Design of A new Algorithm by Using Standard Deviation Techniques in Multi Edge Computing with IoT Application

HASNAIN A. ALMASHHADANI[1*], XIAOHENG DENG[1], OSAMAH R. AL-HWAIDI[2], SARMAD T. ABDUL-SAMAD[2], MOHAMMED M. IBRAHM[1], and SUHAIB N. ABDUL LATIF[3]

[1]School of Computer Science and Engineering, Central South University of China
[2]Ministry of Higher Education and Scientific Research of Iraq
[3]Middle Technical University of Iraq
[e-mail : hasnainalmashhadani@gmail.com]
*Corresponding author: HASNAIN A. ALMASHHADANI

Received September 20, 2022; revised November 24, 2022; revised January 2, 2023; revised March 3, 2023; accepted March 16, 2023; published April 30, 2023

Abstract

The Internet of Things (IoT) requires a new processing model that will allow scalability in cloud computing while reducing time delay caused by data transmission within a network. Such a model can be achieved by using resources that are closer to the user, i.e., by relying on edge computing (EC). The amount of IoT data also grows with an increase in the number of IoT devices. However, building such a flexible model within a heterogeneous environment is difficult in terms of resources. Moreover, the increasing demand for IoT services necessitates shortening time delay and response time by achieving effective load balancing. IoT devices are expected to generate huge amounts of data within a short amount of time. They will be dynamically deployed, and IoT services will be provided to EC devices or cloud servers to minimize resource costs while meeting the latency and quality of service (QoS) constraints of IoT applications when IoT devices are at the endpoint. EC is an emerging solution to the data processing problem in IoT.

In this study, we improve the load balancing process and distribute resources fairly to tasks, which, in turn, will improve QoS in cloud and reduce processing time, and consequently, response time.

Keywords: IoT, Edge Computing, Cloud computing, standard deviation, load balancing

1. Introduction

The impact of the Internet of Things (IoT) on how data are shared and processed is unprecedented [1]. The number of devices that will be connected to IoT is estimated to reach 125 billion in 2030 [2]. These devices will generate a huge amount of data, which will be sent to cloud data centres for processing [3]; consequently, the load on cloud data centres and networks in general will be increased [4]. An optimal task scheduling strategy should always control this balance and improve the transmission rate when the task delay requirement is high; otherwise, the transmission rate should be reduced when the task delay requirement is low to achieve energy saving [5]. Despite the numerous advantages of cloud computing, many IoT applications cannot function efficiently on cloud. Effective load balancing achieves high availability of resources, which reduces delays in responding, especially for emergency tasks. Also, the adoption of central management in the load balancing process may cause a delay in executing the task due to the increased wait time for tasks to be allocated.

The dynamic nature of IoT environments, its associated real-time requirements, and the increasing processing capacity of edge devices are some issues that must be addressed [6]. When applications are run in clouds that are remote from the user, unpredictable latency occurs across the network [7]. However, many distributed end nodes with untapped resources can be used to achieve low latency and bandwidth in IoT networks [8].

Traditionally, IoT opens new possibilities and opportunities to transform our community into a connected world. Simultaneously, however, it poses some problems and risks. A huge number of devices are communicating with one another in IoT, and each device chooses an agent, which is established in advance within the range of scalability and efficiency.
Edge computing (EC) optimizes cloud computing systems by performing data processing at the edge of a network nearest to the data source. Low off-loading and latency result due to the closeness to end users.

An EC model has been proposed to address the aforementioned challenges by bringing processing and storage centres closer to the edge of a network [9]. The EC architecture uses edge-to-cloud hierarchy to reduce network traffic and improve quality of service (QoS) for delay-sensitive applications [10]. One of the most important challenges that EC is facing is the distribution of IoT services on available resources within this hierarchy [11]. Given the dynamic nature of IoT services and the dispersion of their locations over a wide geographical scale [12], ineffective load balancing will deteriorate QoS [13]. Effective load balancing results in high availability of resources, which reduces delays in responding, particularly for emergency tasks [14]. Moreover, the adoption of central management in the load balancing process may delay the execution of a task due to an increase in the waiting time of tasks to be allocated [15]. Therefore, the current work focuses on a decentralized management model based on reinforcement learning and collaborative artificial intelligence, allowing nodes to cooperate with one another to allocate tasks to available resources within the EC hierarchy. The main goal with load balancing is to reduce the standard deviation of the load between nodes so that its value is very close to zero. The standard deviation, is used to measure the extent to which the data is scattered about the average value.

This study presents a decentralized load balancing model for IoT services that can achieve effective load balancing and reduce resource usage cost. As evident in many IoT applications, this framework investigates the idea that adopting semantic technologies for the attributes of object capabilities should include improved communication skills.

