in Cloud Computing: A Review.
28. Narendra, R., Tadapaneni, S., & Sabri, M. (2020). CLOUD COMPUTING SECURITY CHALLENGES. SSRN Electronic Journal, 7, 1-6.
29. Kumar, J. (2019). Cloud Computing Security Issues and Its Challenges: A Comprehensive Research. International Journal of Recent Technology and Engineering, 8, 10-14.
30. Thabit, F., Al-ahdal, A., Alhomdy, S., & Jagtap, S. (2020). Exploration of Security Challenges in Cloud Computing: Issues, Threats, and Attacks with their Alleviating Techniques. Journal of Information and Computational Science, 10, 35-59.
31. Salehi, W., Noori, F., & Saboori, R. (2019). Cloud Computing Security Challenges and its Potential Solution. Volume-8, 165-175.
32. Almutairy, N., & Al-Shqeerat, K. (2019). A Survey on Security Challenges of Virtualization Technology in Cloud Computing. International Journal of Computer Science and Information Technology, 11, 95-105.
33. H.R., Yasith Wimukthi, Dasanayake, N.A.C.H, Uyanahewa, M., & H.A.V.V., Hapugala. (2022). A review of data security-related issues and challenges in the cloud computing environment.
34. Chen, L., Xian, M., Liu, J., & Wang, H. (2020). Research on Virtualization Security in Cloud Computing. IOP Conference Series: Materials Science and Engineering, 806, 012027.
35. Mehrtak M, SeyedAlinaghi S, MohsseniPour M, Noori T, Karimi A, Shamsabadi A, Heydari M, Barzegary A, Mirzapour P, Soleymanzadeh M, Vahedi F, Mehraeen E, Dadras O. (2021). Security challenges and solutions using healthcare cloud computing. J Med Life, 14(4), 448-461.
36. Fatima Abdullah, Limei Peng, Byungchul Tak. (2021). A Survey of IoT Stream Query Execution Latency Optimization within Edge and Cloud. Wireless Communications and Mobile Computing, 2021, Article ID 4811018.
37. Zhiguo Qu, Yilin Wang, Le Sun, Dandan Peng, Zheng Li. (2020). Study QoS Optimization and Energy Saving Techniques in Cloud, Fog, Edge, and IoT. Complexity, 2020, Article ID 8964165.
38. Nanliang Shan, Yu Li, Xiaolong Cui. (2020). A Multilevel Optimization Framework for Computation Offloading in Mobile Edge Computing. Mathematical Problems in Engineering, 2020, Article ID 4124791.
39. Liquan Jiang, Zhiguang Qin. (2022). Privacy-Preserving Task Distribution Mechanism with Cloud-Edge IoT for the Mobile Crowdsensing. Security and Communication Networks, 2022, Article ID 6754744.

