Simulation of effective task scheduling algorithm for cloud environments

MSc Research Project
Cloud Computing

Aarthy Ramaswamy
Student ID: X20188838

School of Computing
National College of Ireland

Supervisor: Divyaa Manimaran Elango

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Aarthy Ramaswamy		
Student ID:	X20188838		
Programme:	Cloud Computing	**Year:**	2021-22
Module:	Research Project		
Supervisor:	Divyaa Manimaran Elango		
Submission Due Date:	January 31 2022		
Project Title:	Simulation of effective task scheduling algorithm for cloud environments		
Word Count:	6732	**Page count**	23

Signature: Aarthy Ramaswamy…………………………………………………………………………

Date: 31/1/2022……………………………………………………………………………………

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

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

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

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

Simulation of effective task scheduling algorithm for cloud environments

Aarthy Ramaswamy
X20188838

Abstract

Energy costs are a significant expenditure in Cloud Data Centers (CDCs), and several academics are attempting to lower them on a variety of levels. Energy cost reduction and efficiency improvement may be accomplished at the DC level as well as at the server level, where job scheduling occurs. Not only does task scheduling prioritise energy efficiency, but it also prioritises resource allocation in a timely way by optimising energy use. Blockchain is one of the new technologies that have emerged in this period. In terms of security and storage, blockchain technology has been combined with the cloud. This article presents a heuristic method to cloud work scheduling through blockchain. The effort is being done to optimise characteristics such as wait time, execution time, and Service Level Agreement (SLA), which will ultimately result in a decrease in energy usage and cost. The purpose of this study is to determine the advantages of using blockchain in the cloud to schedule work and to investigate its feasibility.

1 Introduction

In recent years, as communication technology has advanced, the Internet's size has resulted in an increase in the speed and capability of processing jobs over the Internet. How to manage massive volumes of data and deliver comfortable and quick services to customers is a tough topic that must be addressed as the Internet develops. Cloud computing was born out of this environment. Cloud computing offers end users with access to computer infrastructure and applications as a service. The cloud computing model's application services have complicated scheduling, composition, configuration, and deployment criteria (Agarwal et,al, 2015). Even with multiple-condition requirements, it is capable of balancing the scheduling of virtual services in response to temporal changes. As a result, the need for improving cloud computing scheduling strategies in order to assure cloud data security and resilience is growing (Amalarethinam, Kavitha, 2017)

1.1 Background and Motivation

According to International Data Corporation (IDC), there are more than 8.5 million DCs functioning worldwide, using a significant amount of energy to power them. [41] According to polls done by [43] and [44], cloud firms now spend approximately $200 billion annually, the figure that is expected to reach above $260 billion in 2025 as shown in Figure 1. One of the most concerned area in the cloud is task scheduling, since it is affected by a variety of circumstances. Numerous techniques have been developed to optimize scheduling in order to enhance the

Quality of Service (QoS) and other characteristics. Security of data stored in cloud has been a source stopper in achieving an efficient and reliable real time algorithm. Through vast research done on this area of cloud task scheduling, Blockchain is proposed to be a heuristic approach to maintain security through its concepts of distributed architecture, cryptography and consensus algorithm. The previous research articles describe how researchers use the Cloudsim Simulator tool to create a virtual environment in which they may plan activities and assess their performance. [42] Typically, scheduling is done centrally and then the job is dispersed to other DCs based on network bandwidth costs. The suggested research proposes to plan the work utilizing the decentralized method and blockchain technology, which would effectively distribute resources while using less energy, thus lowering the cost of energy. The fog-edge design helps in reducing the time between receiving and transmitting tasks.

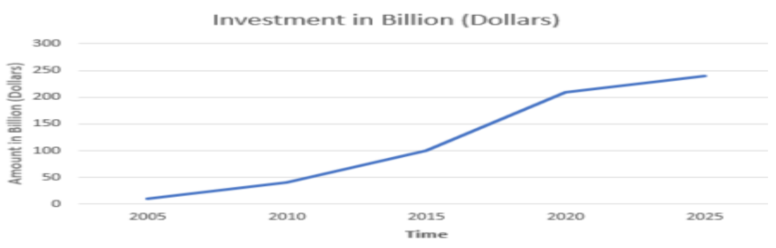


Figure 1: Investment in cloud computing

1.2 Research Question (RQ)

Will the Cloudsim simulation algorithms provide an efficient job scheduling in cloud environment in terms of time efficacy? Blockchain technology and the Fog-edge cloud simulator are a breakthrough in real-time deployment, demonstrate their usefulness, and highlight the need to cloud development.

1.3 Document Structure

The proposed study adds to the actual implementation of cloud task scheduling, including results assessment and its applicability in real time. The remainder of the thesis is organised as follows. Section 2 discusses the literature papers associated and their relevance to the suggested study issue. Section 3 describes the methodology adopted for the work plan. Section 4 of the thesis elaborates on the model design with a suitable figure. The designed model along with implementation is illustrated in Section 5. The next Section 6 summarises the predicted outcomes and provides an evaluation. The evaluated results are used to conclude the outcome of this study and future improvements or enhancements is suggested in the final Section 7

2 Literature Survey

Cloud technology is a primary driver of distributed systems, automation, and internet applications advancement. Cloud technology is a subset of internet-based computing that allows the delivery of infrastructure as a service (IaaS), platform as a service (PaaS), and software as a service (SaaS) through the internet (SaaS). In SaaS, the cloud hosting provides access to operating systems. PaaS is a model in which an implementation processes for applications is made available as a service to developers for the purpose of developing web-based applications. IaaS provides computational infrastructure in the form of Virtual Machines as a service to the requester (VM). Cloud computing is still in its infancy and faces several obstacles and challenges. Among the numerous concerns in cloud computing, scheduling plays a critical role in determining its proper implementation (Agarwal and Jain, 2014).

2.1 CLOUD TASK SCHEDULING CHALLENGES

We present the related work of job scheduling in a cloud computing context in this section. The author of the study (Buyya, Ranjan and Calheiros, 2009) provided a quick overview of the cloud Sim toolkit and its capabilities. In this article, we learnt how to simulate a job using various methodologies and scheduling policies. In his article (Jeyarani, Ram and Nagaveni, 2010), the author developed a method for job scheduling algorithms in cloud computing that is based on load balancing. This article discussed a two-tier job scheduling scheme based on load balancing. This method of work scheduling not only meets the requirements of the user, but also maximises resource use. This article discussed the construction of an efficient Meta-Scheduler based on Quality of Service (QoS) and a light weight Virtual Machine Scheduler based on Backfill approach for task dispatching. The authors of the research (Gan, Huang and Gao, 2010) provided an efficient work scheduling technique based on genetic simulated annealing. This takes into account the QoS needs for many types of jobs, such as completion time, bandwidth, cost, distance, and dependability. After selection, crossover, and mutation, annealing is used to increase the genetic algorithm's local search capability.

Hierarchical scheduling is discussed in this article (Rajavel and Mala, 2012), which enables the achievement of Service Level Agreements with a timely response from the service provider. In our suggested strategy, we accomplish a QoS measure such as response time by first executing high priority tasks and then spawning priority jobs from the remaining jobs using Task Scheduler. The author of (Cao, Wei and Gong, 2009) provided an efficient work scheduling technique based on Activity Based Costing (ABC). This method utilises cost drivers to give a priority level to each activity. ABC quantifies both the cost of the thing and its performance. (Yang et al., 2008) Described a transaction-intensive cost-constrained cloud Work flow scheduling technique. The algorithm prioritises execution cost and execution time. Under specific user-defined dates, the algorithm reduces the cost. Our suggested technique is mostly dependent on Virtual Machines' computing capacity. In the paper [26], a new VM Load Balancing Algorithm is described: Weighted Active Monitoring Load Balancing Algorithm Using CloudSim Tools.

The proposed Priority-based Performance-Improving Algorithm (PPIA) in [2] is to be implemented under the following assumptions:

- The executable programmes are a collection of indivisible tasks, sometimes referred to as meta-tasks.
- Tasks on each Resource's expected execution time is known in advance.
- Consistency of resources is assumed.
- Scheduling is to be done statically.
- The task's priority should be related with the user's priority.
- The total number of meta-tasks and resources available in the cloud computing environment are known.

At any given moment, a single job must be done in each resource in the sequence in which they were allocated. PPIA, the proposed scheduling approach, takes user priority into account. Users that use cloud services on a regular basis might benefit from improved service. The duties of the average user are given considerable importance. Other tasks are classified as having a regular priority. The supplied meta-task collection should be sorted into two categories: high-priority tasks and low-priority activities. The first step is to schedule work from the high priority task group using the Min-Min algorithm. Then, using the Max-Min algorithm, schedule low-priority activities. This approach, like the Min-Min algorithm, calculates the average time required to complete priority activities. In comparison to the Min-Min algorithm, it minimises its make span and enhances resource consumption rate.

They present a scheme for task allocation optimization in [4], entitled Privacy-aware Fog-enhanced Blockchain-assisted Scheduling Scheme (PF-BTS). This technique made use of BC miners to execute a sequence of ACO cycles in order to effectively and reliably assign internet VRs to do serves a purpose by applications (e.g. IoT), and thus obtain the shortest possible task computation time on cloud VRs. Furthermore, we recommend that fog elements be used to govern and control interaction between both the IoT layer, the BC system, and the Cloud layer. While PF-BTS protects the privacy of end systems and actions, it enables the server to recompense BC miners using decentralized applications. As a consequence, PF-BTS allows the construction of a confidentiality BaaS, augmenting reporting and data analysis at the cloudlet stages.