The main contributions of this paper can be summarized as follows:

1. The proposed model is characterized by a decentralized multi-agent system for collective learning that utilizes edge-to-cloud nodes to jointly balance the input workload across the network and minimize the costs involved in service execution
2. A collaborative learning model is adopted when requests for services or resources arrive in the cloud to allocate incoming tasks to resources in a manner that achieves maximum use of available resources with the lowest possible cost to complete the tasks.
3. A new algorithm is designed using standard deviation techniques to shorten time delay and response time by achieving effective load balancing in EC and cloud computing.
4. The load balancing process and the distribution of available resources to tasks to be performed are improved, positively reflecting on increasing the QoS provided to reduce processing time, and thus, decrease latency.
5. Load balancing is enhanced by using standard deviation techniques between EC and cloud computing to achieve the maximum use of available resources and the lowest possible cost for completing tasks in IoT applications.
6. Inclusive grasp of several pointes, workload division method, and the allowed workload redistribution level in the network.

2. Related Work

Heterogeneous networks and sensing resources have to be frequently shared with multiple applications in IoT environments [16]. To meet the objective of meeting the ultralow latency requirements of IoT applications, load balancing plays an essential role in providing QoS guarantees in cloud computing [17]. Traditional data mining techniques used for the analysis of flat vectorial data are inefficient for handling large-scale data with inherent relational dependencies, weights, edge directions, and heterogeneity between system elements [18]. Given its proximity to end users, low latency, and other advantages, EC has elicited considerable interest in the research community [19]. The smaller the value, the more balanced the load distribution of a cloud computing system, and the better the performance of the current system. A request will not meet the deadline if the response time is longer than the deadline or if an IoT device (user) cannot reach any instance of a service [20, 21]. Meanwhile, fog nodes may also offload requests to other fog nodes, to balance their workload and minimize the response delay of IoT requests [22]. The authors of [23] required running multiple repetitions of experiments to obtain the average and deviation of the results. The authors of [24] helped gain smooth load balancing within a virtual machine, increasing the throughput. This methodology considers the order of jobs that a virtual machine is handling. Various load balancing algorithms have been used in cloud computing systems, such as the round-robin algorithm [25]. The authors of [26] believed that successful expert-based weight rebalancing estimation would be diminished when imperfection occurred in the framework, including balancing load distributions between edge and cloud resources [27]. The authors of [28] aimed to enhance the service latency and offloading exhaustion of a new EC paradigm applied to IoT compared with traditional cloud scenarios. Network QoS is achieved by proposing a QoS- and connection-aware cloud service structure process for accepting end-to-end QoS requirements in the cloud [29]. The authors of [30] achieved load balancing by immediately involving datacentre power consumption with a variety of workloads in the cloud. The accuracy of the

application results will not diminish by balancing tasking frequency among edge nodes [31]. In [32], optimal resource management policies, such as capacity allocation, load balancing, energy optimization, and QoS guarantee, can be efficiently implemented, **Table 1** shows us the offerings of an evaluation of the interrelated studies and the proposed work.

Table 1. Features of the citation papers in the Related work

Reference	Miscellaneous	Quality of Service (QoS)	Load Balancing	Distributed IoT
[16]	*	*	-	*
[17]	-	*	*	*
[18]	*	*	-	-
[19]	*	-	-	-
[20]	*	*	-	*
[21]	*	*	-	*
[22]	*	-	-	-
[23]	*	*	-	*
[24]	*	*	*	
[25]	*	-	*	-
[26]	*	-	-	*
[27]	*	-	*	*
[28]	*	*	-	*
[29]	*	*	-	-
[30]	*	-	*	-
[31]	-	-	*	*
[32]	-	*	*	*

3. Design the Conceptual Architecture for IoT-based Computing Networks.

The physical network is modelled in the form of an undirected graph denoted by the symbol G= (V, E), as shown in **Fig. 1**

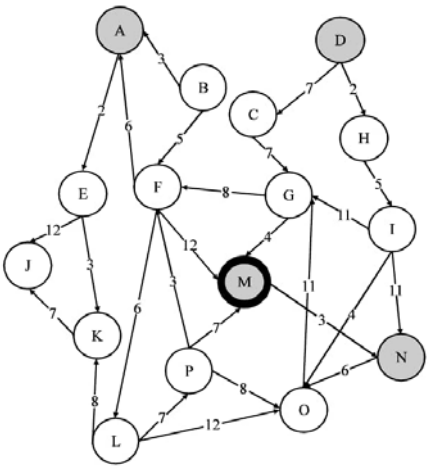


Fig. 1. The Physical Network Is Modelled

Where:

$V = (C \cup F)$.

F: The group of nodes at the edges of the cloud.
C: The group of nodes located in the centre of the cloud.
E: Edges between nodes.

Each node is denoted by fj and each node is represented by:

- Capacity fj: It is Measured in Multi Instructions per Second (MIPS).
- Memory capacity Rfj and measured in bytes.
- Storage capacity is Sfj and is measured in bytes.

Assume that Group A represents a set of IoT services to be implemented. Each service $\alpha_i \in A$ must be allocated to one of the nodes in the cloud in accordance with the service requirements (deadline, processing capacity, memory, and storage).

When IoT services reach the edge of the cloud, the nodes at the edge are responsible for allocating the task.

