*sustainability*

MDPI

*Article*

# A Fog-Cluster Based Load-Balancing Technique

**Prabhdeep Singh [1,2], Rajbir Kaur [3], Junaid Rashid [4,*], Sapna Juneja [5], Gaurav Dhiman [1,6,7], Jungeun Kim [4,8,*] and Mariya Ouaissa [9]**

1 Department of Computer Science and Engineering, Graphic Era Deemed to be University, Dehradun 248002, India; ssingh.prabhdeep@gmail.com (P.S.); gdhiman0001@gmail.com (G.D.)
2 Department of Computer Science and Engineering, Punjabi University, Patiala 147001, India
3 Department of Electronics & Communication Engineering, Punjabi University, Patiala 147001, India; rajbir277@yahoo.in
4 Department of Computer Science and Engineering, Kongju National University, Cheonan 31080, Korea
5 Department of Computer Science, KIET Group of Institutions, Delhi NCR, Ghaziabad 201206, India; sapnajuneja1983@gmail.com
6 Department of Computer Science, Government Bikram College of Commerce, Patiala 147001, India
7 University Centre for Research and Development, Department of Computer Science and Engineering, Chandigarh University, Mohali 140413, India
8 Department of Software, Kongju National University, Cheonan 31080, Korea
9 Department of Computer Science, Moulay Ismail University, Marjane 2, BP: 298, Meknes 50050, Morocco; ouaissa.maria04@gmail.com
* Correspondence: junaidrashid062@gmail.com (J.R.); jekim@kongju.ac.kr (J.K.)

**Abstract:** The Internet of Things has recently been a popular topic of study for developing smart homes and smart cities. Most IoT applications are very sensitive to delays, and IoT sensors provide a constant stream of data. The cloud-based IoT services that were first employed suffer from increased latency and inefficient resource use. Fog computing is used to address these issues by moving cloud services closer to the edge in a small-scale, dispersed fashion. Fog computing is quickly gaining popularity as an effective paradigm for providing customers with real-time processing, platforms, and software services. Real-time applications may be supported at a reduced operating cost using an integrated fog-cloud environment that minimizes resources and reduces delays. Load balancing is a critical problem in fog computing because it ensures that the dynamic load is distributed evenly across all fog nodes, avoiding the situation where some nodes are overloaded while others are underloaded. Numerous algorithms have been proposed to accomplish this goal. In this paper, a framework was proposed that contains three subsystems named user subsystem, cloud subsystem, and fog subsystem. The goal of the proposed framework is to decrease bandwidth costs while providing load balancing at the same time. To optimize the use of all the resources in the fog sub-system, a Fog-Cluster-Based Load-Balancing approach along with a refresh period was proposed. The simulation results show that "Fog-Cluster-Based Load Balancing" decreases energy consumption, the number of Virtual Machines (VMs) migrations, and the number of shutdown hosts compared with existing algorithms for the proposed framework.

**Keywords:** load balancing; cloud computing; fog computing



**Citation:** Singh, P.; Kaur, R.; Rashid, J.; Juneja, S.; Dhiman, G.; Kim, J.; Ouaissa, M. A Fog-Cluster Based Load-Balancing Technique. *Sustainability* **2022**, *14*, 7961. https://doi.org/10.3390/su14137961

Academic Editor: Petra Poulová

Received: 1 April 2022
Accepted: 20 June 2022
Published: 29 June 2022



## 1. Introduction

Cloud computing relies on the internet to run huge applications and data servers for customers or end-users as a new technology. It refers to the control of workloads remotely in a data center through the Internet. On-site data centers are no longer required since cloud computing removes the need for hardware, software, and controls and manages them. The cloud provider's network allows for data to be replicated across many spare locations. In a catastrophe, it enables data backup, disaster recovery, and business continuity. Cloud computing services are provided to customers, and they are charged depending on the

amount of time spent using these services. Some programs are unable to function effectively on the cloud because of an inherent difficulty. Problems with bandwidth develop when data are not being transported to the cloud quickly enough. Due to the high volume of data that must be transmitted back to the edge nodes after processing in the cloud, delays cannot be allowed in critical scenarios such as smart healthcare. As a result, fog computing was developed as a solution to this problem.

A new identity for cloud computing, "fog computing", or "fogging", was coined by CISCO. The Internet of Things (IoT) network paradigm uses it to make it easier for scattered devices to send data wirelessly. Computing resources and application services are brought to the edge where data are created through fog computing, making them more accessible to end-users. Fog computing significantly reduces the quantity of data that must be transferred to the cloud for processing and analysis. As a result, quality of service and security are improved because of the reduced traffic on the cloud (i.e., the primary server).

Cloud data centers and end-users are in the midst of the fog computing virtual platform, which provides networking, processing, and storage capabilities near where data are created. These services serve as a bridge between cloud computing and fog computing. Only summarized information will be sent to the cloud, and bandwidth will be reduced to a considerable degree since most of the information is being processed locally in a foglet. As a result, packet loss and latency will be reduced. When it comes to performance, cloud computing is not replaced with fog computing; rather, both must co-exist to complement one other. Customers make requests to the fog layer or middleware for communication and storage reasons, which are processed on the fog server before being communicated back to customers. Requests requiring more computational power will be sent to the cloud.

To load the status of every fog node is a dynamic phenomenon, and therefore, the state of every fog node needs to be refreshed periodically. Then it needs to compute an optimal refresh period to optimize the load-balancing approach. Therefore, there is a need for a strategy to obtain the Optimal Refresh Period dynamically from time to time.

The fog nodes are very small and smart devices. They have very limited storage capacity. So, we need to make sure that these nodes are not over burdened with data. Data should be short-lived on these nodes and should be flushed/removed at a regular interval. However, a key concern in a distributed system is that if you remove data from one node, then there might be data inconsistency and inaccuracy. Therefore, an algorithm is needed that can effectively flush the data from fog nodes while making sure that this data flushing does not cause any data inconsistency.

The primary contribution of this paper is to provide the following:

- Firstly, an architecture of cloud-based fog-paradigm for real-time (delay-sensitive) applications was proposed. In this proposed architecture, a fog layer is designed to design a fog layer in such a manner that it can take advantage of the fog concept as well as reduce the delay for mission-critical applications.
- Secondly, a load-balancing algorithm for effective distribution of the task load to fog nodes placed in fog cluster. This algorithm covers the solution to the problem of determining Optimal Refresh Period.
- Finally, the data flushing algorithm effectively flushes the data from fog nodes.

Section 2 discusses the related work and the proposed framework explained in Section 3. The experimental results and the conclusion are explained in Sections 4 and 5.