Cloud Computing's primary concerns are resource management and work scheduling. As a result, cloud providers should offer services, create virtual machines (VMs), and develop scheduling criteria for assigning VMs to users' tasks [27]. Static scheduling enables the pre-fetching of essential data and the pipelining of tasks at various phases of execution. Static scheduling puts a lower overhead on the runtime system. In the case of dynamic scheduling, the job components/tasks are unknown in advance. Thus, the job's execution duration may be unknown, and task allocation occurs on the fly while the programme is executing [28]. Pricing in cloud computing refers to the income received by a service provider from an end user in return for their services [29].

[5] Introduces an improved static job scheduling technique. The improvement method operates on the same fundamental premise as our prior work in [30]. VM based on the ratio of its required

processing power to the total processing power of all VMs, with the exception that the enhancement algorithm also considers the price of the VMs during the allocation process, using Amazon EC2 and Google Pricing models [31, 32]. To test the enhancement algorithm's performance, a comparison study was conducted between the enhancement algorithm, the default first-come-first-serve (FCFS) method, and the previously implemented GA and PSO algorithms. The experimental findings demonstrate that the improvement algorithm beats other algorithms in terms of reducing the overall execution time of users' activities and lowering the total cost of performing all jobs on available resources.

The majority of methods for scheduling time-sensitive jobs on cluster, grid, and cloud platforms need complete knowledge of the system status. Typically, their scheduling algorithms are centralised in nature and thus have inherent scalability constraints. [33] Emphasize how this precludes them from being used to plan multi-task applications in situations with millions of nodes and provide a lightweight alternative. Cloud computing is infinitely scalable. This results in the fact that cloud services may change dynamically. The static scheduling approach is not the ideal solution in this circumstance. Additionally, all cloud-based applications will be utilised by a significant number of people. And, consequently, different users have varying needs for QoS. Thus, a difficulty for cloud-based workflow systems is how to accommodate diverse QoS needs for different users. However, the majority of existing scheduling algorithms are optimised for scheduling a single complicated workflow instance or for scheduling a single QoS constraint, rather than for scheduling multiple QoS constraints across multiple workflows. To address this issue, we must create new solutions. The new algorithms must be capable of scheduling various operations and adhering to a variety of QoS criteria. As a result, we suggest a decentralised approach that makes judgments based on local and partial data. The suggested system is a completely decentralised scheduler that efficiently handles huge numbers of jobs with deadlines. Users submit scheduling requests specifying the application's attributes, such as the number and duration of tasks, the deadline, and the amount of memory and disc space needed.

Typically, the majority of current scheduling systems and algorithms create a queue or simple array of jobs for allocation on each available machine/server independently. The scheduler's primary requirement is that the end users specify the shortest time possible to accomplish all submitted jobs. Typically, end users lack further tools for confirming that the created schedule is optimum. To address this issue somewhat, we suggest a block chain-based scheduler in which created schedules must be accepted by the cloud cluster's neighbour nodes and the ideal schedule for end users is determined by the cloud providers' internal consensus. As a result, end consumers are unable to get a better deal from alternative suppliers.

2.2 BLOCKCHAIN IN CLOUD

Requirements include all of the qualities and attributes that a Blockchain system should possess, as well as the constraints that the system must adhere to in order to function successfully and efficiently. As a result, they have an effect on the entire operation of the Blockchain system and might have a greater impact on its design. The following are some of the non-functional characteristics of the blockchain network [7].

Openness: Due to the compatible nature of the nodes, the blockchain is capable of using and exchanging information throughout a transaction.

Concurrency: As nodes process concurrently, blockchain performance improves.

Scalability: Blockchains are scalable due to the addition and deletion of additional nodes. Scalability is primarily concerned with three parameters:

- Distribution transaction processing rate:

– Dimensions: – Lag or manageability:

Fault tolerance: The fault-tolerant characteristic of the Blockchain network ensures that any failure at any node is visible to all other nodes in the network, allowing the network to continue operating normally in the event of a failure.

Transparency: Every transaction on the Blockchain is visible to every node in the network.

Security: To safeguard data, the Blockchain network employs robust cryptographic methods such as SHA-255.

Quality of Service (QoS) on the Blockchain network is determined by response time and dependability, the time necessary to complete a transaction, and the commitment to offer the required services.

Failure Management: There must be a method in place that ensures the robustness of the Blockchain network and identifies the root cause of failure. It may propose how to automatically recover from a failure.

To tackle these issues, this article presents an integrity fair scheduling paradigm for cloud-fog-edge systems. It starts by establishing a decentralised trust structure via the use of blockchain systems in order to solve the problem of provider reliability. After discussing Berger's economic growth idea, the paper proposes the concept and measuring method of among whole. When examining software product situations, we separate planning into two layers. The work makes three significant contributions[9]. [34] Proposed the benefits of blockchain technology in cloud computing for lowering the internet's energy consumption. The author provides a concise explanation of how the blockchain, consensus, and smart contracts work. Additionally, a novel approach to task scheduling was implemented. [35] Utilized this decentralised technology to reduce the cost of energy usage while scheduling tasks. The author calculated the energy cost using the mining concept. The authors did not use Proof of Work (PoW) in this approach because the tasks were assigned to the DCs using First Come First Serve Scheduling.

This paper [12] discusses blockchain technology and conducts a survey of the blockchain by analysing generic technology and research trends, as well as discussing a solution for safely using bitcoin and future research areas. By comprehending the blockchain security trend, we can aid in the development of future blockchain technology.

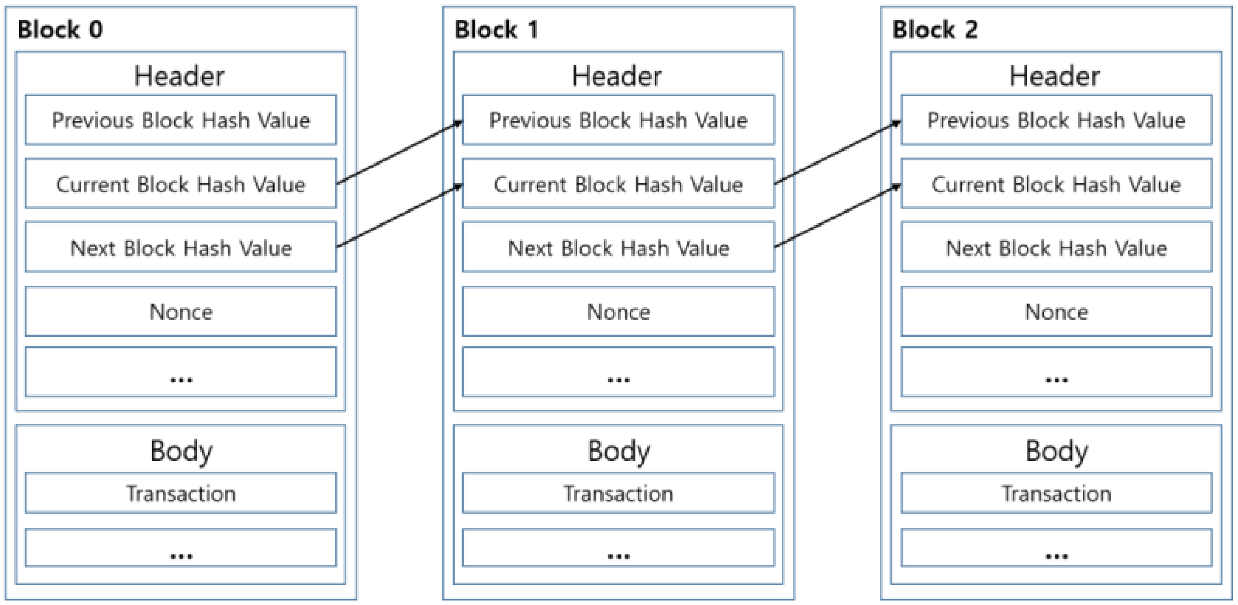


Figure 2: Blockchain connection structure.

A blockchain is a technology that enables all members to maintain a ledger including all transaction data and to update their ledgers in real time to ensure the integrity of the system when a new transaction occurs. Since the development of the Internet and encryption technologies enabled all participants to independently check the validity of a transaction, the single point of failure associated with reliance on an approved third party has been eliminated.

The blockchain includes broker-free (peer-to-peer) characteristics, which eliminates excessive costs associated with peer-to-peer transactions that are not authorised by a third party. Because widespread ownership of transaction information makes hacking more difficult, security costs are reduced, transactions are automatically authorised and documented by widespread involvement, and promptness is ensured. Additionally, the system may be readily built, linked, and extended through open source, and transaction data can be made publicly accessible to increase transparency and save regulatory costs [36]. Although the block does not include the hash value of the preceding block, it is included as a precaution (Figure 2) [37].

Due to the fact that the hash values saved in each peer in the block are impacted by the previous blocks' values, it is very difficult to falsify and modify the registered data. Although data modification is theoretically achievable if 51% of peers are hacked simultaneously, the attack scenario is practically very challenging. Both public, key-based verification and a decryptable hash function are utilised to secure the blockchain. The electronic signature method ECDSA (Elliptic Curve Digital Signature Algorithm), which validates the digital signature issued during a transaction between persons, is used to establish that the transaction data has not been changed.

2.3 FOG/EDGE COMPUTING (FEC) SIMULATION IN CLOUD

It's worth noting that FC devices are not always located at the network's edge, but rather near it. In comparison, edge devices are often located near the network's perimeter and are frequently the initial point of contact for IoT end devices. FC and EC devices are both near to IoT end-devices, while EC devices are often closer. Fog computing and edge computing have been used interchangeably in several studies. Some see FC as a subset of the EC and micro data centre (MDC) paradigms for the Internet of Things (IoT) [39].