The proposed model in **Fig. 2** is based on EC hierarchy, and its architecture has three levels. The first level represents the edge of a network. It is a group of nodes near the areas where IoT devices are located and connected with a local area network (LAN). This level is called network edge.

The second level represents the group of nodes that are farthest from the areas where IoT devices are located. The resources here are higher than those in the first level, and this level is called network fog. The network edge is connected to the network fog through wide area networks (WANs).

The third level represents the centre of the cloud. It is farther from the previous two levels than the areas where IoT devices are located. It contains nodes with higher resources than the two previous levels. The cloud centre is connected to the network fog and edge through WANs.

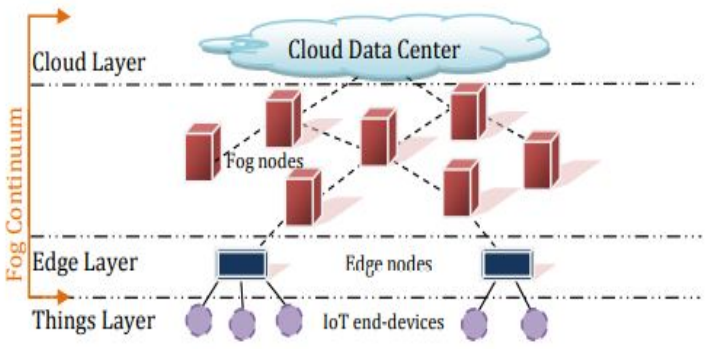


Fig. 2. Proposed Network Architecture

Load balancing refers to the process of allocating large loads to small processing nodes to achieve optimal response time and optimum investments in hardware and software resources, improving overall system performance. Load balancing helps in the equal distribution of computing resources to achieve a high-level of customer satisfaction. The correct use of

resources and appropriate load balancing will result in the optimal investment of available resources, eliminating the need to add new equipment at high costs.

Load balancing is used to distribute loads on processing nodes in a manner that achieves maximum throughput and short response time while avoiding overloading the nodes. Load balancing should be implemented correctly because failure of one of the nodes can lead to data unavailability. A good load balancing algorithm avoids overload and low load on a given node, such that no nodes are overloaded while other nodes are idle.

The objective of the current research is to improve the quality and efficiency of cloud computing by developing a methodology for distributing loads on processing nodes, achieving balanced and stable load in the cloud and improving processing time, and consequently, response time.

The resources allocated to each node are different, and they typically change dynamically depending on the resources booked by the tasks assigned to a node and the resources released when a node finishes executing a task. The capability of each node is calculated by considering the resources available to each of them as follows:

$$\tau_{CPU} = \frac{P_{CPU}}{P_{Max}} \times 100\% \, CPU \quad (1)$$

$$\tau_{mi} = \frac{m_i}{m_{iMax}} \times 100\% \, Internal \;\; Storage \quad (2)$$

$$\tau_{me} = \frac{m_e}{m_{eMax}} \times 100\% \quad External \; Storage \quad (3)$$

$$\tau_j = (\varphi_1 \times \tau_{CPU}) + (\varphi_2 \times \tau_{mi}) + (\varphi_3 \times \tau_{me}) : \sum \varphi = 1 \quad (4)$$

where

τ_j: capability of a node,

P_{CPU}: processing power available within a node,

m_i: internal storage available within a node,

m_e: external storage available within a node,

P_{MAX}: total processing power of a node,

M_{iMax}: total internal storage of a node,

M_{eMax}: total external storage of a node,

φ: weight parameter for adjusting the impact degree of resources.

The capability of each node is constantly changing, depending on several factors.

The capability of a node τ_j decreases when a new task is assigned to it. The amount of decrease (1−μ) is based on the ratio of the resources consumed to the total resources allocated to this node.

$$\tau_j(t+1) = (1-\mu) \times \tau_j(t) \quad (5)$$

where μ is a coefficient for determining the ratio of consumed resources to the total amount of resources.

The capability of node τ_j increases upon completion of a certain task. The amount of increase is based on the ratio of released resources that are dedicated to that task.

$$\tau_j(t+1) = (1+v) \times \tau_j(t) \quad (6)$$

where v is a coefficient for determining the ratio of the released resources to the total amount of resources.

Response time is defined as the time between the moment the request is sent and the moment the first response to this request is received. It is equal to the sum of propagation time, waiting time, and execution time. Response time must be less than the maximum deadline. Therefore, the expected response time of a task on a node must be shorter than the deadline.

The estimated task execution time (ET) on each node is calculated using the following equation:

$$ET = \frac{TL}{Capacity \times cores(T)} \quad (7)$$

Where

TL (task length), length of task T,

Capacity: rate of processing capacity per core (MIPS),

Cores(T), number of cores needed by task T

Therefore, the estimated response time in cloud $RT_{i,j}$ is defined as the sum of waiting time and estimated task execution.

$$RT_{i,j} = WT_i + ET_{i,j} \quad (8)$$

Where

WT_i : represents the waiting time for the task to be allocated,

$ET_{i,j}$: represents the estimated execution time.