## 2. Related Work

In ref. [1], a hybrid routing approach and a multiple mesh ring (MMR) topology architecture. As part of the load-balancing network design process, three stages are completed: leader selection, role determination, and scattered building. To begin, each master seeks out his nearby slaves to choose a coordinator for the network. Then calculate the number of scattered links needed for each master and sends the information to the coordinator. Finally, to build the sparse balanced link MMR, each chose master links to its related nodes, including slaves, intrabridges, and internal bridges. When an API request is received,

ref. [2] proposes task chain-based load balancing for microservices, which examines the system resource utilization of each service instance and selects target services for all tasks in this call of APIs. Data transfers between physical machines are reduced with the help of the proposed microservice, which speeds up API responses. The author concludes that Genetic Algorithm (GA) outperforms Particle Swarm Optimization (PSO), Simulated Annealing (SA), and Genetic Algorithm (GA) when task chain-based load balancing for microservices is adopted. Using TCLBM, the author performed service instance load balancing and minimized API response times by up to 10%. A new hybrid power amplifier load modulation (LM) design that employs three amplifiers coupled in phase via a quadrature couplerDoherty PA (DPA) and load-modulated balanced amplifier (LMBA) modes may be used at various power levels in conjunction with the well-aligned turning-on sequence [3]. High-linearity design compatibility is further enhanced by this unique hybrid LM mode, which widens the dynamic power range for efficiency improvement. Two stages of genetics are used to balance a load of virtual machine hosts (VMHs) in cloud computing, as this article shows [4]. Previous solutions often assume that this is a problem of optimizing work assignment and simply take into account current loads of VMHs; however, these methods may only acquire limited efficacy in actual systems without addressing the loads of VMHs after balancing. Genetic-based approaches are combined and provided. Virtual machine (VM) performance models are first generated from the parameters used to create them and their accompanying performance measurements in a cloud computing environment. Regression models based on gene expression programming (GEP) characterize virtual machine performance and are used to estimate the load of VMHs after a shift in workload. To further aid in moving and distributing VMs, GEP estimates the VMH load and uses this information to determine which VMs should be assigned to each of the available VMAs. An open-source dynamic load-balancing library is introduced in this study, and it incorporates effective literature-based scheduling techniques [5]. LB4OMP is a research framework that encourages and supports research programming for the benefit of multi-threaded applications.

Ref. [6] utilized the MPI library to develop an automated load-balancing solution for the discontinuous Galerkin time-domain (DGTD). There is a depicts the connection between computational burden (CPU time) and the number of DoFs in the hex and tetrahedral meshes, respectively, from numerical studies. To reduce CPU time discrepancies across subdomains, an automated iteration load-balancing approach was then used to optimize the placements of the interfaces using the diagram created in the first stage. Using a greedy method, a one-click solution is provided when the original model is divided, which is the only external action required. Ref. [7] presents an active charge modulation power amplifier (PA) design known as Hybrid Asymmetric Charge Modulated Balanced Amplifier (HALMA) (H-ALMBA). The LMBA Control Amplifier (CA) can be used as a carrier amplifier, while the Balanced Amplifier (BA) can be used as a peaking amplifier with different threshold settings for two sub-amplifiers BA1 and BA2. With the correct control of amplitude and phase and the cooperative alignment of the firing sequence of BA1 and BA2, the entire H-ALMBA can achieve better efficiency over a longer power interruption range, comparable to a Doherty PA.

Ref. [8] presented a model in which distributed systems store data in several locations to share the burden. An evaluation of the load-balancing performance of storage systems in which d distinct nodes store the same amount of items for each object is performed. The proposed model uses a fixed cumulative value to sample evenly at random from all of the load vectors for the objects. As a result, increasing the number of nodes in an n-node system can exponentially improve load balancing, as long as the number of nodes is o(log(n)). We show that using XOR objects of r instead of object copies improves load balancing in the same way as d. In such redundant systems, r multiplicatively reduces storage overhead. However, the material from nodes r must be unloaded to obtain an object. The load balance also increases by r, but additively. Ref. [9] proposed a dynamic load-balancing solution evaluated using the computational paradigm related

to the spatially related Cellular Automata. The major contribution of this research is the development of simple closed-form equations that enable the calculation of the best workload assignment dynamically, intending to assure a perfectly balanced workload distribution throughout the parallel execution. The MPI technology is used to develop an algorithm for balancing the execution of cellular automata based on these expressions. A concentrate on the use of multi-server load-balancing solutions for request migration. A disutility function is defined for each server and its response time is lowered as much as possible [10]. Server availability is also concentrated, which directly influences the server's processing capacity and, consequently, its usefulness. The author utilized the concept of variational inequality theory (VI) to tackle the issue and argue that the defined game has a set of Nash equilibrium solution set. An iterative proximal algorithm (IPA) is presented to calculate a Nash equilibrium solution. Ref. [11] proposed a load-balancing routing method for wireless mesh networks to balance the network's traffic. Load-balancing routers that meet QoT restrictions are selected using the principles of software-defined networking (SDN). Ref. [12] proposed a technique that utilized the concept of multiple metrics, which were used to monitor controllers' load, and an ideal target was selected based on its available resources. This resulted in better load-balancing performance. HESM chooses switches based on minimizing migration expenses and lowering extra migration expenditures. Additional advantages include the ability to migrate numerous controllers concurrently, which enhances the overall efficiency of switch migration. According to simulation data, HESM greatly outperforms current systems and lowers the migration cost when it comes to controller load-balancing.

Ref. [13] analyzed radio efficiency power efficiency and average network power savings in various network configurations to address the core challenges of time-spatial traffic intensity dynamics and renewable energy (RE). A network utility, such as the use of green energy and user association based on a group coordination strategy, is fairly maintained as a result of this effort. During periods of low arrivals, lightly loaded Base Stations (BS) can be turned off to further save energy. The proposed CoMP-based load-balancing algorithm increases energy efficiency compared with traditional location- and traffic-centric strategies by efficiently controlling the allocation of resources to new users. Ref. [14] proposed an inter-domain communication system (DOLPHIN) which is a tailored solution for various SDN controllers. It expands the programmability of wireless devices, such as the Internet of Things or vehicle networks, beyond the virtual switch components in intra and interdomain communication. According to extensive simulation findings, the traffic load is evenly dispersed over several connections linking distinct domains. Load balancing may considerably increase the flow completion times of various kinds of network traffic in data centers and 5G vehicle networks.