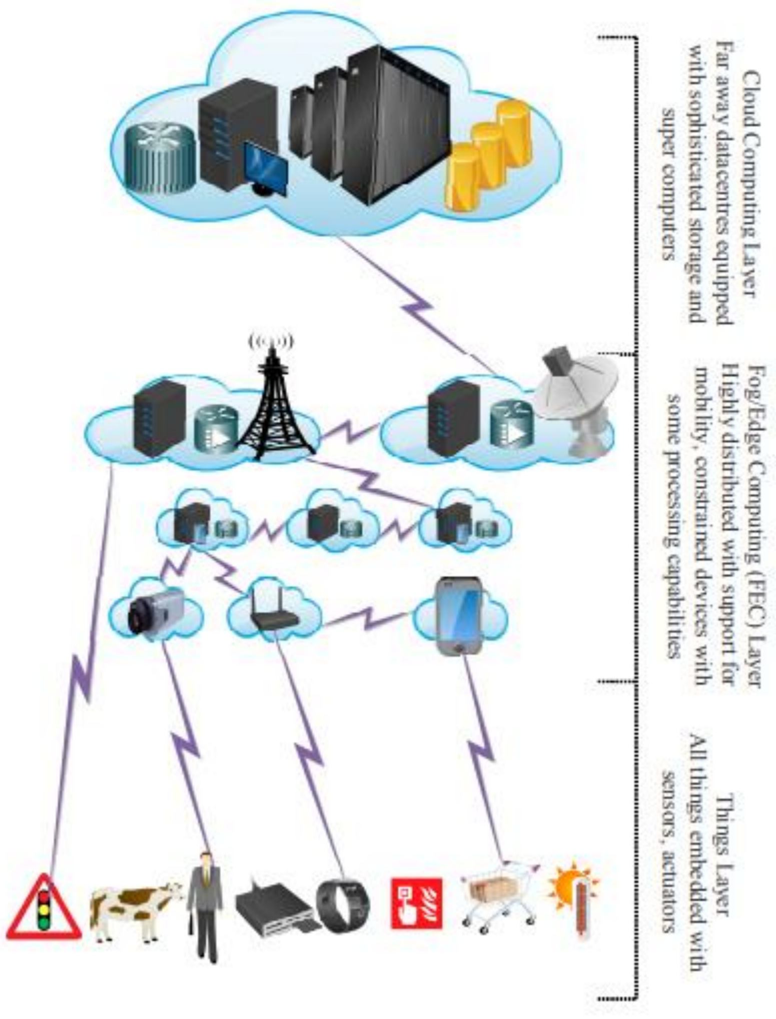

Cloud Computing Layer
Far away datacentres equipped with sophisticated storage and super computers
Fog/Edge Computing (FEC) Layer
Highly distributed with support for mobility, constrained devices with some processing capabilities
Things Layer
All things embedded with sensors, actuators

Figure 3: In the Cloud-to-Things paradigm, the design and interface of FECs are critical.

The proliferation of IoT applications may result in increased competition for resources and increased delay [38]. In essence, EC has a higher resource contention than FC owing to its closeness to IoT end-devices. Additionally, EC is more focused on the domain of things, while fog computing is more focused on the infrastructure domain. FEC is founded on a number of pillars, including security, scalability, openness, autonomy, dependability, agility, hierarchical structure, and programmability, all of which are inherent in both FC and EC. As such, the rationale for combining the FC and EC is based on their differences.

1) They both use a virtualized infrastructure as a service platform and support multi-tenancy of applications at the network's edge.

2) They both complement the cloud's functionality and are situated between end users and data centres.

3) They may both be co-located physically with access points, roadside devices, base stations, routers, switches, and gateways.

4) They are both primarily implemented wirelessly and enable low latency, low jitter, and in-system cognition.

5) They both distribute computing resources geographically in order to reduce the burden on the cloud.

FC and EC can coexist and overlap functionally within the CPS [40]. Fig. 3 illustrates the FEC IoT model with its many domains.

3 Research Methodology

Today's Cloud environments are facing fast growth and usage in huge amounts due to the growing data and IoT applications. Task scheduling helps users to effectively use the cloud resources without any wastage of energy, resource, time and achieve QoS. Having seen drastic usage, the phenomenon of scheduling and allocating resources to the tasks and jobs in these public/private clouds still remains a source stopper for efficient utilization of resources. To overcome the shortcomings of the literatures discussed above, a well analyzed method is devised. Implementing blockchain technology for task scheduling in the cloud with fog and edge simulators addresses the challenges of security, efficiency, and reduced latency. Cloud simulator is used to implement the algorithm with its two different approaches and compare their results for conclusion. Every component used in the work is explained in details below.

3.1 Blockchain Technology

The distributed decentralized network communicated between its nodes or each element of the network without any regulatory or supervisory body. This creates a sense of secure and reliable communication of the data. The peer-to-peer movement of information secures it as the chain grows. The structure of blockchain is designed in a suitable way to enhance this property. A common blockchain includes a unique Id, timestamp, and the hash of the previous block in the network as shown in Figure 2. The aspect of adding more blocks to the chain builds a increased unbreakable encryption to the data communicated within the blockchain. The consensus mechanism remains the deciding factor for every new addition of members to the network. Here we implement the Proof of Work consensus to validate the blockchain which is allowed to handle the data transmission from cloud networks to the data centers. Implying this mechanism provides reliable communication network without any data leakage, energy wastage or time delay due to network related issues in other communication channels. Thus a blockchain is to be designed and implemented in the real time cloud environments where the algorithm is to be implemented.

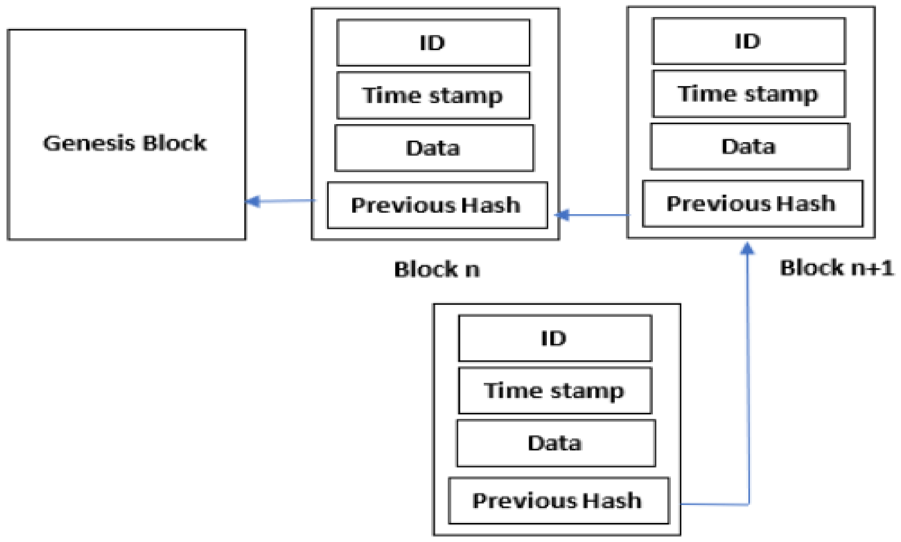


Figure 4: The Blockchain network

3.2 Fog – edge Simulation

IoT devices are wide spread across the globe and they contact datacenters located in specified areas every milliseconds, this leads to network trafficking across the cloud network. the tasks sent and received from these IoT devices are our priority consideration for they are huge in number. Controlling and regulating these communication trafficking can reduce a huge load on cloud network and datacenters. To reduce distance between the data devices and DCs, the concept of Fog/Edge computing is implied in the research work. These are computer devices installed at the edge of the network to contact the devices immediately and receive task instructions from them to send to the cloud as need arises. The concept of Fog-Edge computing reduces latency, process time delay, handling of huge data at datacenters and lastly lowers energy consumption. This is suggested from the research of numerous papers such as (Xu, Hao, Zhang and Sun, 2019) which shows more than 20% decreased time of processing the task communication. The model to implement this design is illustrated from the Figure 5.

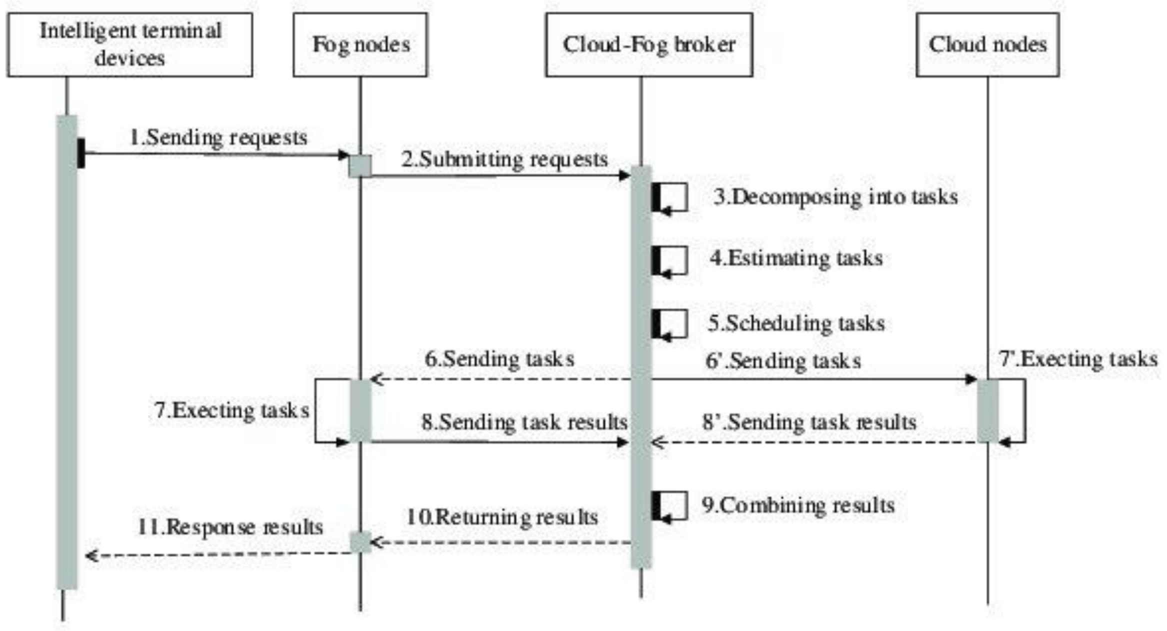


Figure 5: Model of the Fog-Cloud Architecture.