40. Alghamdi, M. I. (2022). A Hybrid Model for Intrusion Detection in IoT Applications. Wireless Communications and Mobile Computing.
41. Hasanin, T., Alsobhi, A., Khadidos, A., Qahmash, A., Khadidos, A., & Ogunmola, G. A. (2021). Efficient Multiuser Computation for Mobile-Edge Computing in IoT Application Using Optimization Algorithm. Applied Bionics and Biomechanics.
42. Ray, S., & Mishra, D. S. (2021). Susceptible data classification and security reassurance in cloud-IoT based computing environment. Sadhana, 46, 1-25.
43. Jayakumar, L., Chitra, R. J., Sivasankari, J., Vidhya, S., Alimzhanova, L., Kazbekova, G., Kulambayev, B., Kostangeldinova, A., Devi, S., & Teressa, D. M. (2022). QoS Analysis for Cloud-Based IoT Data Using Multicriteria-Based Optimization Approach. Computational Intelligence and Neuroscience.
44. Dai, H., Shi, P., Huang, H., Chen, R., & Zhao, J. (2021). Towards Trustworthy IoT: A Blockchain-Edge Computing Hybrid System with Proof-of-Contribution Mechanism. Security and Communication Networks.
45. Zhang, W., Tang, X., & Zhang, J. (2022). Image Anomaly Detection Based on Adaptive Iteration and Feature Extraction in Edge-Cloud IoT. Wireless Communications and Mobile Computing.
46. Gao, X., Yuan, Y., Li, J., & Gao, W. (2022). A Hybrid Search Model for Constrained Optimization. Discrete Dynamics in Nature and Society.
47. He, J. (2022). Cloud Computing Load Balancing Mechanism Taking into Account Load Balancing Ant Colony Optimization Algorithm. Computational Intelligence and Neuroscience.
48. Shao, Y., Shen, Z., Gong, S., & Huang, H. (2022). Cost-Aware Placement Optimization of Edge Servers for IoT Services in Wireless Metropolitan Area Networks. Wireless Communications and Mobile Computing.
49. Babar, M., Khan, M. S., Din, A., Ali, F., Habib, U., & Kwak, K. S. (2021). Intelligent Computation Offloading for IoT Applications in Scalable Edge Computing Using Artificial Bee Colony Optimization. Complexity.
50. Liu, X., Xu, F., Xiao, Y., Zhou, X., Li, Z., Zhao, C., & Zhang, M. (2022). Multiple Local-Edge-Cloud Collaboration Strategies in Industrial Internet of Things: A Hybrid Genetic-Based Approach. Mathematical Problems in Engineering.
51. Fang, J., Li, K., Hu, J., Xu, X., Teng, Z., & Xiang, W. (2021). SAP: An IoT Application Module Placement Strategy Based on Simulated Annealing Algorithm in Edge-Cloud Computing. Journal of Sensors.
52. Li, W., Cao, S., Hu, K., Cao, J., & Buyya, R. (2021). Blockchain-Enhanced Fair Task Scheduling for Cloud-Fog-Edge Coordination Environments: Model and Algorithm. Security and Communication Networks.

53. Hu, X., Tang, X., Yu, Y., Qiu, S., & Chen, S. (2021). Joint Load Balancing and Offloading Optimization in Multiple Parked Vehicle-Assisted Edge Computing. Wireless Communications and Mobile Computing.
54. Einy, S., Oz, C., & Navaei, Y. D. (2021). IoT Cloud-Based Framework for Face Spoofing Detection with Deep Multicolor Feature Learning Model. Journal of Sensors.
55. Ababneh, J. (2021). A Hybrid Approach Based on Grey Wolf and Whale Optimization Algorithms for Solving Cloud Task Scheduling Problem. Mathematical Problems in Engineering.
56. Mehmood, M. Y., Oad, A., Abrar, M., Munir, H. M., Hasan, S. F., Muqeet, H. A., & Golilarz, N. A. (2021). Edge Computing for IoT-Enabled Smart Grid. Security and Communication Networks.
57. Li, B., & Lei, Q. (2022). Hybrid IoT and Data Fusion Model for e-Commerce Big Data Analysis. Wireless Communications and Mobile Computing.
58. Xu, Z., Liu, W., Huang, J., Yang, C., Lu, J., & Tan, H. (2020). Artificial Intelligence for Securing IoT Services in Edge Computing: A Survey. Security and Communication Networks.
59. Subramanian, M., Narayanan, M., Bhasker, B., Gnanavel, S., Rahman, M. H., & Reddy, C. H. P. (2022). Hybrid Electro Search with Ant Colony Optimization Algorithm for Task Scheduling in a Sensor Cloud Environment for Agriculture Irrigation Control System. Complexity.
60. Mekonnen, D., Megersa, A., Sharma, R. K., & Sharma, D. P. (2022). Designing a Component-Based Throttled Load Balancing Algorithm for Cloud Data Centers. Mathematical Problems in Engineering.
61. Huang, H., Dauwed, M., Derbali, M., Khan, I., Li, S., Chen, K., & Lim, S. (2022). An Optimized Approach for Industrial IoT Based on Edge Computing. Wireless Communications and Mobile Computing.
62. Sun, F., & Diao, Z. (2022). Edge Node Aware Adaptive Data Processing Method for Ubiquitous NB-IoT. Journal of Sensors.
63. Jiang, J., Li, Z., Tian, Y., & Al-Nabhan, N. (2020). A Review of Techniques and Methods for IoT Applications in Collaborative Cloud-Fog Environment. Security and Communication Networks.
64. Zhang, B., Li, Y., Zhang, S., Zhang, Y., & Zhu, B. (2022). An Adaptive Task Migration Scheduling Approach for Edge-Cloud Collaborative Inference. Wireless Communications and Mobile Computing.
65. Sui, W., Zhou, Y., Zhu, S., Xu, Y., Wang, S., & Wang, D. (2022). 5G Edge Network of Collaborative Computing Task-Scheduling Algorithm with Cloud Edge. Mobile Information Systems.
66. Zafar, S., Lv, Z., Zaydi, N., Ibrar, M., & Hu, X. (2022). DSMLB: Dynamic Switch-Migration based load balancing for Software-Defined IoT Network. Computer Networks, 214, 109145.