In ref. [15], security measures for the combined load balance and computation offloading (CO) approach were proposed for MEC systems. Algorithms to distribute MDUs across network switches are presented first. To protect data during transmission, an Advanced Encryption Standard (AES) cryptographic approach based on electrocardiogram (ECG) signal-based encryption and the decryption key is also provided as a security layer. It is also structured as an integrated balance, CO, and security model to save time and resources. Ref. [16] presented a Power-efficient and Load-balanced Online Flow Route architecture based on software-defined networking to accomplish the necessary tradeoff between power saving and load balancing. An optimization problem is defined that considers both power conservation and load balancing. A flow scheduling method is proposed to solve the NP-hard issue after it is NP-hard. A route updating technique is also proposed to allow flow scheduling induced path updates. Moreover, the experimental findings show that the suggested method is superior to other algorithms when it comes to power conservation and load balancing.

According to [17], and the work scheduling may be improved by using a combination of chimpanzee optimization (ChOA), marine predators (MPA), and the disruption operator. The key limitations of the newly created algorithm, known as CHMPAD, are its inability

to resist being sucked into local optima and to increase the original ChOA's exploitation capacity. CHMPAD's efficiency and applicability are shown by conducting experiments on both synthetic and actual workloads from the Parallel Workload Archive. To illustrate how the cost of computing resources can be reduced [18]. The analytical model uses a network of queues to determine the bare minimum amount of computing resources needed to ensure that service-level agreement is met. Ref. [19] analyzed the performance of Fog computing and the analytical model that the author developed. Discrete event simulator simulations are used to evaluate the analytical model. By analyzing network tasks and transferring them to physical machines over the network, ref. [20] utilize a profit function phase to enhance the quality of service (QoS). In ref. [21] the simulation of discrete events was employed to evaluate and cross-validate our analytical model. According to our analytical models, this model can reliably and effectively anticipate how many computing resources are needed for health data services to meet the response time requirements under varied workload situations. To avoid data starvation in a scalable fog architecture, ref. [22] suggested an effective scheduling method based on complicated event processing. The cost resources reduced and performance requirements are still being met. The First, the author provided an arrival–service model for FoT data traffic based on multilayer waiting for lines with finite-size intervals. The proposed policy was compared to first-in-first-out and multi-priority-discipline queue strategies with the help of a complete study of wait times and gaps in wait times [23–25]. Machine learning and Clustering based various methods are used for the text analysis [26], internet of things [27–33], and disease detection [34]. Under any given IoT workload, the mathematical model estimates the minimum number of fog nodes required to meet QoS requirements. Formulas for performance indicators such as response time, system loss rate, and average number of messages requested from the model.

## 3. Proposed Framework

The architectural view of the proposed framework is shown in Figure 1, which contains three subsystems: (1) User subsystem and (2) Fog subsystem and (3) Cloud subsystem. The user subsystem is responsible for collecting user data.

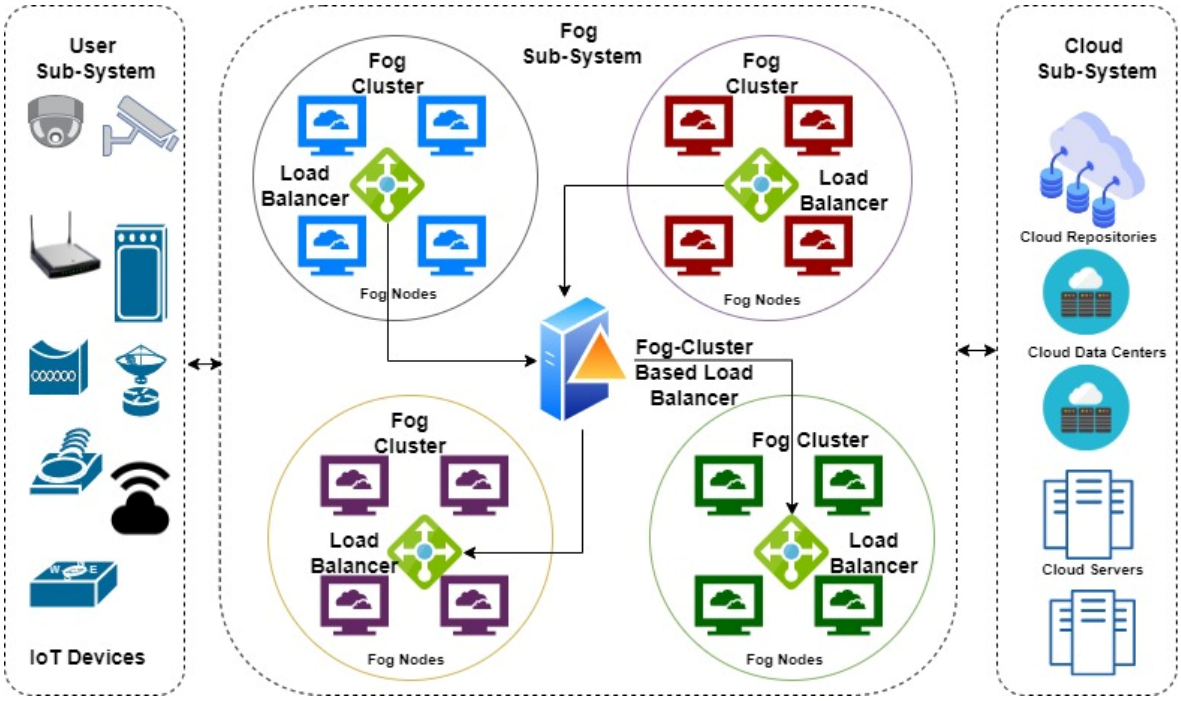


**Figure 1.** Proposed framework.

The fog subsystem processes the data, and finally, it is stored in cloud repositories placed in the cloud subsystem.

### *3.1. User Subsystem*

The user subsystem is a data generator system consisting of sensor nodes/IoT devices. The critical area of interest is to provide remote monitoring and data collection. The sensors are made as small and energy efficient as possible; they do not have the computing power to preprocess any data, and so send all collected data to the fog subsystem [35–37].

The fog subsystem contains fog cluster and fog storage that have the processing capability and storage capability, respectively, to organize the data generated with the various devices. It is responsible for collecting data from the user subsystem [38,39]. In this subsystem, every fog cluster contains multiple clusters where each cluster can have a certain number of hosts and associated VMs.

The fog cluster has a load-balancer program [40]; the load-balancer program takes care of scheduling jobs while keeping the load in mind. However, the clusters' state needs to be refreshed from time to time. The problem here is to determine a cluster-based load-balancing technique with an optimal refresh period [41–43]. Evaluation of the refresh period to have a refresh threshold is very important for optimizing load balancing using the fog clustering approach.

The controller in the Fog cluster-based approach needs to have such a mechanism for better performance. Since Quality of Service (QoS) is very important in fog computing, load balancing plays a crucial role in it [32–34].