3.3 CloudSim

CloudSim is a cloud computing simulating tool that is used to construct a computing environments simulated environment. [42] illustrates the modular structure of CloudSim. It facilitates processes such as scheduling, work plans, virtual machine transfer, packet forwarding, and connectivity across multiple data centre elements, including servers, virtual machines, the database servers, the trader, and cloud resources.The primary advantage of the CloudSim technology is its adaptability and efficiency. Numerous researchers use CloudSim to perform experiments and analyse the outcomes. Cloudlets, Datacenter Characteristics, Datacenter Broker, Network Topology, and Virtual Machines are the CloudSim components that are relevant to the proposed work.

1. Cloudlets: Cloudlets are used to specify the jobs that are done inside a data centre. The cloudlets' input, output, and task length parameters are as follows.
2. DataCenter Characteristics: This class contains data on the DC's resources.
3. Datacenter Broker: This class serves as a go-between for the cloud user and the cloud provider. This class is responsible for enforcing the QoS criteria.

4. Hosts: This class is in charge of allocating physical resources such as the CPU, VM, and Memory.
5. Virtual Machine: This class is in charge of providing resources to virtual machines depending on their capabilities.

6. Network Topology: In the cloudsim model, this class contains information about network configuration.

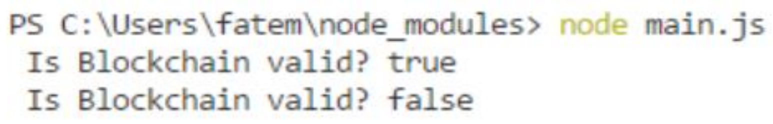
4 Implementation

The researched techniques are implemented for the experimental verification of their efficiency and results. The set up includes developing a blockchain with limited size to schedules tasks in cloud for lower energy estimate. The real time experiment of these task scheduling and cloud environment set up is practically beyond research bandwidth and experimentation, thereby a simulated environment where we can implement similar cloud data center like set up and run the tasks to understand their energy estimate and process time is taken into consideration. Cloudsim forms a vital part of this simulated environment as it is proved to be efficient in task scheduling algorithms and their experiment. Thereby an elaborated Cloudsim setup is also explained in the subsections to perform the experiment.

4.1 Blockchain setup

In this modelling we have utilized a basic blockchain with few blocks to communicate the data transfer. To construct the Blockchain network, we utilise Visual Studio code. Assume we've constructed a blockchain network that holds data from three geographically dispersed DCs. We have created parameters such as TimeStamp, Index, Hash from previous block, and Data for the purpose of storing the DC's entire load. We start with a genesis block and then add two more blocks that provide a random integer that determines the DC's load.

Attempt to verify the blockchain network by tampering with the data in block 1. If the blockchain network has not been compromised, it returns true; otherwise, it returns false. Even if the hash of the block is recalculated, the blockchain network is still invalid.

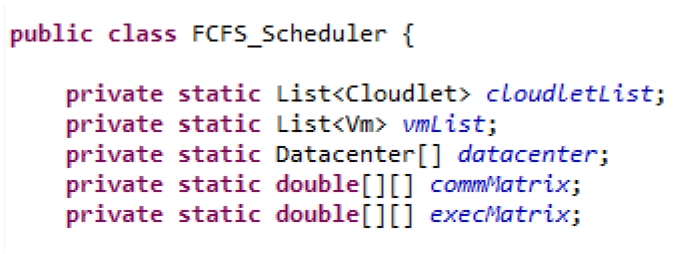


```
PS C:\Users\fatem\node_modules> node main.js
 Is Blockchain valid? true
 Is Blockchain valid? false
```

Figure 6: Blockchain validation

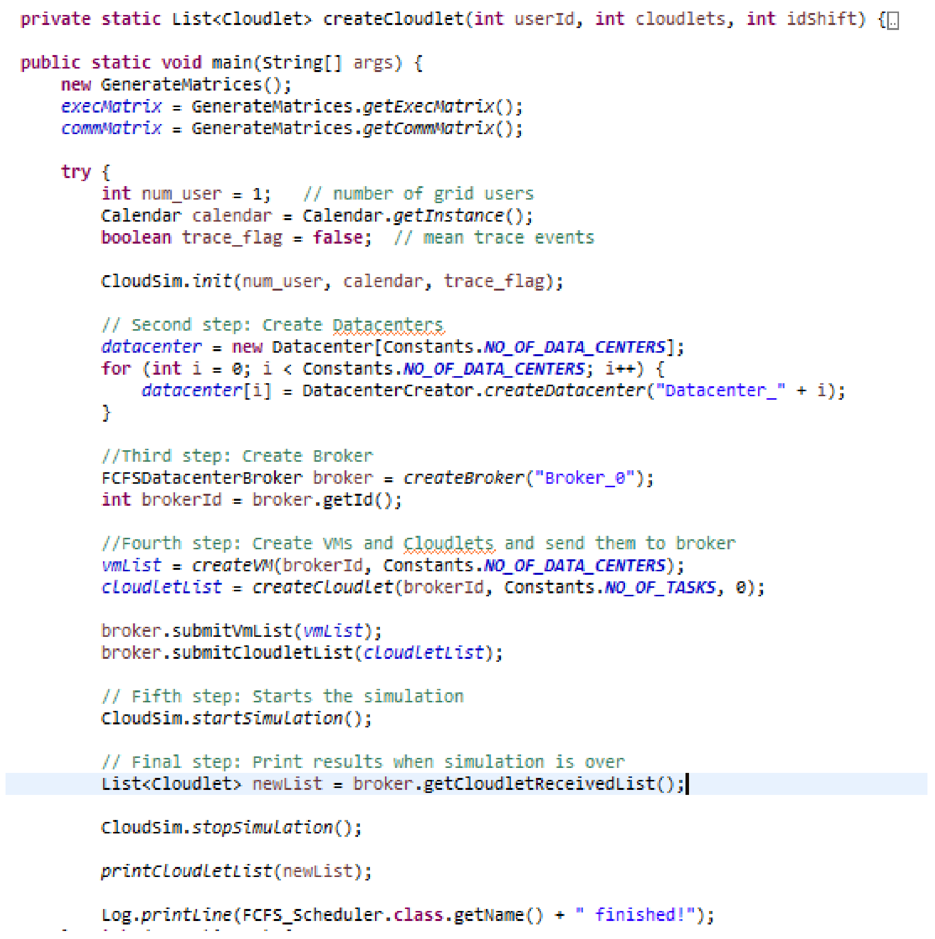
4.2 FCFS and SJF algorithms in Cloudsim

CloudSim 3.0.3 was utilised to assess the method in the suggested research study. This simulator enables the testing of different techniques for job scheduling, load balancing, virtual machine allocation, and other cloud computing-related concerns. The advantage of this simulator is that errors may be identified prior to implementation by evaluating the work in a simulated environment rather than during real-time execution. The simulation tool uses default algorithms to allocated cloudlets for the tasks in data centers, these include FCFS (First come first serve) scheduling algorithm and SJF (short job first) algorithm. The structure of FCFS scheduler and its entities are shown in Figure 7. Initially, all of the jobs are communicated to the datacenter as cloudlets. The controller, which pushes the jobs to the datacenter, connects with the broker. The dc has several hosts, each of which will be simulated as a separate VM with its own unique identifier. Tasks will be assigned to those VMs according to a set of scheduling policies. The implemented FCFS algorithm code is shown in Figure 8.



```java
public class FCFS_Scheduler {

    private static List<Cloudlet> cloudletList;
    private static List<Vm> vmList;
    private static Datacenter[] datacenter;
    private static double[][] commMatrix;
    private static double[][] execMatrix;
```