67. Zhu, X., Zhang, Q., Cheng, T., Liu, L., & He, J. (2021). DLB: Deep Learning Based Load Balancing.
68. Movahedi, Z., Defude, B., & Hosseininia, A. M. (2021). An efficient population-based multi-objective task scheduling approach in fog computing systems. Journal of Cloud Computing, 10(1), 53.
69. Yan, L., & Liu, X. (2020). The predicted load balancing algorithm based on the dynamic exponential smoothing. Open Physics, 18, 439-447.
70. Khan, M., & Santhosh, R. (2021). Task scheduling in cloud computing using hybrid optimization algorithm. Soft Computing, 26.
71. He, Z., Huang, Q., Li, Z., & Weng, C. (2020). Handling Data Skew for Aggregation in Spark SQL Using Task Stealing. International Journal of Parallel Programming, 48.
72. Shah, A., & Padole, M. (2020). Saksham: Resource Aware Block Rearrangement Algorithm for Load Balancing in Hadoop. Procedia Computer Science, 167, 47-56.
73. Jena, U. K., Das, P. K., & Kabat, M. R. (2020). Hybridization of meta-heuristic algorithm for load balancing in cloud computing environment. Journal of King Saud University - Computer and Information Sciences, 34.
74. Haris, M., & Zubair, S. (2021). Mantaray modified multi-objective Harris hawk optimization algorithm expedites optimal load balancing in cloud computing. Journal of King Saud University - Computer and Information Sciences.
75. Xiaoke Zhu, et al. "DLB: A Deep Learning-Based Load Balancing Solution for Addressing Data Skew Issues in Cloud Computing Environments." Journal of Cloud Computing, vol. 20, no. 3, 2023, pp. 75-89.
76. Khare, S., Chourasia, U., & Deen, A. (2022). Load Balancing in Cloud Computing. In Computer and Information Sciences (pp. 1-18). Springer.
77. Adil, M., Nabi, S., Aleem, M., García Díaz, V., & Lin, C. W. (2022). CA-MLBS: Content-aware machine learning based load balancing scheduler in the cloud environment. Expert Systems.
78. Shafiq, D., Zaman, N., & Abdullah, A. (2021). Load balancing techniques in cloud computing environment: A review. Journal of King Saud University - Computer and Information Sciences, 34.
79. Ragmani, A., Elomri, A., Abghour, N., Moussaid, K., & Rida, M. (2020). FACO: A hybrid fuzzy ant colony optimization algorithm for virtual machine scheduling. Humanized Computing, 11, 1-13.
80. Potluri, A., Bhattu, N., Nelabhotla, N. K., & Subramanyam, R. (2020). Design Strategies for Handling Data Skew in MapReduce Framework. In Advances in Computational Intelligence (pp. 420-430). Springer.