Each Fog cluster is associated with a load balancer to schedule given jobs to different VMs. A load status, i.e., BUSY, WORKING, and FREE, is associated with each cluster. This will have an intuitive meaning, as BUSY reflects the fact that the VMs in the cluster is busy and they are working with the fullest capacity possible. In other words, the cluster is incapable of accommodating more jobs unless the jobs being processed are completed in this state. WORKING indicates the state of the cluster, which reflects that the VMs associated with the cluster are not free, but the load is normal. In other words, the cluster can accommodate more jobs until the state turns out to be BUSY. Whereas, FREE indicates that the cluster is idle [43–45] and it has many VMs that are presently not processing any jobs. All the clusters are controlled by a controller. The controller can take jobs and schedule them to various clusters based on the cluster's state. There might be several clusters made in the fog cluster. Each cluster can have a different number of VMs and a load balancer for each cluster. Each cluster in turn is linked to a controller which takes care of the proposed Optimal Refresh Period and load-balancing algorithms. The dynamic situations that prevail at runtime are considered to understand the number of clusters and the status of each cluster to make load-balancing decisions. Two algorithms are proposed: fog-cluster-based load-balancer algorithm and the optimal refresh period computation algorithm, respectively [46–48].

#### 3.1.1. Overall Strategy of Fog Cluster-Based Load Balancing (FCBLB)

Jobs arrive at the main controller. Since jobs come in large quantities, an iterative approach is applied to each job that arrives at the controller. Fog clusters are chosen as they are available [49], and then any one of the clusters is considered for verification. If the cluster is found with a state BUSY/WORKING, that cluster is not considered for allocation of jobs. It is skipped for the time being, and another cluster is considered. If the cluster is found to be FREE, it is considered to be the cluster to which the jobs are to be sent. After reaching a cluster, an associated load-balancer program will make further decisions. The controller also makes intelligent decisions on the REFRESH period of clusters to reflect the true state that improves load-balancing efficiency [50].

3.1.2. Fog Cluster-Based Load-Balancing (FCBLB) Algorithm

FCBLB Algorithm 1 is responsible for balancing the load when the jobs arrive at the main controller. The algorithm should be aware of the number of clusters and the number of jobs that arrive. It is supposed to take stock of the states of different Fog clusters. The states in turn reveal the VMs that are associated with hosts of clusters and their relative load with jobs assigned. The algorithm and decisions verify each cluster are made to send jobs to a specific cluster based on the load of the cluster then. The allocation of jobs to clusters is not a static or trivial thing, as it is very dynamic due to the arrival of jobs in fog from various users across the globe. FCBLB provides the necessary steps to deal with the load that arrives at the controller in the fog clustering architecture. These algorithm steps are to be executed iteratively for jobs arriving at the controller. Provided the number of clusters and the number of jobs that arrive at the controller, the algorithm has a search mechanism to find out candidate clusters and then make job allocation decisions at the cluster level based on the state of the given cluster. The variable cluster State holds one of the three states as BUSY, WORKING, and FREE and is not static. It does mean that the state of a cluster may change from time to time. Thus, it is essential to have a triggering state change. If there is no guidance for triggering the state change, it will affect load-balancing performance. The load-balancing algorithm will lead to mediocre performance if the states are not refreshed to reflect the new load dynamics. When the refresh period is less, it may have unnecessary overhead on the system. If the refresh period is greater, it may lead to inefficiency in load balancing. Therefore, the problem of the Optimal Refresh Period is considered. The following subsection deals with the algorithm required for the determination of the optimal refresh period.

3.1.3. Determination of Refresh Period

This section sheds light onto optimizing load balancing by determining the Optimal Refresh Period. When the Optimal Refresh Period is determined, the controller will have the states correctly reflected besides getting rid of overhead on the system. When the time interval is large, certain important aspects may surface and go unnoticed, besides being incapable of reflecting true load dynamics. Therefore, Optimal Refresh Period computation is a dynamic phenomenon that is provided in the Optimal Refresh Period Computation (ORP) Algorithm 2.

The ORP has important considerations for the refresh period. It is a dynamically adjusted value based on the computed load factor. The refresh period threshold Tr is determined at run-time to update the state of clusters. This algorithm is executed by the controller, where it must make use of states of clusters from time to time for load balancing. ORP reduces the burden on the system while leveraging efficiency in load balancing.

**Algorithm 1:** Fog cluster Based Load Balancing (FCBLB).

**Result:** Balanced load leading to efficiency
**Input:** Jobs, clusters
**Output:** Balanced load leading to efficiency

1. **if** PS=FREE | | PS=WORKING **then**
2. Allocate j to FP
3. job j processing
4. **else**
5. Trace job cluster
6. end

Algorithm 2 takes D, $\alpha$, $\gamma$, init method, numPre$ as inputs and generates Tr as output. Both $\alpha$ and $\gamma$ help to minimize the error rate.

**Algorithm 2:** Calculation of the Optimal Refresh Period.

**Result:** Optimal Refresh Period Computation
**Output:** Tr

```
Create vector original based on length of dataset plus number of predictions
Create vector smoothed based on the length of dataset
Create vector b based on the length of dataset
I=1
J=1
Sum=0
original[0]=smoothed[0]+D[0]
if (!Avail initMethods=0) then
b[0]=D[1]-D[0]
if /initmethod = landD.length > 4) then
b[0]=D[3]-D[0]/3
else
b[0]=D[D.length]-D[0]/D]D.length
end
end
for Each d in D do
Calculate smoothed[i]
Calculate b[i]
Calculate original[i+l]
i=i+1
end
for each prediction in numPre do
Calculate original[j]
j=j+1
sum=sum+original[j]
end
```

Computations of smoothed[i], b[i], and original[i+1] are made as follows.

```
i ← 10
if i > 6 then
i ← i − 1
else
if i < 3 then
i ← i + 2
end if
end if
```

Our proposed fog model is designed to make retrieving data faster and lower the delay for time-sensitive applications. The fog is intended to store only recent data, not historical data. The threshold value has been set by the proposed algorithm. If the amount of data on the fog nodes goes above this threshold value, the older data would be remotely flushed from the fog nodes by the cloud, given the data have reached the cloud. Data at the fog nodes are short-lived as there is a physical limitation of data storage. So, data need to be flushed from the fog nodes regularly. In our fog model, the cloud monitors and controls this data flushing operation as a supreme authority in the architecture.