Figure 7: FSFC Entities

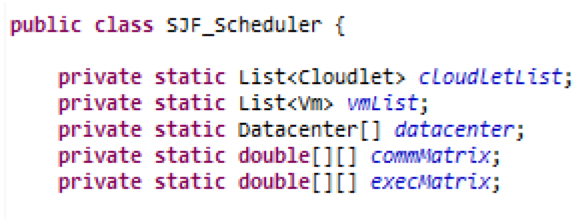


```java
private static List<Cloudlet> createCloudlet(int userId, int cloudlets, int idShift) {

public static void main(String[] args) {
    new GenerateMatrices();
    execMatrix = GenerateMatrices.getExecMatrix();
    commMatrix = GenerateMatrices.getCommMatrix();

    try {
        int num_user = 1;   // number of grid users
        Calendar calendar = Calendar.getInstance();
        boolean trace_flag = false;  // mean trace events

        CloudSim.init(num_user, calendar, trace_flag);

        // Second step: Create Datacenters
        datacenter = new Datacenter[Constants.NO_OF_DATA_CENTERS];
        for (int i = 0; i < Constants.NO_OF_DATA_CENTERS; i++) {
            datacenter[i] = DatacenterCreator.createDatacenter("Datacenter_" + i);
        }

        //Third step: Create Broker
        FCFSDatacenterBroker broker = createBroker("Broker_0");
        int brokerId = broker.getId();

        //Fourth step: Create VMs and Cloudlets and send them to broker
        vmList = createVM(brokerId, Constants.NO_OF_DATA_CENTERS);
        cloudletList = createCloudlet(brokerId, Constants.NO_OF_TASKS, 0);

        broker.submitVmList(vmList);
        broker.submitCloudletList(cloudletList);

        // Fifth step: Starts the simulation
        CloudSim.startSimulation();

        // Final step: Print results when simulation is over
        List<Cloudlet> newList = broker.getCloudletReceivedList();

        CloudSim.stopSimulation();

        printCloudletList(newList);

        Log.printLine(FCFS_Scheduler.class.getName() + " finished!");
```

Figure 8: FCFS Setup

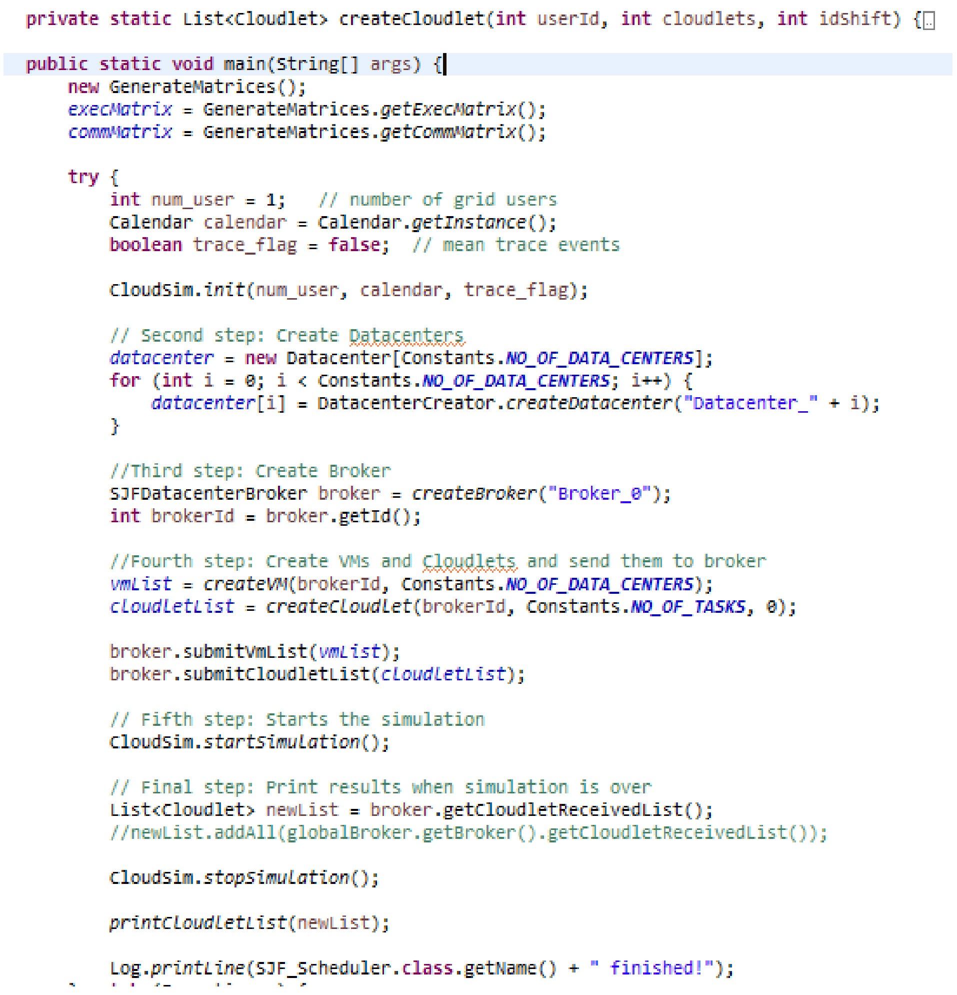
The suggested experiment was conducted using a Windows 10 platform powered by an Intel i7 core CPU. The experiment's settings included a Data Center, a Virtual Machine, a Host, and Cloudlets.

Another Cloudsim algorithm widely used to lower the wait time of scheduler is SJF (short job first). In this algorithm the list of tasks are sorted based on their process length and allocation is done sequentially. The experimental set up of SJF is Figure 9 and Figure 10. This algorithm is conventionally used in cloud schedulers to minimize wait time and achieve efficiency.



```
public class SJF_Scheduler {

    private static List<Cloudlet> cloudletList;
    private static List<Vm> vmList;
    private static Datacenter[] datacenter;
    private static double[][] commMatrix;
    private static double[][] execMatrix;
```

Figure 9: SJF Entities

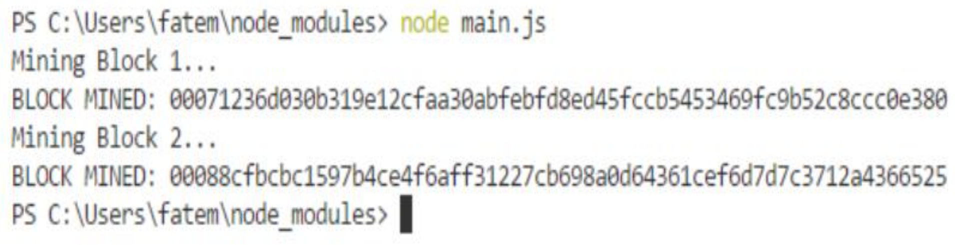


```
private static List<Cloudlet> createCloudlet(int userId, int cloudlets, int idShift) {

public static void main(String[] args) {
    new GenerateMatrices();
    execMatrix = GenerateMatrices.getExecMatrix();
    commMatrix = GenerateMatrices.getCommMatrix();

    try {
        int num_user = 1;   // number of grid users
        Calendar calendar = Calendar.getInstance();
        boolean trace_flag = false;  // mean trace events

        CloudSim.init(num_user, calendar, trace_flag);

        // Second step: Create Datacenters
        datacenter = new Datacenter[Constants.NO_OF_DATA_CENTERS];
        for (int i = 0; i < Constants.NO_OF_DATA_CENTERS; i++) {
            datacenter[i] = DatacenterCreator.createDatacenter("Datacenter_" + i);
        }

        //Third step: Create Broker
        SJFDatacenterBroker broker = createBroker("Broker_0");
        int brokerId = broker.getId();

        //Fourth step: Create VMs and Cloudlets and send them to broker
        vmList = createVM(brokerId, Constants.NO_OF_DATA_CENTERS);
        cloudletList = createCloudlet(brokerId, Constants.NO_OF_TASKS, 0);

        broker.submitVmList(vmList);
        broker.submitCloudletList(cloudletList);

        // Fifth step: Starts the simulation
        CloudSim.startSimulation();

        // Final step: Print results when simulation is over
        List<Cloudlet> newList = broker.getCloudletReceivedList();
        //newList.addAll(globalBroker.getBroker().getCloudletReceivedList());

        CloudSim.stopSimulation();

        printCloudletList(newList);

        Log.printLine(SJF_Scheduler.class.getName() + " finished!");
```

Figure 10: SJF Setup

6 Evaluation

The Goal of this study is to experimentally simulate the task scheduling using the Cloudsim 3.0.3 tool with its conventional default algorithms and derive at a best result in terms of efficiency and efficacy. The blockchain implemented has reduced the security threats and increased reliability of data held withing its network. A sample of executed blocks is shown in the Figure 11. The following experiments are used to examine the findings. Additionally, a comprehensive conclusion and recommendations for future study are included.