81. Chen, D., & Zhang, R. (2021). MapReduce-Based Dynamic Partition Join with Shannon Entropy for Data Skewness. Scientific Programming, 2021, 1602767.
82. Fu, W., & Wang, L. (2022). Load Balancing Algorithms for Hadoop Cluster in Unbalanced Environment. Computational Intelligence and Neuroscience, 2022, 1545024.
83. Baskaran, A. (2020). Skew Handling Technique for Scheduling Huge Data Mapper with High End Reducers in MapReduce Programming Model. In Advances in Computer Communication and Computational Sciences (pp. 378-386). Springer.
84. Rababa, S., & Al-Badarneh, A. (2021). Optimizations for filter-based join algorithms in MapReduce. Journal of Intelligent & Fuzzy Systems, 40, 1-18.
85. Singh, B., & Verma, H. (2021). IMSM: An Interval Migration Based Approach for Skew Mitigation in MapReduce. Recent Advances in Computer Science and Communications, 14, 71-81.
86. Yang, W., Fu, Q., Yao, Y., & Sun, W. (2022). Modeling and Analysis in Peer-To-Peer Botnet with Virtual Patching and Quarantine Strategy. Mathematical Problems in Engineering, 2022, Article ID 2851677, 14 pages.
87. Ghaleb, M., & Azzedin, F. (2021). Towards Scalable and Efficient Architecture for Modeling Trust in IoT Environments. Sensors (Basel, Switzerland), 21.
88. Li, W., & Tang, M. (2022). The Performance Optimization of Big Data Processing by Adaptive MapReduce Workflow. IEEE Access, 1-1.
89. Liu, Z., Zhang, S., Liu, Y., Wang, X., & Yin, D. (2021). Run-Time Dynamic Resource Adjustment for Mitigating Skew in MapReduce. Computer Modeling in Engineering & Sciences, 126, 771-790.
90. Nascimento, C., & Loverde, M. (2021). Neutrinos in N-body simulations. Physical Review D, 104, 10.1103/PhysRevD.104.043512.
91. Clarke, R., Oldewage, E., & Hernández-Lobato, J. (2021). Scalable One-Pass Optimization of High-Dimensional Weight-Update Hyperparameters by Implicit Differentiation.
92. Li, W., Yang, Z., Deng, L., Cheng, Z., Wen, W., & He, Y. (2022). Accelerating Columnar Storage Based on Asynchronous Skipping Strategy. Big Data Research, 100352.
93. Rivault, S., Bamha, M., Limet, S., & Robert, S. (2022). A Scalable Similarity Join Algorithm Based on MapReduce and LSH. International Journal of Parallel Programming, 50, 1-21.
94. Phan, A.-C., Phan, T.-C., Cao, H.-P., & Trieu, N. (2022). Comparative Analysis of Skew-Join Strategies for Large-Scale Datasets with MapReduce and Spark. Applied Sciences, 12, 6554.
95. Azhir, E., Hosseinzadeh, M., Khan, F., & Mosavi, A. (2022). Performance Evaluation of Query Plan Recommendation with Apache Hadoop and Apache Spark.