### *3.2. Cloud Sub-System*

In this subsystem, data received from the fog sub-system are processed as well as stored inefficiently. Various cloud repositories are placed for data storage with their unique ID, whereas high-end cloud servers are responsible for the processing. The cloud repository is capable enough of storing long-term storage. The cloud repository is designed with proper authentication and authorization such that no one can access the user's personal information except the user himself/herself. In our proposed model, the cloud maintains

two types of storage, Temporary and Permanent. These two storages are internally synchronized. Once a file that comes from any of the lower nodes in the hierarchy pops up in the Temporary storage, it is copied to Permanent storage. Temporary storage is synchronized with the fog nodes as well. So, the data from the fog nodes directly come to the temporary storage.

## 4. Experiment Setup & Results

iFogSim was used to demonstrate the efficiency of load balancing with the fog clustering approach with the determination of the Optimal Refresh Period. iFogSim has the capability to support fog system-level components and model behaviors such as load balancing, job scheduling, and so on. The components it accommodates include data centers, hosts, VMs, various baseline algorithms, and resource-provisioning policies. iFogSim has an essential set of classes written in Java programming language. The former is for balancing load, while the latter is meant for determining of Optimal Refresh Period to have effective load balancing. The application has the provision to have dynamically taken some clusters and jobs at run-time to have different sets of experiments. Figure 2 shows the performance comparison of execution time. The baseline approach provided by iFogSim to balance the load was compared with that of the proposed method. Optimal load-balancing decisions were made based on the runtime experience in terms of the number of clusters available and the number of jobs that arrive at the controller. Load balancing at the cluster level was handled by the local load balancer that considers VMs and their capacities to balance the load. The main observations here were the controller functionalities concerning the states associated with clusters. There is evidence of making decisions to send jobs to various clusters. The way tasks are assigned to clusters was based on the FCBLB algorithm, while the determination of the optimal refresh period to increase efficiency in load balancing was handled by the ORP algorithm.

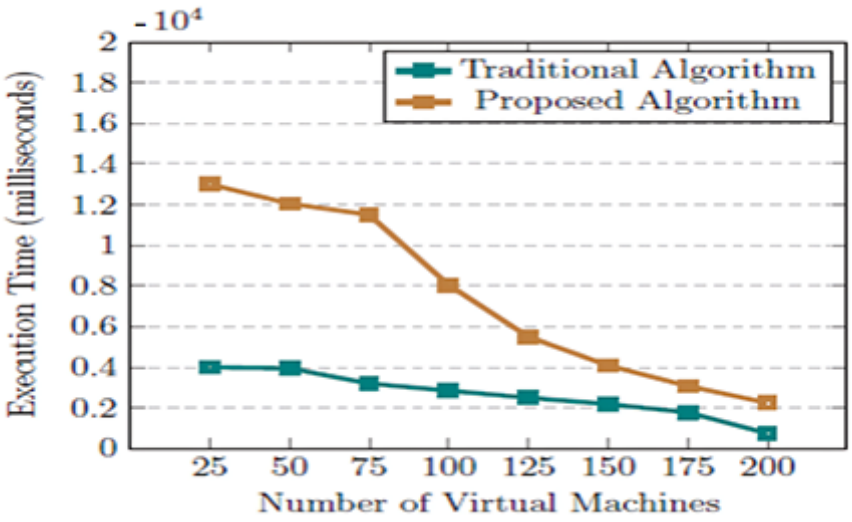


**Figure 2.** Performance comparison in terms of execution time.

When both the algorithms are associated by the controller and executed from time to time, the results revealed the performance enhancement of the proposed methodology when compared with that of the baseline method provided by iFogSim by default. The job allocation patterns can be traced to know the effectiveness of the proposed approach based on cloud clusters for load balancing. The first two clusters were overloaded, then the controller decided to use the third cluster. When all three clusters were overloaded, the fourth cluster was used for further job allocation. The third cluster was almost overloaded. As the simulation continued, the first and second clusters became normal once the allocated jobs were completed. In such cases, the controller decisions changed dynamically at run-time for effective balancing of load.

The simulation study carried out with the proposed algorithms using iFogSim revealed that the fog clustering-based approach for load balancing is effective due to the benefits of clusters, the load-balancing algorithm FCBLB, and the optimal refresh period algorithm ORP. The algorithms resulted in an improvement of job execution and CPU utilization. Figure 3 shows the performance comparison of CPU cycles. When the number of VMs changed, the execution time of jobs and the utilization of CPU cycles changed. The observations recorded with varying numbers of VMs shown on the horizontal axis. In the vertical axis, the execution time is presented. The number of VMs considered was 25 to 200 increasing by 25. There were two trends found in the results. The first trend is that for all VMs, the proposed approach's execution time was lower, while the existing approach was greater. The second trend is that when the number of VMs was greater, the execution time was decreased. The CPU cycles used are presented for the proposed and baseline approaches with varying numbers of VMs used in experiments.

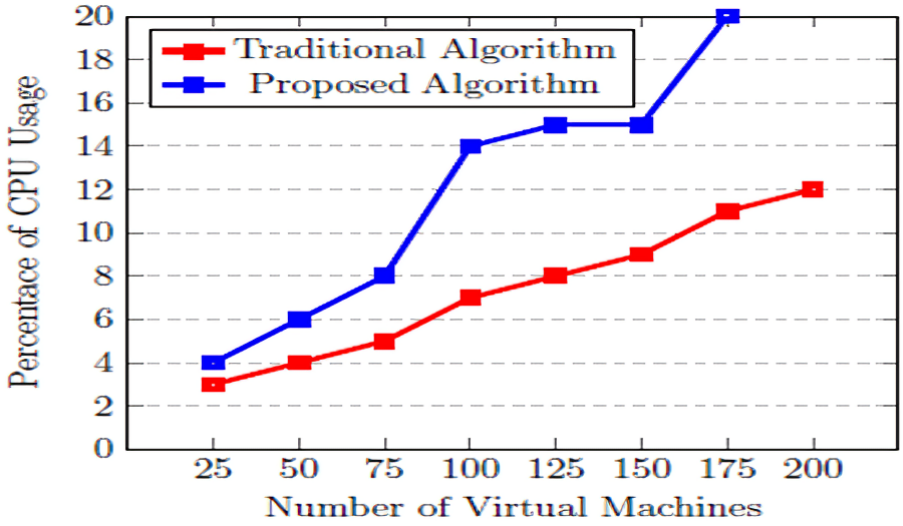


**Figure 3.** Performance comparison in terms of CPU cycles.

The usage of VMs with varying numbers is presented on a horizontal axis, while the vertical axis shows the percentage of CPU usage. Experiments were carried out with 25 to 200 VM increasing by 25. The results revealed that the number of VMs has an impact on the usage of CPU. When the number of VMs increased, the percentage of CPU usage also increased linearly. The proposed load-balancing approach outperformed the baseline load-balancing approach in terms of using less CPU usage. The observations showed the efficiency of the proposed methodology. Figures 4 and 5 demonstrate the results of make span and throughput, which are better for the proposed approach. Figure 6 shows that the proposed approach takes less energy consumption compared with the traditional approach. The key consideration of load balancing based on cluster states and refreshing the states by determining the optimal refresh period made a significant difference in the enhancement of load-balancing performance.