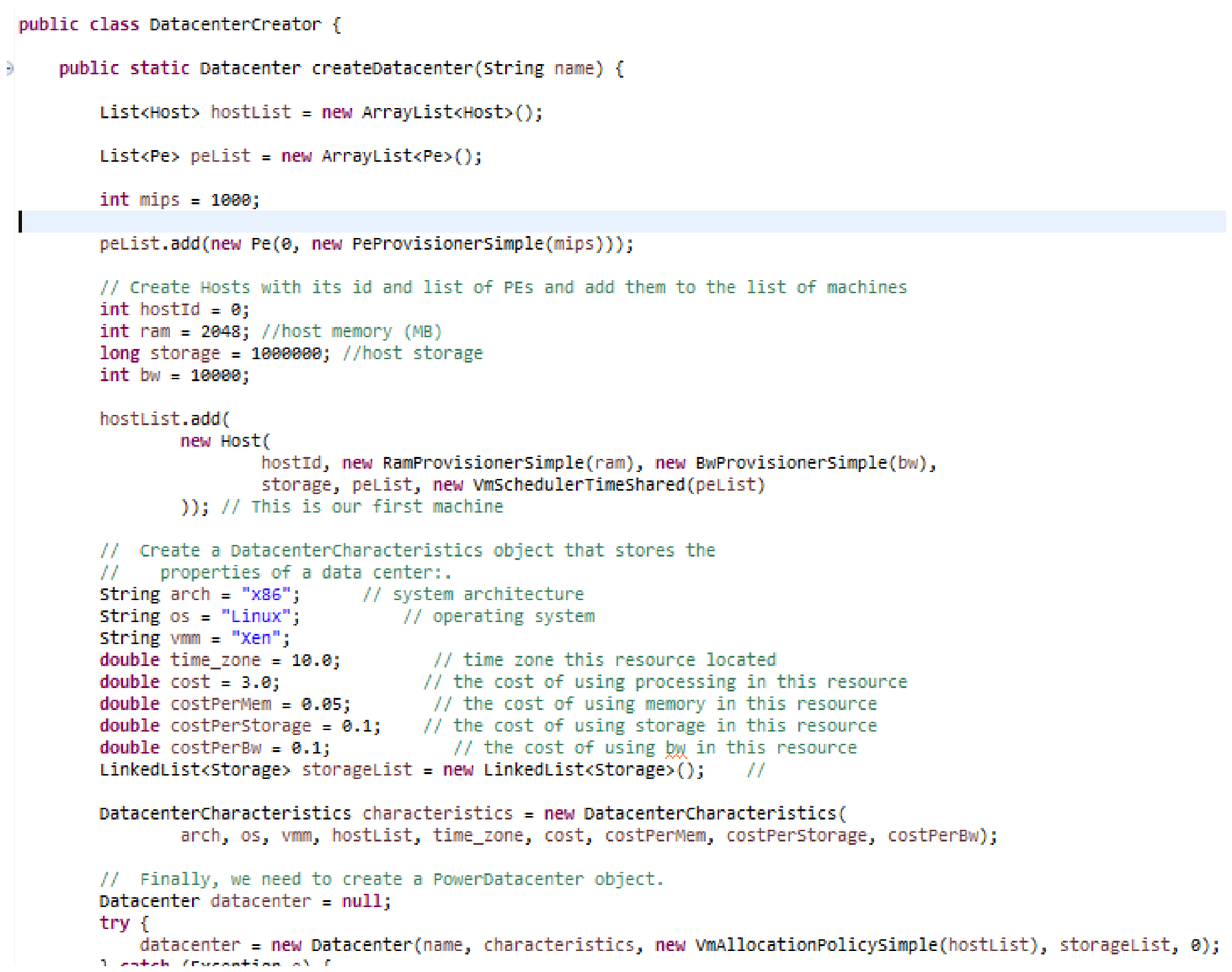


```
PS C:\Users\fatem\node_modules> node main.js
Mining Block 1...
BLOCK MINED: 00071236d030b319e12cfaa30abfebfd8ed45fccb5453469fc9b52c8ccc0e380
Mining Block 2...
BLOCK MINED: 00088cfbcbc1597b4ce4f6aff31227cb698a0d64361cef6d7d7c3712a4366525
PS C:\Users\fatem\node_modules>
```

Figure 11: Blockchain validation

6.1 Experiment

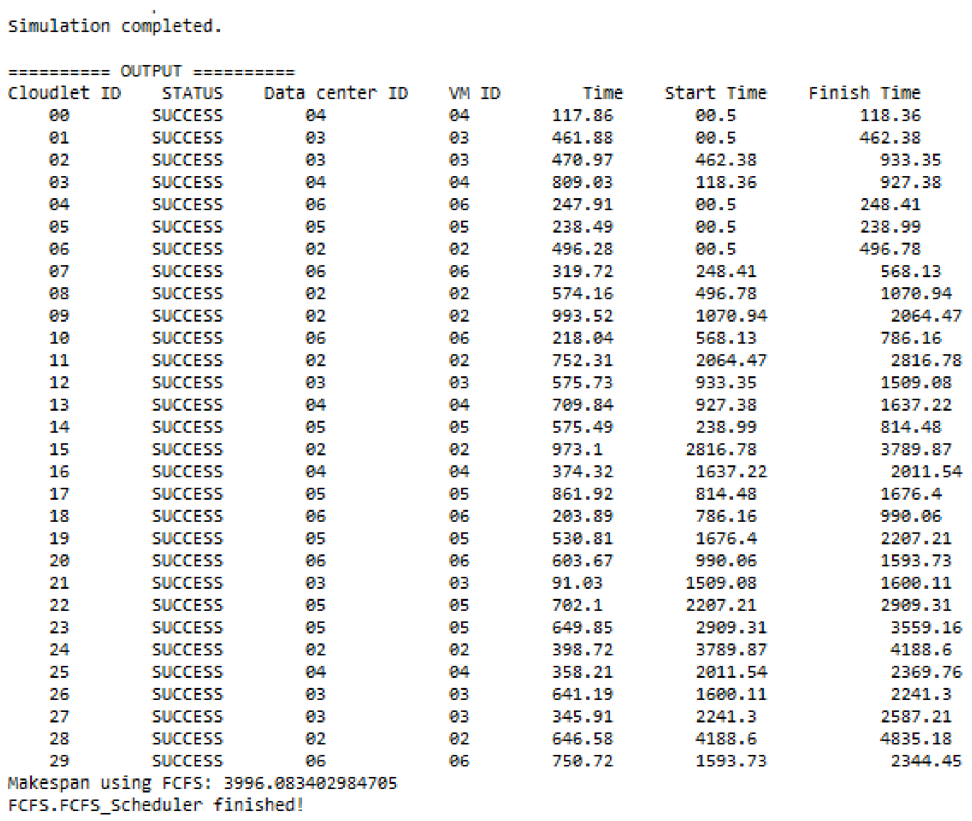
The Cloudsim scheduler is run to perform task scheduling using the SJF algorithm and FCFS for static activities with a pre-determined processing time. The workloads, cloudlet length etc are pre defined and scheduler is run for these values. The cloudlet length is not taken into account by the FCFS algorithm. It starts with the first task (cloudlet) and then moves on to the next. However, the SJF algorithm sorts all cloudlets sent to the dc broker first. The experiment is carried out with 5 datacenters, and minimum of 30 cloudlets set up. The data center is created as shown in Figure 12.



```java
public class DatacenterCreator {

    public static Datacenter createDatacenter(String name) {

        List<Host> hostList = new ArrayList<Host>();

        List<Pe> peList = new ArrayList<Pe>();

        int mips = 1000;

        peList.add(new Pe(0, new PeProvisionerSimple(mips)));

        // Create Hosts with its id and list of PEs and add them to the list of machines
        int hostId = 0;
        int ram = 2048; //host memory (MB)
        long storage = 1000000; //host storage
        int bw = 10000;

        hostList.add(
                new Host(
                        hostId, new RamProvisionerSimple(ram), new BwProvisionerSimple(bw),
                        storage, peList, new VmSchedulerTimeShared(peList)
                )); // This is our first machine

        //  Create a DatacenterCharacteristics object that stores the
        //    properties of a data center:.
        String arch = "x86";      // system architecture
        String os = "Linux";          // operating system
        String vmm = "Xen";
        double time_zone = 10.0;         // time zone this resource located
        double cost = 3.0;              // the cost of using processing in this resource
        double costPerMem = 0.05;        // the cost of using memory in this resource
        double costPerStorage = 0.1;    // the cost of using storage in this resource
        double costPerBw = 0.1;            // the cost of using bw in this resource
        LinkedList<Storage> storageList = new LinkedList<Storage>();    //

        DatacenterCharacteristics characteristics = new DatacenterCharacteristics(
                arch, os, vmm, hostList, time_zone, cost, costPerMem, costPerStorage, costPerBw);

        //  Finally, we need to create a PowerDatacenter object.
        Datacenter datacenter = null;
        try {
            datacenter = new Datacenter(name, characteristics, new VmAllocationPolicySimple(hostList), storageList, 0);
        } catch (Exception e) {
```