The processing procedure uses the M/M/1 queuing system. In that system, if the access rate (MIPS) to the fj node is equal to $z_{f,j}$ and the processing capacity of the fj node is equal $Capacity_{f,j}$, then the expected delay is given as a probability ratio in the following equation:

$$Latency_{i,j} = \frac{1}{Capacity_{f,j} - z_{f,j}} \quad (9)$$

The primary objective of load balancing is to reduce the standard deviation σ of the load between nodes until its value is extremely close to zero. The standard deviation σ is used to measure the extent to which data are scattered about the average value.

$$\sigma = \sqrt{\frac{1}{V} \times \sum_{j=1}^{V} (TL(f_j) - Average(TL))^2} \quad (10)$$

Where

TL: current load on fj node (measured in MIPS),

V: number of nodes.

4. Proposed Solution

IoT devices create service requests that are sent to the nodes at the edge of a network to determine which node the tasks will be allocated to.

Each node knows the requirements of the received task and the resources available to nearby fog nodes. All nodes that receive tasks participate in creating a scheme for allocating tasks and choosing the best plan.

Schema creation: When receiving a task, each node at the edge creates a set of schemas for allocating tasks and calculating the sum of the expected total execution time for all the tasks in each schema in accordance with the following equation:

$$ET_{Total} = \sum ET_{i,j} \quad (11)$$

e. Thus, if the nearest end nodes are capable of executing incoming tasks, then the tasks will be assigned to them to reduce deadline violations and reduce network traffic, which, in turn, increases QoS.

Fig. 3 shows the machine used:

- The system view on each node represents a file that contains its adjacent nodes (the nearest nodes) and the resources available to each.
- Service view represents a file that contains the incoming tasks to the node and the requirements for each task. All nodes exhibit the most diverse characteristics and capabilities, i.e., different residual energy, power consumption, processing capacity, available memory, and capability to perform a limited number of tasks.

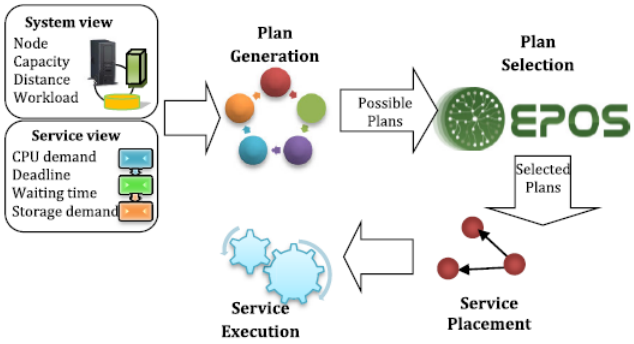


Fig. 3. Machine Used

The edge is connected to IoT devices by LANs, because the proposed model is close to IoT devices; hence, propagation delay from IoT devices to the edge is extremely small and negligible. The first level (edge) is connected to the second level (fog) and the third level (cloud) by WANs. The purpose of this setup is to reduce propagation time as much as possible, allowing the task to reach the edge of the network with a negligible propagation delay. Thus, propagation time, waiting time, and processing time are shortened as much as possible.

IoT devices created service query and submit them to the edge nodes for status award and performance. It is supposed that the receiver nodes/agents know the matter of the received requests and the capabilities of the neighbouring nodes.

The proposed model features a multi-agent decentralization system that uses nodes in EC and cloud to achieve a balance between available resources and implement the task requests received from the application of IoT.

For the process of generating a single assignment schema:

- Each edge node arranges incoming tasks from lowest to highest in terms of difference between the deadline and waiting time of the task.
- A node randomly selects a set of adjacent fog nodes that are available to receive the task.
- The edge node arranges the chosen fog nodes in accordance with distance from nearest to farthest.
- The node allocates the tasks arranged in Step 1 one by one, calculates the total estimated execution time and standard deviation, and then generates the schema.

- The generated schemas are exported to all edge nodes to search for the best optimization of the allocation and learning process from the edge to the cloud centre.

Scheme selection: Each agent generates a certain number of possible plans with the estimated total implementation time and standard deviation for each scheme. In the next step, all agents collaborate to select the best plan from among the feasible plans by considering two goals (local goal L to reduce the estimated execution time and global goal G to achieve effective load balancing). The following relationship can be used to weight one of the two objectives above the other:

$$\lambda L_{(t)} + (1-\lambda) G_{(t)} \qquad (12)$$

λ is a weight parameter whose value is between [0–1] that is used to weight one goal over the other in accordance with the requirements of the task.

When $\lambda = 1$, the relationship becomes $\lambda L_{(t)}$. Thus, in choosing the scheme, full priority is given to the local goal, which is to reduce the estimated execution time, regardless of whether it achieves the best load balancing (a standard deviation value close to zero). This case is used for very urgent and sensitive tasks that are required to be implemented as quickly as possible.

When $\lambda = 0$, the relationship becomes $G_{(t)}$. Thus, in choosing the scheme, full priority is given to the global goal of achieving the best load balancing.