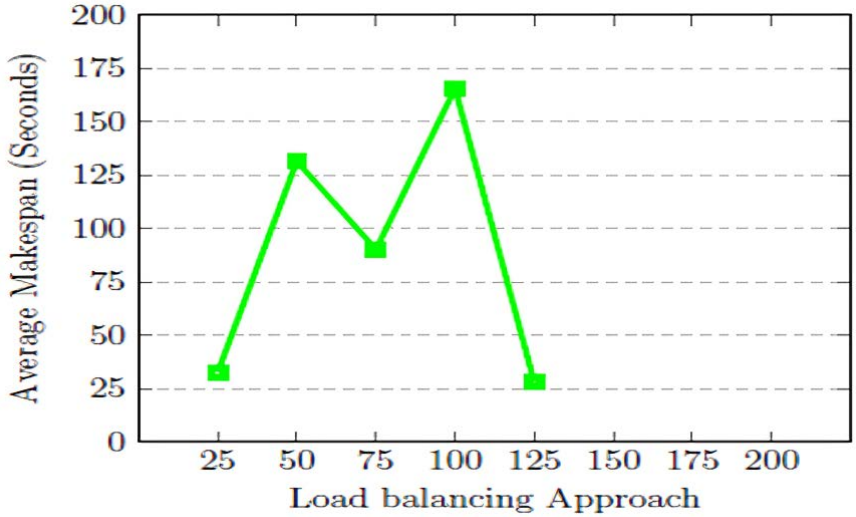


**Figure 4.** Make span comparison using workload.

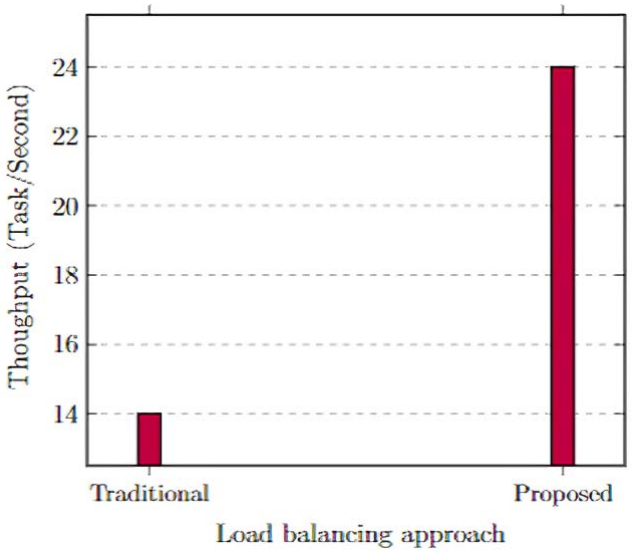


**Figure 5.** Thoughput comparison.

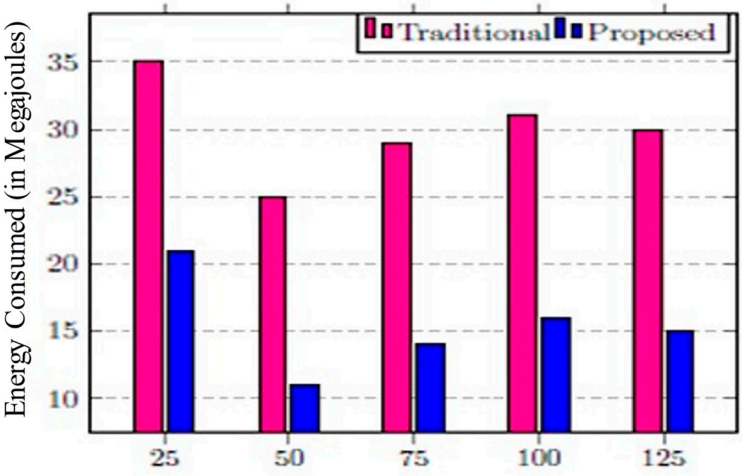


**Figure 6.** Energy consumption of devices.

## 5. Conclusions

This paper aimed to investigate how to minimize bandwidth costs and manage resources efficiently in a cooperative IoT fog-clod computing environment. This paper covered the solution to the problem of determining the optimal refresh period with a fog clustering-based load-balancing approach. The fog clustering-based load-balancing algorithm named FCBLB and an algorithm named ORP were designed and implemented. The developed methodology exploits the knowledge at runtime to determine the optimal refresh period to be rid of problems with either a too long refresh period or too short a refresh period. It strikes a balance between them to increase the efficiency of load balancing and reduce overhead on the system. iFogSim-based implementation is presented with visualization to let users intuitively observe the proposed approach's functioning. The baseline load-balancing approach provided by iFogSim was compared with the proposed method. The results shown in the view of execution time and CPU usage revealed the advantages of the proposed methodology that solve the problem of optimal refresh period for changing states in clustering-based load balancing. In the future, more optimization techniques could be applied for load balancing.

**Author Contributions:** Conceptualization, P.S., R.K., J.R. and S.J.; funding acquisition, J.K. and J.R.; investigation, P.S., R.K., J.R. and G.D.; methodology, P.S., R.K., J.R., S.J., G.D., J.K. and M.O.; resources, P.S., R.K., J.R., S.J., G.D., J.K. and M.O.; visualization, P.S., R.K., J.R., S.J., G.D., J.K. and M.O.; writing—review and editing, P.S., R.K., J.R., S.J., G.D., J.K. and M.O. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research was partly supported by the Technology Development Program of MSS [No. S3033853] and by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. 2021R1A4A1031509).