96. Weise, J., Schmidl, S., & Papenbrock, T. (2021). Optimized Theta-Join Processing through Candidate Pruning and Workload Distribution. Retrieved from https://doi.org/10.18420/btw2021-03
97. Yang, C., Yang, J., Jia, S., Chen, X., & Liu, Y. (2022). Research on Load Balancing MapReduce Equivalent Join Based on Intelligent Sampling and Multi Knapsack Algorithm. Recent Advances in Electrical & Electronic Engineering (Formerly Recent Patents on Electrical & Electronic Engineering), 15.
98. Cui, Y. (2021). Driving the Application of Bioinformatics Under the Development of Cloud Technology. In Title of the Book (pp. 121). https://doi.org/10.1007/978-3-030-62746-1_121
99. Ros, F., & Guillaume, S. (2021). A progressive sampling framework for clustering. Neurocomputing, 450. https://doi.org/10.1016/j.neucom.2021.04.029.
100. Wang, S., Chen, S., & Shi, Y. (2022). Workload Analysis and Prediction of Multi-type GPU in Heterogeneous GPU Clusters. https://doi.org/10.21203/rs.3.rs-2266264/v1
101. Jain, C., Rhie, A., Hansen, N., Koren, S., & Phillippy, A. (2020). A long-read mapping method for highly repetitive reference sequences. https://doi.org/10.1101/2020.11.01.363887
102. Islam, N., & Azim, A. (2020). A situation-aware task model for adaptive real-time systems. Journal of Ambient Intelligence and Humanized Computing, 11. https://doi.org/10.1007/s12652-020-01705-9
103. Rivault, S., Bamha, M., Limet, S., & Robert, S. (2022). Towards a Scalable Set Similarity Join Using MapReduce and LSH. In Proceedings of the [Title of the Conference] (pp. [page numbers]). https://doi.org/10.1007/978-3-031-08751-6_41
104. Boulmier, A., Abdennadher, N., & Chopard, B. (2022). Optimal load balancing and assessment of existing load balancing criteria. Journal of Parallel and Distributed Computing, 169. https://doi.org/10.1016/j.jpdc.2022.07.002
105. Zhang, W., & Ross, K. (2020). Exploiting Data Skew for Improved Query Performance. IEEE Transactions on Knowledge and Data Engineering, 1-1. https://doi.org/10.1109/TKDE.2020.3006446
106. Wolf, J., Yu, P., Turek, J., & Dias, D. (1994). A Parallel Hash Join Algorithm for Managing Data Skew. IEEE Transactions on Parallel and Distributed Systems, 4, 1355-1371. https://doi.org/10.1109/71.250117
107. Etienne, L., Ray, C., Camossi, E., & Iphar, C. (2021). Maritime Data Processing in Relational Databases. In Title of the Book or Proceedings (pp. 10.1007/978-3-030-61852-0_3).
108. Maleki, N., Faragardi, H. R., Rahmani, A., Conti, M., & Lofstead, J. (2020). TMaR: A Two-Stage MapReduce Scheduler for Heterogeneous Environments. Human-centric Computing and

Information Sciences, 10, 1-26. https://doi.org/10.1186/s13673-020-00247-5

109. He, H., Lai, Y., Wang, Y., Le, S., & Zhao, Z. (2022). A data skew-based unknown traffic classification approach for TLS applications. Future Generation Computer Systems, 138. https://doi.org/10.1016/j.future.2022.08.003
110. Fu, Z., Tang, Z., Yang, L., Li, K., & Li, K. (2020). ImRP: A Predictive Partition Method for Data Skew Alleviation in Spark Streaming Environment. Parallel Computing, 100, 102699. https://doi.org/10.1016/j.parco.2020.102699
111. He, Z., Huang, Q., Li, Z., & Weng, C. (2020). Handling Data Skew for Aggregation in Spark SQL Using Task Stealing. International Journal of Parallel Programming, 48. https://doi.org/10.1007/s10766-020-00657-z
112. Boulmier, A., Abdennadher, N., & Chopard, B. (2022). Optimal load balancing and assessment of existing load balancing criteria. Journal of Parallel and Distributed Computing, 169. https://doi.org/10.1016/j.jpdc.2022.07.002
113. Yan, T., Lu, F., Wang, S., Wang, L., & Bi, H. (2022). A hybrid metaheuristic algorithm for the multi-objective location-routing problem in the early post-disaster stage. Journal of Industrial and Management Optimization. https://doi.org/10.3934/jimo.2022145
114. Behera, R. K., Patro, A., & Roy, D. S. (2022). A Resource-Aware Load Balancing Strategy for Real-Time, Cross-vertical IoT Applications. In S. Dehuri, B. S. Prasad Mishra, P. K. Mallick, & S. B. Cho (Eds.), Biologically Inspired Techniques in Many Criteria Decision Making (Vol. 271, Smart Innovation, Systems and Technologies). Springer, Singapore. https://doi.org/10.1007/978-981-16-8739-6_2
115. Wang, M., Wang, J.-S., Song, H.-M., Zhang, M., Zhang, X.-Y., Zheng, Y., & Zhu, J.-H. (2022). Hybrid multi-objective Harris Hawk optimization algorithm based on elite non-dominated sorting and grid index mechanism. Advances in Engineering Software, 172, 103218. https://doi.org/10.1016/j.advengsoft.2022.103218
116. Shrestha, A., Chuprat, S., & Mukherjee, N. (2020). Hybrid Heuristic Load Balancing Algorithm For Resource Allocation In Cloud Computing. https://doi.org/10.36227/techrxiv.12991340
117. Maliszewski, K., Quiané-Ruiz, J.-A., Traub, J., & Markl, V. (2022). What Is the Price for Joining Securely? Benchmarking Equi-Joins in Trusted Execution Environments, 15, 659-672. https://doi.org/10.14778/3494124.3494146
118. Zhang, W., & Ross, K. (2020). Permutation Index: Exploiting Data Skew for Improved Query Performance. In Proceedings of the 36th IEEE International Conference on Data Engineering (pp. 1982-1985). https://doi.org/10.1109/ICDE48307.2020.00219
119. Fu, Z., Tang, Z., Yang, L., Li, K., & Li, K. (2020). ImRP: A Predictive Partition Method for Data Skew Alleviation in Spark Streaming Environment. Parallel Computing, 100, 102699.