The criterion for completing iterations is determined by the system administrator with a specified number of iterations. After all iterations are completed, each agent allocates incoming tasks to the fog nodes or the cloud centre in accordance with the selected scheme among all agents. The proposed algorithm is explained as follows:

Proposed Algorithm: -

$Input$: $\left\{ a: \textit{ set of request services}, \; n: \textit{ set of network nodes} \right\}$
$Output$: $\{\Delta: \textit{ set of possible plans}\}$
$for(q = 1 \textit{ to } \Delta) \textit{ do}$
$Sort \; a \textit{ in the order } (Deadline_i - WT_i) \textit{ from low to high};$
$h \leftarrow \textit{select neighboring nodes from } n;$
$Sort \; h \textit{ in term of proximity from low to high};$
$i, j \leftarrow 0;$
$while(\; a \textit{ is not empty}) \textit{ do}$
$Select \; a_i \textit{ from } a \textit{ and } f_j \textit{ from } h;$
$if(f_j \textit{ resources can hosted } a_i) \textit{ then}$
$assign \; a_i \textit{ to } f_j;$
$Update \; f_j \textit{ load according to } Equation(5);$
$elseif \left(\begin{matrix} \textit{the cloud node } (c_k) \\ \textit{has enough capacity} \end{matrix} \right) \textit{ then}$
$assign \; a_i \textit{ to } c_k;$
$Update \; c_k \textit{ load according to } Equation(5);$
$end \; if;$
$Remove \; a_i \textit{ from } a;$
$i++, \; j++;$
$end \; while;$
$calcuate \; ET_{Total} \textit{ according to } Equation(11);$
$calcuate \; \sigma \textit{ according to } Equation(10);$

$end\ for$
$Sort\ \Delta\ in\ the\ order\ of\ ET_{Total}$
$from\ low\ to\ high;$
$Re\,turn\ \Delta;$

5. Evaluation

Given the cost and complexity of implementing a realistic model of this system, it will be designed on the basis of mathematical modelling and graphs to model the relationships among the cloud center, fog infrastructure, and edge. In this research, a different network topology is designed through three graphic models: Barbasi Albert (BA), Watts Strogazt (WS), and Erdos Renyi (ER).

The following **Fig. 4** show the network diagrams of the three previous models for a network of (1000, 800, 600, 400, and 200) nodes.

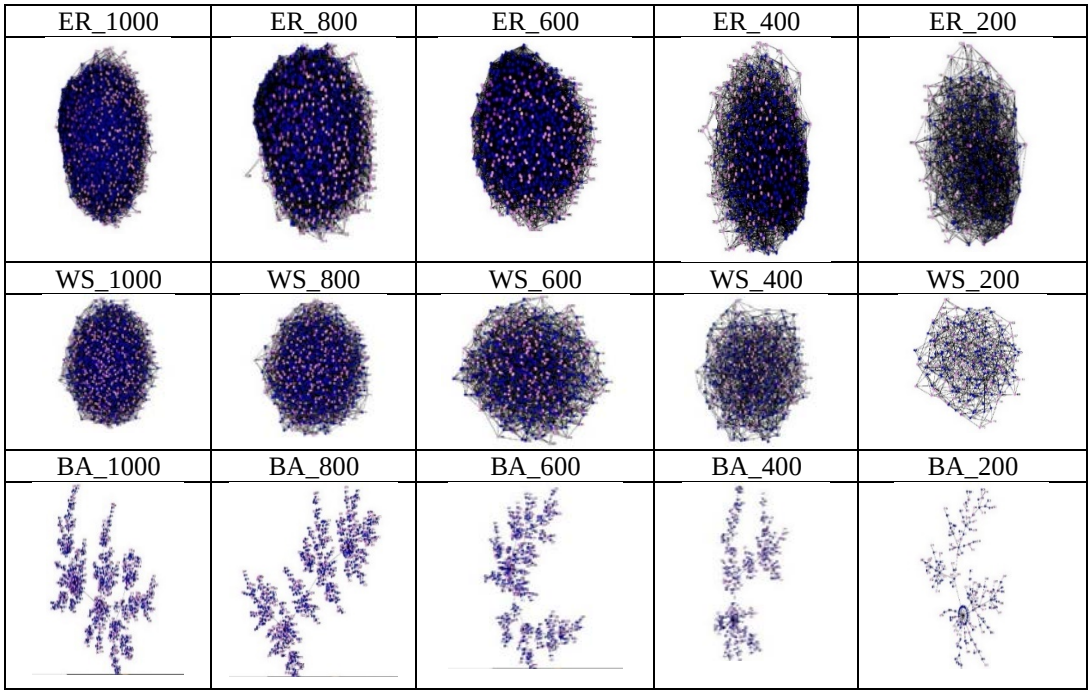


Fig. 4. The Network Used to Model of Topology Diagrams

The simulation was performed with NetBeans by using Java to simulate an edge-to-cloud network of nodes. Previous graphic modelling was performed using the Java Graphic Stream library. The input workload was used based on the Google Cluster Trace dataset（89000）tasks events, which contains data collected from a variety of input workloads on 12500 devices for a period of 30 days.