**Institutional Review Board Statement:** Not applicable.

**Informed Consent Statement:** Not applicable.

**Data Availability Statement:** Data will be made available on request.

**Acknowledgments:** The authors thank the anonymous reviewers who helped to improve the quality of the paper.

**Conflicts of Interest:** The authors declare no conflict of interest.

## References

1. Yu, C.-M.; Ku, M.L.; Wang, L.-C. Joint topology construction and hybrid routing strategy on load balancing for Bluetooth low energy networks. *IEEE Internet Things J.* **2021**, *8*, 7101–7102. [CrossRef]
2. Liang, Y.; Lan, Y. Tclbm: A task chain-based load balancing algorithm for microservices. *Tsinghua Sci. Technol.* **2020**, *26*, 251–258. [CrossRef]
3. Lyu, H.; Chen, K. Hybrid load-modulated balanced amplifier with high linearity and extended dynamic range. *IEEE Microw. Wirel. Compon. Lett.* **2021**, *31*, 1067–1070. [CrossRef]
4. Hung, L.-H.; Wu, C.-H.; Tsai, C.-H.; Huang, H.-C. Migration-based load balance of virtual machine servers in cloud computing by load prediction using genetic-based methods. *IEEE Access* **2021**, *9*, 49760–49773. [CrossRef]
5. Korndörfer, J.H.M.; Eleliemy, A.; Mohammed, A.; Ciorba, F.M. Lb4omp: A dynamic load balancing library for multithreaded applications. *arXiv* **2021**, arXiv:2106.05108. [CrossRef]
6. Mi, J.; Ren, Q.; Su, D. Parallel subdomain-level dgtd method with automatic load balancing scheme with tetrahedral and hexahedral elements. *IEEE Trans. Antennas Propag.* **2020**, *69*, 2230–2241. [CrossRef]
7. Cao, Y.; Chen, K. Hybrid asymmetrical load modulated balanced amplifier with wide bandwidth and three-way-doherty efficiency enhancement. *IEEE Microw. Wirel. Compon. Lett.* **2021**, *31*, 721–724. [CrossRef]
8. Aktas, M.F.; Behrouzi-Far, A.; Soljanin, E.; Whiting, P. Evaluating load balancing performance in distributed storage with redundancy. *arXiv* **2021**, arXiv:1910.05791. [CrossRef]
9. Giordano, A.; De Rango, A.; Rongo, R.; D'Ambrosio, D.; Spataro, W. Dynamic load balancing in parallel execution of cellular automata. *IEEE Trans. Parallel Distrib. Syst.* **2020**, *32*, 470–484. [CrossRef]
10. Liu, C.; Li, K.; Li, K. A game approach to multi-servers load balancing with load-dependent server availability consideration. *IEEE Trans. Cloud Comput.* **2018**, *9*, 1–13. [CrossRef]

11. Duong, T.-V.T.; Binh, L.H. Load balancing routing under constraints of quality of transmission in mesh wireless network based on software defined networking. *J. Commun. Netw.* **2021**, *23*, 12–22.
12. Liu, Y.; Gu, H.; Yan, F.; Calabretta, N. Highly-efficient switch migration for controller load balancing in elastic optical inter-datacenter networks. *IEEE J. Sel. Areas Commun.* **2021**, *39*, 2748–2761. [CrossRef]
13. Jahid, A.; Alsharif, M.H.; Uthansakul, P.; Nebhen, J.; Aly, A.A. Energy efficient throughput aware traffic load balancing in green cellular networks. *IEEE Access* **2021**, *9*, 90587–90602. [CrossRef]
14. Latif, Z.; Sharif, K.; Li, F.; Karim, M.M.; Biswas, S.; Shahzad, M.; Mohanty, S.P. Dolphin: Dynamically optimized and load balanced path for inter-domain SDN communication. *IEEE Trans. Netw. Serv. Manag.* **2020**, *18*, 331–346. [CrossRef]
15. Zhang, W.-Z.; Elgendy, I.A.; Hammad, M.; Iliyasu, A.M.; Du, X.; Guizani, M.; Abd El-Latif, A.A. Secure and optimized load balancing for multitier IoT and edge-cloud computing systems. *IEEE Internet Things J.* **2020**, *8*, 8119–8132. [CrossRef]
16. Zhao, Y.; Wang, X.; He, Q.; Zhang, C.; Huang, M. Plofr: An online flow route framework for power saving and load balance in SDN. *IEEE Syst. J.* **2020**, *15*, 526–537. [CrossRef]
17. Attiya, I.; Abualigah, L.; Elsadek, D.; Chelloug, S.A.; Abd Elaziz, M. An Intelligent Chimp Optimizer for Scheduling of IoT Application Tasks in Fog Computing. *Mathematics* **2022**, *10*, 1100. [CrossRef]
18. El Kafhali, S.; Salah, K. Performance modelling and analysis of Internet of Things enabled healthcare monitoring systems. *IET Netw.* **2019**, *8*, 48–58. [CrossRef]
19. Dhankhar, A.; Juneja, S.; Juneja, A.; Bali, V. Kernel parameter tuning to tweak the performance of classifiers for identification of heart diseases. *Int. J. E-Health Med. Commun.* **2021**, *12*, 1–16. [CrossRef]
20. Lakzaei, M.; Sattari-Naeini, V.; Sabbagh Molahosseini, A.; Javadpour, A. A joint computational and resource allocation model for fast parallel data processing in fog computing. *J. Supercomput.* **2022**, *78*, 12662–12685. [CrossRef]
21. El Kafhali, S.; Salah, K.; Alla, B.S. Performance evaluation of IoT-fog-cloud deployment for healthcare services. In Proceedings of the 2018 4th International Conference on Cloud Computing Technologies and Applications (Cloudtech), Brussels, Belgium, 26–28 November 2018.
22. Serdaroglu, K.C.; Baydere, S. An Efficient Multipriority Data Packet Traffic Scheduling Approach for Fog of Things. *IEEE Internet Things J.* **2021**, *9*, 525–534. [CrossRef]
23. Mekala, M.S.; Dhiman, G.; Srivastava, G.; Nain, Z.; Zhang, H.; Viriyasitavat, W.; Varma, G.P.S. A DRL-Based Service Offloading Approach Using DAG for Edge Computational Orchestration. *IEEE Trans. Comput. Soc. Syst.* **2022**. [CrossRef]
24. Yadav, K.; Jain, A.; Ahmed, O.S.N.M.; Hamad, S.A.S.; Dhiman, G.; Alotaibi, S.D. Internet of Thing based Koch Fractal Curve Fractal Antennas for Wireless Applications. *IETE J. Res.* **2022**, 1–10. [CrossRef]
25. Sumathy, B.; Chakrabarty, A.; Gupta, S.; Hishan, S.S.; Raj, B.; Gulati, K.; Dhiman, G. Prediction of Diabetic Retinopathy Using Health Records with Machine Learning Classifiers and Data Science. *Int. J. Reliab. Qual. E-Healthc.* **2022**, *11*, 1–16. [CrossRef]
26. Rashid, J.; Shah, S.M.A.; Irtaza, A. An efficient topic modeling approach for text mining and information retrieval through K-means clustering. *Mehran Univ. Res. J. Eng. Technol.* **2020**, *39*, 213–222. [CrossRef]
27. Zeidabadi, F.A.; Dehghani, M.; Trojovský, P.; Hubálovský, Š.; Leiva, V.; Dhiman, G. Archery algorithm: A novel stochastic optimization algorithm for solving optimization problems. *Comput. Mater. Contin.* **2022**, *72*, 399–416. [CrossRef]
28. Singh, N.; Houssein, E.H.; Singh, S.B.; Dhiman, G. HSSAHHO: A novel hybrid Salp swarm-Harris hawks optimization algorithm for complex engineering problems. *J. Ambient Intell. Humaniz. Comput.* **2022**, 1–37. [CrossRef]
29. Kanwal, S.; Rashid, J.; Kim, J.; Juneja, S.; Dhiman, G.; Hussain, A. Mitigating the Coexistence Technique in Wireless Body Area Networks by Using Superframe Interleaving. *IETE J. Res.* **2022**, 1–15. [CrossRef]
30. Juneja, S.; Dhiman, G.; Kautish, S.; Viriyasitavat, W.; Yadav, K. A Perspective Roadmap for IoMT-Based Early Detection and Care of the Neural Disorder, Dementia. *J. Healthc. Eng.* **2021**, *2021*, 6712424. [CrossRef]
31. Dhiman, G.; Kaur, G.; Haq, M.A.; Shabaz, M. Requirements for the Optimal Design for the Metasystematic Sustainability of Digital Double-Form Systems. *Math. Probl. Eng.* **2021**, *2021*, 2423750. [CrossRef]
32. Das, S.R.; Sahoo, A.K.; Dhiman, G.; Singh, K.K.; Singh, A. Photo voltaic integrated multilevel inverter based hybrid filter using spotted hyena optimizer. *Comput. Electr. Eng.* **2021**, *96*, 107510. [CrossRef]
33. Kansal, L.; Gaba, G.S.; Sharma, A.; Dhiman, G.; Baz, M.; Masud, M. Performance Analysis of WOFDM-WiMAX Integrating Diverse Wavelets for 5G Applications. *Wirel. Commun. Mob. Comput.* **2021**, *2021*, 5835806. [CrossRef]
34. Rashid, J.; Batool, S.; Kim, J.; Wasif Nisar, M.; Hussain, A.; Juneja, S.; Kushwaha, R. An Augmented Artificial Intelligence Approach for Chronic Diseases Prediction. *Front. Public Health* **2022**, *10*, 860396. [CrossRef] [PubMed]
35. Bangare, S.L.; Prakash, S.; Gulati, K.; Veeru, B.; Dhiman, G.; Jaiswal, S. The Architecture, Classification, and Unsolved Research Issues of Big Data extraction as well as decomposing the Internet of Vehicles (IoV). In Proceedings of the 2021 6th International Conference on Signal Processing, Computing and Control (ISPCC), Solan, India, 7–9 October 2021; pp. 566–571.
36. Dhiman, G.; Soni, M.; Pandey, H.M.; Slowik, A.; Kaur, H. A novel hybrid hypervolume indicator and reference vector adaptation strategies based evolutionary algorithm for many-objective optimization. *Eng. Comput.* **2021**, *37*, 3017–3035. [CrossRef]
37. Oliva, D.; Esquivel-Torres, S.; Hinojosa, S.; Pérez-Cisneros, M.; Osuna-Enciso, V.; Ortega-Sánchez, N.; Dhiman, G.; Heidari, A.A. Opposition-based moth swarm algorithm. *Expert Syst. Appl.* **2021**, *184*, 115481. [CrossRef]
38. Kumar, R.; Dhiman, G. A comparative study of fuzzy optimization through fuzzy number. *Int. J. Mod. Res.* **2021**, *1*, 1–14.
39. Vaishnav, P.K.; Sharma, S.; Sharma, P. Analytical review analysis for screening COVID-19 disease. *Int. J. Mod. Res.* **2021**, *1*, 22–29.
40. Chatterjee, I. Artificial intelligence and patentability: Review and discussions. *Int. J. Mod. Res.* **2021**, *1*, 15–21.