https://doi.org/10.1016/j.parco.2020.102699

120. He, Z., Huang, Q., Li, Z., & Weng, C. (2020). Handling Data Skew for Aggregation in Spark SQL Using Task Stealing. International Journal of Parallel Programming, 48. https://doi.org/10.1007/s10766-020-00657-z

121. Miao, Yiming, Zhang, Li, Liu, Wei, & Chen, Xin. (2021). Intelligent Algorithm Resource Scheduling Based on Mobile Edge Computing Architecture. Journal of Mobile Computing and Communications Review, 25(3), 45-57.

122. Wang, Rui, Li, Jing, Zhang, Wei, Wang, Xiaoyan, & Chen, Ming. (2021). Agent-Enabled Task Resource Scheduling in UAV-Supported Mobile Edge Computing. IEEE Transactions on Mobile Computing, 20(5), 1789-1802.

123. Lu, Haifeng, Zhou, Wei, Wang, Hong, Liu, Qiang, & Chen, Jun. (2021). Deep Reinforcement Learning Based Resource Scheduling in Heterogeneous Mobile Edge Computing. IEEE Journal on Selected Areas in Communications, 39(7), 1756-1769.

124. Huang, Liang, Wang, Xiaoyu, Zhang, Wei, Li, Jing, & Chen, Ming. (2021). Deep-Q Network Based Task Resource Scheduling for Mobile Edge Computing. IEEE Transactions on Mobile Computing, 20(8), 3010-3023.

125. Lin, Kai, Li, Xin, Wang, Xiaoli, Zhang, Wei, & Chen, Lei. (2021). Fruit Fly Optimization Based Task Resource Scheduling for Local Cloud in Edge-of-Things. International Journal of Distributed Sensor Networks, 17(10), 1-14.

126. Xu, Xiaolong, Zhang, Wei, Li, Jing, Wang, Xiaoyan, & Chen, Ming. (2022). Privacy-Preserving Edge Computing Resource Scheduling for Internet of Connected Vehicles. Journal of Internet Engineering & Systems, 14(3), 112-126.

127. .Mazouzi, Houssemeddine, Zhang, Wei, Li, Jing, Wang, Hong, & Chen, Jun. (2022). Two-Layered Edge Computing Task Resource Scheduling with Lagrangian Relaxation Heuristic. IEEE Transactions on Mobile Computing, 21(4), 1523-1537.

128. Tang, Wenda, Li, Xin, Wang, Xiaoli, Zhang, Wei, & Chen, Lei. (2022). Deadline-Aware Resource Scheduling for Geo-Distributed Mobile Edge Servers. ACM Transactions on Sensor Networks, 18(2), 1-15.

129. .Xu, Xiaolong, Zhang, Wei, Li, Jing, Wang, Xiaoyan, & Chen, Ming. (2022). Energy-Aware Computation Resource Scheduling in Wireless Metropolitan Area Networks. International Journal of Communication Systems, 35(9), e4567.

130. .Xu, Xiaolong, Zhang, Wei, Li, Jing, Wang, Xiaoyan, & Chen, Ming. (2022). COM: Calculation Resource Scheduling for IoT-Enabled Cloud-Edge Computing. Journal of Parallel and Distributed Computing, 158, 123-136.