The number of plans for each agent is set to 20. Evaluation is implemented in five periods by using the topology ER_1000, BA_1000, and BS_1000. The results of each period are extracted.

Comparison is made with the first fit (FF) model and cloud centre without edge, which depends on reducing transition delay between nodes. Each node tracks transition delay between it and the remaining nodes (number of hops) and prepares a list of its neighbouring nodes to establish priority in assigning tasks to them if they have sufficient resources to execute the task.
The proposed approach and the FF model are evaluated by calculating the resource utilization ratio $(1-\tau_j)$ of each node and then its standard deviation, which should be as close as possible to zero.

6. Results and Discussion

Table 2, provides the difference among the proposed model, the FF model, and the cloud centre without edge in terms of the value of the standard deviation of the resources consumed (load) in each of the previous stages.

Table 2. Proposed Model

Number of Type of Topology	FF Model	Proposed Model	Cloud Model
ER_1000_P1	0.398544	0.0714475	0.4521253
ER_1000_P2	0.428555	0.228239	0.4586405
ER_1000_P3	0.318149	0.230933	0.3178336
ER_1000_P4	0.234511	0.226706	0.2349422
ER_1000_P5	0.226706	0.178521	0.186347
WS_1000_P1	0.397351	0.224753	0.4520282
WS_1000_P2	0.41597	0.155158	0.4493863
WS_1000_P3	0.318078	0.26691	0.3192285
WS_1000_P4	0.241978	0.269472	0.2436892
WS_1000_P5	0.250923	0.26703	0.2633613
BA_1000_P1	0.317145	0.267132	0.3206646
BA_1000_P2	0.358709	0.26845	0.3645399
BA_1000_P3	0.258892	0.263427	0.2626296
BA_1000_P4	0.36922	0.268908	0.3743
BA_1000_P5	0.341173	0.267102	0.3456636

Fig. 5 shows the difference among the proposed model, the FF model, and the cloud centre without edge in terms of the standard deviation of the resources consumed (load) in each of the previous stages.

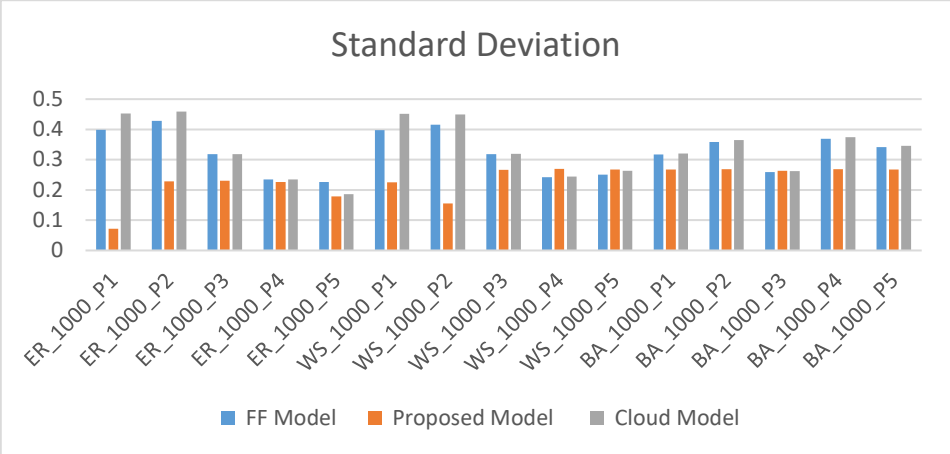


Fig. 5. Standard Deviation

We note from the results of the superiority of the proposed model over the FF model, where the proposed model achieved a standard deviation value of the load between the nodes less than the FF model and therefore the proposed model achieves an optimal utilization of the available resources, which in turn leads to investing the available resources in the best possible way, which leads to reducing the high cost of adding new equipment. The reason for this is that the proposed model provides effective load balancing and distributes tasks to the available resources in a fair and thoughtful manner that takes into account the task size, requirements, deadline and contract resources, thus improving response time. The proposed model also avoids loading nodes with high load and other nodes with low load, thus avoiding the bottleneck problem for high load nodes.

7. Conclusion and Future Research Works

Notably, the results demonstrate the superiority of the proposed model over the FF model with three models for a network of (1000, 800, 600, 400, and 200) nodes. The simulation was performed with NetBeans by using Java, wherein the proposed model achieved a standard deviation value of the load between the nodes that was less than that of the FF model. Therefore, the proposed model achieves optimal utilization of available resources, which, in turn, leads to investing available resources in the best possible manner. Consequently, the high cost of adding new equipment is reduced. The reason for this result is as follows: the proposed model provides effective load balancing and distributes tasks to available resources in a fair and thoughtful manner that considers task size, requirements, deadline, and contract resources, and thus, response time is improved. The proposed model also avoids loading nodes with a high load and other nodes with a low load; hence, the bottleneck problem for high load nodes is avoided. However, with future work, the increases of the IoT devises to get fixability and mobility, to activities to the edge and develop enhancements for the technology by design and new techniques such as Mulita User Superposition Transmission (MUST), release the user and share the resources for sharing information between users with higher performance.