41. Gupta, V.K.; Shukla, S.K.; Rawat, R.S. Crime tracking system and people's safety in India using machine learning approaches. *Int. J. Mod. Res.* **2022**, *2*, 1–7.
42. Sharma, T.; Nair, R.; Gomathi, S. Breast Cancer Image Classification using Transfer Learning and Convolutional Neural Network. *Int. J. Mod. Res.* **2022**, *2*, 8–16.
43. Shukla, S.K.; Gupta, V.K.; Joshi, K.; Gupta, A.; Singh, M.K. Self-aware Execution Environment Model (SAE2) for the Performance Improvement of Multicore Systems. *Int. J. Mod. Res.* **2022**, *2*, 17–27.
44. Shao, C.; Yang, Y.; Juneja, S.; GSeetharam, T. IoT data visualization for business intelligence in corporate finance. *Inf. Process. Manag.* **2022**, *59*, 102736. [CrossRef]
45. Juneja, S.; Jain, S.; Suneja, A.; Kaur, G.; Alharbi, Y.; Alferaidi, A.; Alharbi, A.; Viriyasitavat, W.; Dhiman, G. Gender and Age Classification Enabled Blockschain Security Mechanism for Assisting Mobile Application. *IETE J. Res.* **2021**. [CrossRef]
46. Sharma, S.; Gupta, S.; Gupta, D.; Juneja, S.; Singal, G.; Dhiman, G.; Kautish, S. Recognition of Gurmukhi Handwritten City Names Using Deep Learning and Cloud Computing. *Sci. Program.* **2022**, *2022*, 5945117. [CrossRef]
47. Juneja, S.; Juneja, A.; Dhiman, G.; Jain, S.; Dhankhar, A.; Kautish, S. Computer Vision-Enabled Character Recognition of Hand Gestures for Patients with Hearing and Speaking Disability. *Mob. Inf. Syst.* **2021**, *2021*, 4912486. [CrossRef]
48. Gadekallu, T.R.; Pham, Q.V.; Nguyen, D.C.; Maddikunta, P.K.R.; Deepa, N.; Prabadevi, B.; Pathirana, P.N.; Zhao, J.; Hwang, W.J. Blockchain for edge of things: Applications, opportunities, and challenges. *IEEE Internet Things J.* **2021**, *9*, 964–988. [CrossRef]
49. Priya RM, S.; Bhattacharya, S.; Maddikunta, P.K.R.; Somayaji, S.R.K.; Lakshmanna, K.; Kaluri, R.; Hussien, A.; Gadekallu, T.R. Load balancing of energy cloud using wind driven and firefly algorithms in internet of everything. *J. Parallel Distrib. Comput.* **2020**, *142*, 16–26.
50. Pan, X.; Cai, X.; Song, K.; Baker, T.; Gadekallu, T.R.; Yuan, X. Location Recommendation Based on Mobility Graph with Individual and Group Influences. *IEEE Trans. Intell. Transp. Syst.* **2022**. [CrossRef]