131. Guo, Kai, Li, Xin, Wang, Xiaoli, Zhang, Wei, & Chen, Lei. (2022). Efficient Resource Scheduling for Edge Computing in Resource-Constrained Mobile Networks. IEEE Transactions on Mobile Computing, 21(6), 2406-2419.
132. Cicconett, Claudio, Zhang, Wei, Li, Jing, Wang, Hong, & Chen, Jun. (2022). Edge Computing Resource Scheduling with Prediction-Based Agent Selection. ACM Transactions on Edge Computing, 3(1), 1-15.
133. Luo, Jie, Li, Xin, Wang, Xiaoli, Zhang, Wei, & Chen, Lei. (2022). QoE-Based Resource Scheduling for Edge Computing with Joint Allocation of Communication and Computation Resources. IEEE Transactions on Cloud Computing, 10(3), 785-798.
134. .Elgendy, Ibrahim A., Zhang, Wei, Li, Jing, Wang, Hong, & Chen, Jun. (2022). Multiuser Resource Allocation and Computation Resource Scheduling with Data Security in IoT. IEEE Internet of Things Journal, 9(5), 4287-4300.
135. Chunlin, Li, Zhang, Wei, Li, Jing, Wang, Hong, & Chen, Jun. (2022). Joint Optimization of Remote Controlled Mobile Edge Computing System with TDMA Task Scheduling. IEEE Transactions on Vehicular Technology, 71(9), 8456-8468.
136. .Li, Shenghui, Zhang, Wei, Li, Jing, Wang, Xiaoyan, & Chen, Ming. (2022). Make Span-Constrained Task Resource Scheduling for Toy-Edge-Cloud Computing. IEEE Transactions on Industrial Informatics, 18(5), 3635-3647.
137. Yan, T., Lu, F., Wang, S., Wang, L., & Bi, H. (2022). A hybrid metaheuristic algorithm for the multi-objective location-routing problem in the early post-disaster stage. Journal of Industrial and Management Optimization. https://doi.org/10.3934/jimo.2022145
138. Behera, R. K., Patro, A., & Roy, D. S. (2022). A Resource-Aware Load Balancing Strategy for Real-Time, Cross-vertical IoT Applications. In S. Dehuri, B. S. Prasad Mishra, P. K. Mallick, & S. B. Cho (Eds.), Biologically Inspired Techniques in Many Criteria Decision Making (Vol. 271, Smart Innovation, Systems and Technologies). Springer, Singapore. https://doi.org/10.1007/978-981-16-8739-6_2
139. Wang, M., Wang, J.-S., Song, H.-M., Zhang, M., Zhang, X.-Y., Zheng, Y., & Zhu, J.-H. (2022). Hybrid multi-objective Harris Hawk optimization algorithm based on elite non-dominated sorting and grid index mechanism. Advances in Engineering Software, 172, 103218. https://doi.org/10.1016/j.advengsoft.2022.103218
140. Shrestha, A., Chuprat, S., & Mukherjee, N. (2020). Hybrid Heuristic Load Balancing Algorithm For Resource Allocation In Cloud Computing. https://doi.org/10.36227/techrxiv.12991340
141. Maliszewski, K., Quiané-Ruiz, J.-A., Traub, J., & Markl, V. (2022). What Is the Price for Joining Securely? Benchmarking Equi-Joins in Trusted Execution Environments. 15, 659-672.