References

[1] Staal, T., "The impact of the Internet of Things on the demand of cloud resources," *University of Twente*, 2022. Article (CrossRef Link)

[2] Chakraborty, P., R.N. Dizon, and S. Bhunia, "ARTS: A Framework for AI-Rooted IoT System Design Automation," *IEEE Embedded Systems Letters*, 14(3), pp. 151-154, 2022. Article (CrossRef Link)

[3] Suresh, A. and S. Paiva, *Deep Learning and Edge Computing Solutions for High Performance Computing*, Springer, 2021. Article (CrossRef Link)

[4] Wei, X., et al., "Joint optimization of energy consumption and delay in cloud-to-thing continuum," *IEEE Internet of Things Journal*, 6(2), pp. 2325-2337, 2019. Article (CrossRef Link)

[5] Hazra, A., et al., "Joint computation offloading and scheduling optimization of IoT applications in fog networks," *IEEE Transactions on Network Science and Engineering*, 7(4), pp. 3266-3278, 2020. Article (CrossRef Link)

[6] Cicirelli, F., et al., "Edge Computing and Social Internet of Things for Large-Scale Smart Environments Development," *IEEE Internet of Things Journal*, 5, pp. 2557-2571, 2018. Article (CrossRef Link)

[7] Xiao, K., et al., "EdgeABC: An architecture for task offloading and resource allocation in the Internet of Things," *Future Generation Computer Systems*, 107, pp. 498-508, 2020. Article (CrossRef Link)

[8] Islam, M.J., et al., "Blockchain-SDN-Based Energy-Aware and Distributed Secure Architecture for IoT in Smart Cities," *IEEE Internet of Things Journal*, 9(5), pp. 3850-3864, 2022. Article (CrossRef Link)

[9] Liu, L., et al., "Vehicular edge computing and networking: A survey," *Mobile networks and applications*, 26(3), pp. 1145-1168, 2021. Article (CrossRef Link)

[10] Nezami, Z., et al., "Decentralized edge-to-cloud load balancing: Service placement for the Internet of Things," *IEEE Access*, 9, pp. 64983-65000, 2021. Article (CrossRef Link)

[11] Yang, Z., et al., "Efficient resource-aware convolutional neural architecture search for edge computing with Pareto-Bayesian optimization," *Sensors*, 21(2), pp. 444, 2021. Article (CrossRef Link)

[12] Salaht, F.A., F. Desprez, and A. Lebre, "An overview of service placement problem in fog and edge computing," *ACM Computing Surveys (CSUR)*, 53(3), pp. 1-35, 2020. Article (CrossRef Link)

[13] Moghaddam, S.M., et al., "Metrics for improving the management of Cloud environments—Load balancing using measures of Quality of Service, Service Level Agreement Violations and energy consumption," *Future Generation Computer Systems*, 123, pp. 142-155, 2021. Article (CrossRef Link)

[14] Ghomi, E.J., A.M. Rahmani, and N.N. Qader, "Load-balancing algorithms in cloud computing: A survey," *Journal of Network and Computer Applications*, 88, pp. 50-71, 2017. Article (CrossRef Link)

[15] Joshi, S. and U. Kumari, "Load balancing in cloud computing: Challenges & issues," in *Proc. of 2016 2nd International Conference on Contemporary Computing and Informatics (IC3I)*, 2016. Article (CrossRef Link)

[16] Gupta, H., et al., "iFogSim: A toolkit for modeling and simulation of resource management techniques in the Internet of Things, Edge and Fog computing environments," *Software: Practice and Experience*, 47(9), pp. 1275-1296, 2017. Article (CrossRef Link)

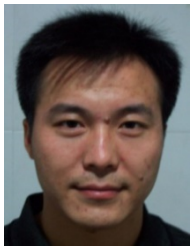
[17] Sharma, M., R. Kumar, and A. Jain, "A QoS-Enabled Load Balancing Approach for Cloud Computing Environment Join Minimum Loaded Queue (JMLQ)," *International Journal of Grid and High Performance Computing (IJGHPC)*, 14(1), pp. 1-19, 2022. Article (CrossRef Link)

[18] Bacciu, D., A. Micheli, and M. Podda, "Edge-based sequential graph generation with recurrent neural networks," *Neurocomputing*, 416, pp. 177-189, 2020. Article (CrossRef Link)

[19] Yu, W., G. Hou, and J. Li, "Supply chain joint inventory management and cost optimization based on ant colony algorithm and fuzzy model," *Tehnički vjesnik*, 26(6), pp. 1729-1737, 2019. Article (CrossRef Link)