Figure 12: Datacenter creation

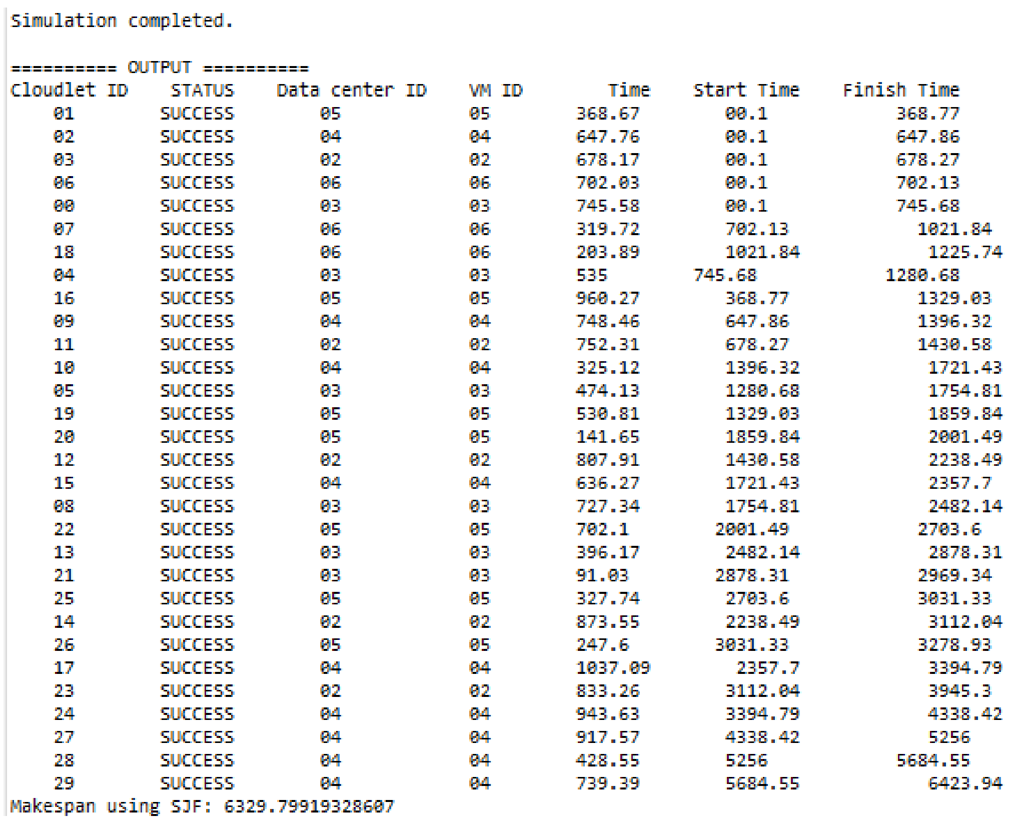
The experiment is repeated several times to derive at a consistent result. The efficiency is checked with the make span time output from the simulation task. Figure 13 depicts the FCFS scheduler results



```
Simulation completed.

========== OUTPUT ==========
Cloudlet ID    STATUS    Data center ID     VM ID       Time     Start Time     Finish Time
    00        SUCCESS        04              04        117.86      00.5            118.36
    01        SUCCESS        03              03        461.88      00.5            462.38
    02        SUCCESS        03              03        470.97      462.38            933.35
    03        SUCCESS        04              04        809.03      118.36            927.38
    04        SUCCESS        06              06        247.91      00.5            248.41
    05        SUCCESS        05              05        238.49      00.5            238.99
    06        SUCCESS        02              02        496.28      00.5            496.78
    07        SUCCESS        06              06        319.72      248.41            568.13
    08        SUCCESS        02              02        574.16      496.78            1070.94
    09        SUCCESS        02              02        993.52      1070.94            2064.47
    10        SUCCESS        06              06        218.04      568.13            786.16
    11        SUCCESS        02              02        752.31      2064.47            2816.78
    12        SUCCESS        03              03        575.73      933.35            1509.08
    13        SUCCESS        04              04        709.84      927.38            1637.22
    14        SUCCESS        05              05        575.49      238.99            814.48
    15        SUCCESS        02              02        973.1      2816.78            3789.87
    16        SUCCESS        04              04        374.32      1637.22            2011.54
    17        SUCCESS        05              05        861.92      814.48            1676.4
    18        SUCCESS        06              06        203.89      786.16            990.06
    19        SUCCESS        05              05        530.81      1676.4            2207.21
    20        SUCCESS        06              06        603.67      990.06            1593.73
    21        SUCCESS        03              03        91.03      1509.08            1600.11
    22        SUCCESS        05              05        702.1      2207.21            2909.31
    23        SUCCESS        05              05        649.85      2909.31            3559.16
    24        SUCCESS        02              02        398.72      3789.87            4188.6
    25        SUCCESS        04              04        358.21      2011.54            2369.76
    26        SUCCESS        03              03        641.19      1600.11            2241.3
    27        SUCCESS        03              03        345.91      2241.3            2587.21
    28        SUCCESS        02              02        646.58      4188.6            4835.18
    29        SUCCESS        06              06        750.72      1593.73            2344.45
Makespan using FCFS: 3996.083402984705
FCFS.FCFS_Scheduler finished!
```

Figure 13: FCFS Scheduler results

Repeating the same parameters for SJF scheduler provides a make span time of 6329 seconds as shown in Figure 14. Which is far beyond the FCFS scheduler.



```
Simulation completed.

========== OUTPUT ==========
Cloudlet ID    STATUS    Data center ID    VM ID       Time     Start Time     Finish Time
    01        SUCCESS         05            05        368.67      00.1          368.77
    02        SUCCESS         04            04        647.76      00.1          647.86
    03        SUCCESS         02            02        678.17      00.1          678.27
    06        SUCCESS         06            06        702.03      00.1          702.13
    00        SUCCESS         03            03        745.58      00.1          745.68
    07        SUCCESS         06            06        319.72      702.13          1021.84
    18        SUCCESS         06            06        203.89      1021.84          1225.74
    04        SUCCESS         03            03        535      745.68         1280.68
    16        SUCCESS         05            05        960.27      368.77          1329.03
    09        SUCCESS         04            04        748.46      647.86          1396.32
    11        SUCCESS         02            02        752.31      678.27          1430.58
    10        SUCCESS         04            04        325.12      1396.32          1721.43
    05        SUCCESS         03            03        474.13      1280.68          1754.81
    19        SUCCESS         05            05        530.81      1329.03          1859.84
    20        SUCCESS         05            05        141.65      1859.84          2001.49
    12        SUCCESS         02            02        807.91      1430.58          2238.49
    15        SUCCESS         04            04        636.27      1721.43          2357.7
    08        SUCCESS         03            03        727.34      1754.81          2482.14
    22        SUCCESS         05            05        702.1      2001.49         2703.6
    13        SUCCESS         03            03        396.17      2482.14          2878.31
    21        SUCCESS         03            03        91.03      2878.31         2969.34
    25        SUCCESS         05            05        327.74      2703.6          3031.33
    14        SUCCESS         02            02        873.55      2238.49          3112.04
    26        SUCCESS         05            05        247.6      3031.33         3278.93
    17        SUCCESS         04            04        1037.09      2357.7          3394.79
    23        SUCCESS         02            02        833.26      3112.04          3945.3
    24        SUCCESS         04            04        943.63      3394.79          4338.42
    27        SUCCESS         04            04        917.57      4338.42          5256
    28        SUCCESS         04            04        428.55      5256         5684.55
    29        SUCCESS         04            04        739.39      5684.55          6423.94
Makespan using SJF: 6329.79919328607
```

Figure 14 SJF Scheduler results.

Thus the evaluation of the results show a varied difference in scheduling time between the algorithms by nearly 45% reduced time consumption in the FCFS scheduling. These results when implemented in real time along with Fog-edge computing at the end of network would further reduce the time and energy consumption. Bring the computation and data transfer closer to the IoT devices using a edge computed is seen to reduce the network communication time by 18%. A comparison of results obtained by running the Clousim scheduler for varied values provide a clear depiction of the results as in Figure 15.

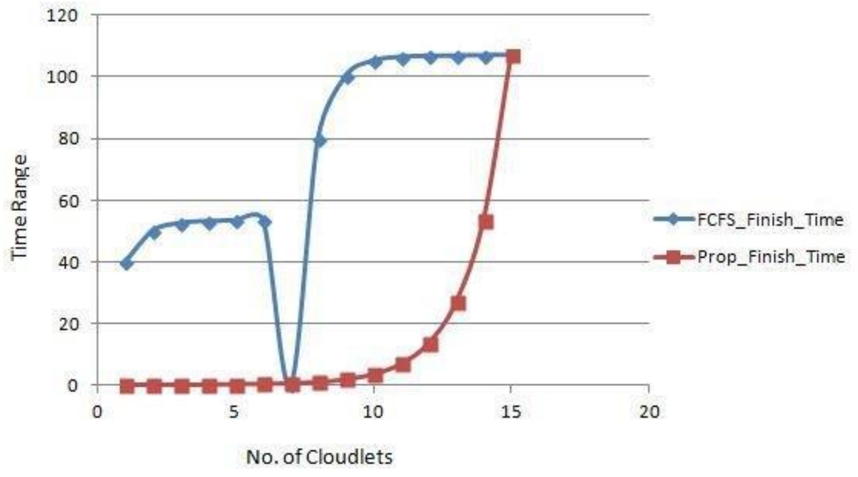


Figure 15: FCFS Vs SJF algorithm

In cloud computing, work scheduling is determined by a variety of variables. Utilizing blockchain technology in conjunction with cloud computing for storage is a viable solution. However, calculating the load on DCs at predetermined intervals would waste a significant amount of energy and time. Additionally, blockchain mining takes a few minutes. As a result, blockchain technology is more secure.

7 Conclusion and Future Work

The primary objective of the study was to increase the Security and reliability of task scheduling by using blockchain technology. The CloudSim simulator aided in the implementation of the FCFS algorithm, which tries to decrease the amount of time spent waiting for other processes. The overall implementation of blockchain and FCFS algorithm was successful in achieving a reduced time and energy consumption along with data leakage prevention. Scheduling jobs in the sequential order possible might ultimately decrease total completion time but would likely fail to minimise SLA breaches.

As a result, scheduling future work utilising a deadline-priority approach would be beneficial for avoiding SLA violations. Additionally, we presented a blockchain architecture for job scheduling by connecting DCs through a blockchain network. The duties are carried out on the least burdened DCs. The load of the DC is defined by this block, which must be increased at regular intervals. The suggested paper's limitation was to construct a smart contract that would assist in selecting the DC with the least load and executing the tasks on it. However, we were unable to draught the contract.

The study paper's next work will be to assess the suggested work in a real-time context in order to determine that job scheduling utilising blockchain produces superior outcomes. Additionally, to construct smart contracts that would assist in allocating the job to the DCs with the lowest load.

Reference

[1].Agarwal, D. and Jain, S., 2014. Efficient Optimal Algorithm of Task Scheduling in Cloud Computing Environment. International Journal of Computer Trends and Technology, 9(7), pp.344-349.

[2].Amalarethinam, D. G. and Kavitha, S. (2017). Priority based performance improved algorithm for meta-task scheduling in cloud environment, Computing and Communications Technologies (ICCCT), 2017 2nd International Conference on, IEEE, pp. 69–73.DOI: 10.1109/ICCCT2.2017.7972250, Conference Location: Chennai, India

[3].Bakarman, A. and Almezeini, N., Factors influencing students' acceptance of e-learning platforms in primary and secondary schools in saudi arabia.