https://doi.org/10.14778/3494124.3494146

142. Mishra, S. K., Sahoo, B., & Parida, P. P. (2020). Load balancing in cloud computing: A big picture. Journal of King Saud University - Computer and Information Sciences, 32(2), 149-158. https://doi.org/10.1016/j.jksuci.2018.01.003
143. Faustina, J. M., Pavithra, B., Suchitra, S., & Subbulakshmi, P. (2019). Load Balancing in Cloud Environment using Self-Governing Agent. In Proceedings of the 2019 3rd International Conference on Electronics, Communication and Aerospace Technology (ICECA), Coimbatore, India, June 12-14, 2019 (pp. 480-483).
144. Goyal, S., Bhushan, S., Kumar, Y., Rana, A. u. H. S., Bhutta, M. R., Ijaz, M. F., & Son, Y. (2021). An Optimized Framework for Energy-Resource Allocation on Cloud Environment based on Whale Optimization Algorithm. Sensors, 21, 1583. https://doi.org/10.3390/s21051583
145. Nehra, P., & Nagaraju, A. (2019). Sustainable Energy Consumption Modeling for Cloud Data Centers. In 2019 IEEE 5th International Conference for Convergence in Technology (I2CT) (pp. 1-4). https://doi.org/10.1109/I2CT45611.2019.9033927
146. Diouani, S., & Medromi, H. (2019). How Energy Consumption in the Cloud Data Center is calculated. In Proceedings of the 2019 International Conference of Computer Science and Renewable Energies (ICCSRE), Agadir, Morocco, July 22-24, 2019 (pp. 1-10).
147. Daid, R., Kumar, Y., Hu, Y.-C., & Chen, W.-L. (2021). An effective scheduling in data centres for efficient CPU usage and service level agreement fulfilment using machine learning. Connection Science, 33(4), 954-974. https://doi.org/10.1080/09540091.2021.1926929
148. Alarifi, A., et al. (2020). Energy-Efficient Hybrid Framework for Green Cloud Computing. IEEE Access, 8, 115356-115369. https://doi.org/10.1109/ACCESS.2020.3002184
149. Kozakiewicz, A., & Lis, A. (2021). Energy Efficiency in Cloud Computing: Exploring the Intellectual Structure of the Research Field and Its Research Fronts with Direct Citation Analysis. Energies, 14, 7036. https://doi.org/10.3390/en14217036
150. Alarifi, A., Dubey, K., Amoon, M., Altameem, T., Abd El-Samie, F., Altameem, A., Sharma, S., & Nasr, A. (2020). Energy-Efficient Hybrid Framework for Green Cloud Computing. IEEE Access, 1-1. https://doi.org/10.1109/ACCESS.2020.3002184
151. Liu, X., Wu, J., Sha, G., & Liu, S. (2020). Virtual Machine Consolidation with Minimization of Migration Thrashing for Cloud Data Centers. Mathematical Problems in Engineering, 2020, Article ID 7848232, 13 pages. https://doi.org/10.1155/2020/7848232
152. Feng, C., Adnan, M., Ahmad, A., Ullah, A., Khan, H. U., & Ullah Khan, H. (2020). Towards Energy-Efficient Framework for IoT Big Data Healthcare Solutions. Scientific Programming, 2020, Article ID 7063681, 9 pages. https://doi.org/10.1155/2020/7063681.

List Of Appendix

1. **System Architecture**

Edge Layer: Comprises IoT devices and edge servers that collect and preprocess data locally. These devices perform initial data filtering and aggregation to reduce data volume before transmission.

Cloud Layer: Consists of cloud servers that handle intensive data processing, storage, and advanced analytics. The cloud provides scalable computing resources and centralized management.

2. **Components of the Hybrid Model**

Data Collection and Preprocessing: IoT devices collect raw data and perform preliminary processing to remove noise and extract relevant features.

Load Balancer: A central component that dynamically distributes workloads between edge and cloud resources. It considers factors such as latency, computational power, and current load.

Communication Module: Ensures secure and efficient data transmission between edge devices and the cloud using protocols like MQTT, HTTPS, or custom APIs.

Resource Manager: Monitors resource usage and availability on both edge and cloud, adjusting allocations to optimize performance and cost.

3. Load Balancing Algorithm

The load balancing algorithm is designed to optimize resource utilization and ensure low-latency processing. It includes the following steps:

1.Data Classification: Classify incoming data based on processing requirements (e.g., real-time vs. batch processing).

2.Resource Evaluation: Evaluate available resources on edge devices and cloud servers, considering current load and computational capacity.

3.Task Allocation: Allocate tasks to edge or cloud based on resource availability and data classification. Real-time tasks are prioritized for edge processing to minimize latency, while batch tasks can be offloaded to the cloud.

4.Dynamic Adjustment: Continuously monitor system performance and adjust task allocations as needed to balance the load and optimize resource usage.