[20] Xu, X., et al., "Dynamic resource provisioning with fault tolerance for data-intensive meteorological workflows in cloud," *IEEE Transactions on Industrial Informatics*, 16(9), pp. 6172-6181, 2020. Article (CrossRef Link)
[21] Lera, I., C. Guerrero, and C. Juiz, "Availability-aware service placement policy in fog computing based on graph partitions," *IEEE Internet of Things Journal*, 6(2), pp. 3641-3651, 2019. Article (CrossRef Link)
[22] Mohmmad, S., et al., "Cost function for energy (CFE) in Fog based IoT networks," in *Proc. of AIP Conference Proceedings*, 2418(1), 2022. Article (CrossRef Link)
[23] Skarlat, O., et al., "A framework for optimization, service placement, and runtime operation in the fog," in *Proc. of 2018 IEEE/ACM 11th International Conference on Utility and Cloud Computing (UCC)*, 2018. Article (CrossRef Link)
[24] LD, D.B. and P.V. Krishna, "Honey bee behavior inspired load balancing of tasks in cloud computing environments," *Applied soft computing*, 13(5), pp. 2292-2303, 2013. Article (CrossRef Link)
[25] Mishra, K. and S.K. Majhi, "A binary bird swarm optimization based load balancing algorithm for cloud computing environment," *Open Computer Science*, 11(1), pp. 146-160, 2021. Article (CrossRef Link)
[26] Jagannath, A., J. Jagannath, and T. Melodia, "Redefining wireless communication for 6G: Signal processing meets deep learning with deep unfolding," *IEEE Transactions on Artificial Intelligence*, 2(6), pp. 528-536, 2021. Article (CrossRef Link)
[27] Dhananjaya, M., K. Sharma, and A.K. Chaturvedi, "A Systematic Review on the QoS-Aware Service Provisioning in Fog Computing," in *Proc. of International Conference on Communication and Artificial Intelligence*, pp. 537-545, 2022. Article (CrossRef Link)
[28] Jassas, M.S. and Q.H. Mahmoud, "Evaluation of Failure Analysis of IoT Applications Using Edge-Cloud Architecture," in *Proc. of 2022 IEEE International Systems Conference (SysCon)*, 2022. Article (CrossRef Link)
[29] Brogi, A., S. Forti, and A. Ibrahim, "Predictive analysis to support fog application deployment," in *Fog and edge computing: principles and paradigm*, Wiley STM, 2019, pp. 191-222.
[30] Kumar, C., et al., "Greening the Cloud: A Load Balancing Mechanism to Optimize Cloud Computing Networks," *Journal of Management Information Systems*, 39(2), pp. 513-541, 2022. Article (CrossRef Link)
[31] Mahmoudi, M., A. Avokh, and B. Barekatain, "SDN-DVFS: an enhanced QoS-aware load-balancing method in software defined networks," *Cluster Computing*, 25(2), pp. 1237-1262, 2022. Article (CrossRef Link)
[32] Jyoti, A. and M. Shrimali, "Dynamic provisioning of resources based on load balancing and service broker policy in cloud computing," *Cluster Computing*, 23(1), pp. 377-395, 2020. Article (CrossRef Link)

Mr. HASNAIN A. ALMASHHADANI Received his B.Sc. degree in computer science from University of Baghdad Iraq, and his M.Sc. degree in computer science and information technology from Central South University –China, he is working in ministry of higher education and scientific research – Iraq (MOHESR), where currently he is PhD. Student in the Central South University –china, his current research interests in the cloud computing, 5G, mobile edge computing, Internet of hings. E-mail: hasnainalmashhadani@gmail.com



Professor XIAOHENG DENG is Vice-dean of School of computer science and engineering Director of Hunan Data Sensing and Exchange Equipment Engineering Center IEEE RS Chapter Changsha Chairman CCF Changsha Executive Committee Member of CCF Pervasive Computing Council.E-mail: dxh@csu.edu.cn
Website : http://faculty.csu.edu.cn/dengxiaoheng/en/index.htm

Mr. OSAMAH R. AL-HWAIDI Received his B.Sc. degree in computer science from Al-Rafidain University College Iraq, recently he is working in Ministry of Higher Education and scientific research Iraq as a computer engineer. E-mail: ubnt1987@gmail.com

Mr. SARMAD T. ABDUL-SAMAD Received his B.Sc. Degree in computer science from Al-Nahrain University Iraq, received his M.Sc. degree in computer science from Al-Nahrain University Iraq, work Content Based Image Retrieval (CBRN) also work on data mining and digital image processing, recently he is working in Ministry of Higher Education and scientific research Iraq. E-mail: Sarmad.thaer991@rdd.edu.iq

Mr. MOHAMMED M. IBRAHM Received his B.Sc. degree in computer science from Al-Rafidain University College Iraq, and his M.Sc. degree in computer science and information technology from Central South University –China, he is working in ministry of higher education and scientific research – Iraq (MOHESR), where currently he is PhD. Student in the Central South University –China. E-mail: mohd_soft2006@yahoo.com

Mr. Suhaib N. Abdul Latif, received his B.Sc.in computer science and engineering from middle technical university in iraq, received his MS from Central south university of technology China, work on middle technical university, Iraq. E-mail: suhaib84iq@gmail.com