[4].Baniata, H., Anaqreh, A. and Kertesz, A., 2021. PF-BTS: A Privacy-Aware Fog-enhanced Blockchain-assisted task scheduling. Information Processing & Management, 58(1), p.102393.

[5].Belgacem, A., Beghdad-Bey, K. and Nacer, H., 2018, April. Task scheduling in cloud computing environment: A comprehensive analysis. In International Conference on Computer Science and its Applications (pp. 14-26). Springer, Cham.

[6].Choudhary, V., Kacker, S., Choudhury, T. and Vashisht, V., 2012. An approach to improve task scheduling in a decentralized cloud computing environment. International Journal of Computer Technology and Applications, 3(1), pp.312-316.

[7].Gupta, A., Siddiqui, S.T., Alam, S. and Shuaib, M., 2019. Cloud computing security using blockchain. Journal of Emerging Technologies and Innovative Research (JETIR), 6(6), pp.791-794.

[8].Keivani, A. and Tapamo, J., 2019. Task Scheduling in Cloud Computing: A Review. 2019 International Conference on Advances in Big Data, Computing and Data Communication Systems (icABCD),.

[9].Li, W., Cao, S., Hu, K., Cao, J. and Buyya, R., 2021. Blockchain-Enhanced Fair Task Scheduling for Cloud-Fog-Edge Coordination Environments: Model and Algorithm. Security and Communication Networks, 2021, pp.1-18.

[10].Liu, K., Chen, J., Yang, Y. and Jin, H., 2008. A throughput maximization strategy for scheduling transaction-intensive workflows on SwinDeW-G. Concurrency and Computation: Practice and Experience, 20(15), pp.1807-1820.

[11].Lokhandwala, F.A., 2018. A Heuristic Approach to Improve Task Scheduling in Cloud Computing using Blockchain technology (Doctoral dissertation, Dublin, National College of Ireland).

[12].Park, J.H. and Park, J.H., 2017. Blockchain security in cloud computing: Use cases, challenges, and solutions. Symmetry, 9(8), p.164.

[13].Razaque, A., Vennapusa, N.R., Soni, N. and Janapati, G.S., 2016, April. Task scheduling in cloud computing. In 2016 IEEE long island systems, applications and technology conference (LISAT) (pp. 1-5). IEEE.

[14].Selvarani, S. and Sadhasivam, G.S., 2010, December. Improved cost-based algorithm for task scheduling in cloud computing. In 2010 IEEE International Conference on Computational Intelligence and Computing Research (pp. 1-5). IEEE.

[15].Stamford (2017). Gartner says worldwide public cloud services market to grow 18 percent in 2017. URL: https://www.gartner.com/newsroom/id/3616417

[16].Wilczyński, A. and Kołodziej, J., 2020. Modelling and simulation of security-aware task scheduling in cloud computing based on Blockchain technology. Simulation Modelling Practice and Theory, 99, p.102038.

[17].Yang, T., Guo, Q., Tai, X., Sun, H., Zhang, B., Zhao, W. and Lin, C. (2017). Applying blockchain technology to decentralized operation in future energy internet, Energy Internet and Energy System Integration (EI2), 2017 IEEE Conference on, IEEE, pp. 1–5. DOI: 10.1109/EI2.2017.8244418, Conference Location: Beijing, China.

[18].Zhang, P. and Zhou, M., 2018. Dynamic Cloud Task Scheduling Based on a Two-Stage Strategy. IEEE Transactions on Automation Science and Engineering, 15(2), pp.772-783.

[19].Zhu, H., Wang, Y., Hei, X., Ji, W. and Zhang, L., 2018. A Blockchain-Based Decentralized Cloud Resource Scheduling Architecture. 2018 International Conference on Networking and Network Applications (NaNA).

[20] Buyya, R., Ranjan, R. and Calheiros, R.N., 2009, June. Modeling and simulation of scalable Cloud computing environments and the CloudSim toolkit: Challenges and opportunities. In *2009 international conference on high performance computing & simulation* (pp. 1-11). IEEE.

[21] Jayarani, R., Sadhasivam, S. and Nagaveni, N., 2009, October. Design and implementation of an efficient two-level scheduler for cloud computing environment. In *2009 International Conference on Advances in Recent Technologies in Communication and Computing* (pp. 884-886). IEEE.

[22] Gan, G.N., Huang, T.L. and Gao, S., 2010, October. Genetic simulated annealing algorithm for task scheduling based on cloud computing environment. In *2010 International Conference on Intelligent Computing and Integrated Systems* (pp. 60-63). IEEE.

[23] Rajavel, R. and Mala, T., 2012. Achieving service level agreement in cloud environment using job prioritization in hierarchical scheduling. In *Proceedings of the International*

Conference on Information Systems Design and Intelligent Applications 2012 (INDIA 2012) held in Visakhapatnam, India, January 2012 (pp. 547-554). Springer, Berlin, Heidelberg.

[24] Cao, Q., Wei, Z.B. and Gong, W.M., 2009, June. An optimized algorithm for task scheduling based on activity based costing in cloud computing. In *2009 3rd international conference on bioinformatics and biomedical engineering* (pp. 1-3). IEEE.

[25] Yang, Y., Liu, K., Chen, J., Liu, X., Yuan, D. and Jin, H., 2008, December. An algorithm in SwinDeW-C for scheduling transaction-intensive cost-constrained cloud workflows. In *2008 IEEE Fourth International Conference on eScience* (pp. 374-375). IEEE.

[26] Jasmin James, Dr. Bhupendra Verma "Efficient Vm Load Balancin Algorithim For A Cloud Computing Environment " In Proceeding of International Journal on Computer Science and Engineering (IJCSE) Vol. 4 No. 09, Sep 2012

[27] Y. Yang, et al., " An Algorithm in SwinDeW-C for Scheduling Transaction- Intensive Cost-Constrained Cloud Workflows," Proc. of 4th IEEE International Conference on e-Science, Indianapolis, USA, PP. 374-375, December 2008.

[28] Y.Chawla, M.Bhonsle, "A Study on Scheduling Methods in Cloud Computing", International Journal of Emerging Trends & Technology in Computer Science, Volume 1, Issue 3, PP. 12-17, September – October 2012.

[29] M.Al-Roomi, S.Al-Ebrahim, S.Buqrais and I.Ahmad, "Cloud Computing Pricing Models: A Survey", Vol.6, No.5 (2013), pp.93-106, International Journal of Grid and Distributed Computing.

[30] Elhossiny Ibrahim, Nirmeen A. El- Bahnasawy, Fatma A. Omara, "Job Scheduling based on Harmonization Between The requested and Available Processing Power in The Cloud Computing Environment", IJCA, Volume 125 – No.13, September 2015.

[31] (2014, 5-11-2014 11:00 pm). Amazon EC2. Available:http://aws.amazon.com/ec2/.

[32] (2016, 1-11-2016 10:00 am). Google Cloud. Available: https://cloud.google.com/compute/pricing.

[33] I. Raicu, Y. Zhao, C. Dumitrescu, I. Foster, and M. Wilde, "Falkon: a Fast and Light-weight tasK executiON framework," in Proceedings of the ACM/IEEE Conference on Supercomputing, Nov. 2007, p. 1–12.

[34] Yang, T., Guo, Q., Tai, X., Sun, H., Zhang, B., Zhao, W. and Lin, C. (2017). Applying blockchain technology to decentralized operation in future energy internet, Energy Internet and Energy System Integration (EI2), 2017 IEEE Conference on, IEEE, pp. 1-5.

[35] Xu, C., Wang, K. and Guo, M. (2018). Intelligent resource management in blockchain-based cloud datacenters, IEEE Cloud Computing 4(6): 50-59.

[36] Kaskaloglu, K. Near zero Bitcoin transaction fees cannot last forever. In Proceedings of the International Conference on Digital Security and Forensics (DigitalSec2014), The Society of Digital Information and Wireless Communication, Ostrava, Czech Republic, 24–26 June 2014.

[37] Ziegeldorf, J.H.; Matzutt, R.; Henze, M.; Grossmann, F.; Wehrle, K. Secure and anonymous decentralized Bitcoin mixing. Future Gener. Comput. Syst. 2016.

[38] P. Hu, S. Dhelim, H. Ning, and T. Qiu, "Survey on Fog Computing: Architecture, Key Technologies, Applications and Open Issues," Journal of Network and Computer Applications, pp. 27-42, Nov. 2017.

[39] M. Aazam, and E. N. Huh, "Fog computing: The cloud-IOT/IoE middleware paradigm," IEEE Potentials, vol. 35, no. 3, pp. 40–44, May 2016.

[40] S. Chen, T. Zhang and W. Shi, "Fog Computing," IEEE Internet Computing, vol. 21, no. 2, pp. 4-6, Mar.-Apr. 2017.

[41] Mastelic, T., Oleksiak, A., Claussen, H., Brandic, I., Pierson, J.-M. and Vasilakos, A. V. (2015). Cloud computing: Survey on energy efficiency, Acm computing surveys (csur) 47(2): 33.

[42] Calheiros, R. N., Ranjan, R., Beloglazov, A., De Rose, C. A. and Buyya, R. (2011). Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms, Software: Practice and experience 41(1): 23–50.

[43] Columbus, L. (2017). Roundup of cloud computing forecasts, 2017. URL: https://www.forbes.com/sites/louiscolumbus/2017/04/29/roundup-of-cloudcomputing-forecasts-2017/31a0d0a531e8

[44] Stamford (2017). Gartner says worldwide public cloud services market to grow 18 percent in 2017. URL: https://www.gartner.com/newsroom/id/3616417