Performance Enhancement of Fog Environment with Deadline Aware Resource Allocation Algorithm

Nirmeen A. El-Bahnasawy
Computer Science & Eng. Dept., Faculty of Electronic Eng. Menoufia University, Menoufia, Egypt.
nirmeena.el-bahnasawy@el-eng.menofia.edu.eg,
https://orcid.org/0000-0002-4542-323X

Amal EL-Nattat
Computer Science & Eng. Dept., Faculty of Computing and Artificial Intelligence.
Sadat City University.
Menoufia, Egypt
amal.elnatat@fcai.usc.edu.eg

Ayman El-Sayed
Computer Science & Eng. Dept., Faculty of Electronic Eng. Menoufia University Menoufia, Egypt.
ayman.elsayed@el-eng.menofia.edu.eg ,
https://orcid.org/0000-0002- 4437-259X

Sahar Elkazzaz,
Computer Science & Eng. Dept., Faculty of Electronic Eng. Menoufia University Menoufia Egypt
Sahar.elkazaz@ejust.edu.eg

***Abstract* – Fog computing is a new computing paradigm that has been proposed to extend cloud computing services to the edges of cloud computing networks. Minimizing the total completion time of an application without violating user-defined deadline is one of the most important problems that are related to task scheduling in fog environments. In this paper, we have proposed a new algorithm called Deadline Aware Resource Allocation (DARA) algorithm. The main contribution of this algorithm is to enhance the performance of fog environment by allocating resources in an efficient manner under deadline constraint. The algorithm is compared with Dynamic Resource Allocation Method (DRAM) algorithm. Simulation results proved that our proposed algorithm provides better performance in terms of makespan, total cost, and resource utilization.**

Keywords* – *Cloud computing, Fog computing, Task scheduling, Resource allocation

1. INTRODUCTION

Cloud computing is a powerful technology that provide many on-demand services to users over the internet. It introduces a lot of features like scalability, flexibility, and performance cost efficiency. Cloud computing is based on virtualization technology [1]. The most important drawback of cloud computing is that cloud data centers are geographically far away from end users. This drawback added some limitations on using cloud computing for time sensitive applications that require low latency. To address the limitations of cloud computing, fog computing has been proposed [2-4].

1.1 Fog computing

Fog computing is a new distributed computing paradigm that was first introduced by Computer Information Systems Corporation "Cisco" in 2012[5]. It acts as an intermediate layer between cloud data centers and end users [6]. Like cloud computing, fog computing is also based on virtualization. Fog computing extends cloud services like data, storage, networking, and computing services closer to end users as shown in Figure 1 [7-9]. So, fog is best suited for real time applications [10]. Fog helps to overcome the limitation of cloud by providing real-time and low latency services. So, fog computing doesn't replace cloud computing, but they complement each other [11-14].

1.2 Task Scheduling and Resource Allocation In Fog Environment

Fog computing environment consists of a set of heterogeneous resources with different capabilities. So, task scheduling is an important issue to specify which resource best fit to which task. The main objective of task scheduling in fog computing is to map tasks to the available resources and determine the order of execution of these tasks in order to minimize the total execution time (makespan). Task scheduling is classified into two classes according to the dependencies between tasks: (1) dependent task scheduling, (2) independent task scheduling. In dependent task scheduling, there are dependency relationship and communication between tasks [15]. On the other side, there are no dependency relationship and communication between tasks in independent task scheduling [16], [17]. Effective scheduling techniques are required to optimize and enhance the overall performance of computing systems.

In fog computing, there are limited physical resources in terms of storage, memory and processors

that are required to serve many user tasks or requests [18].

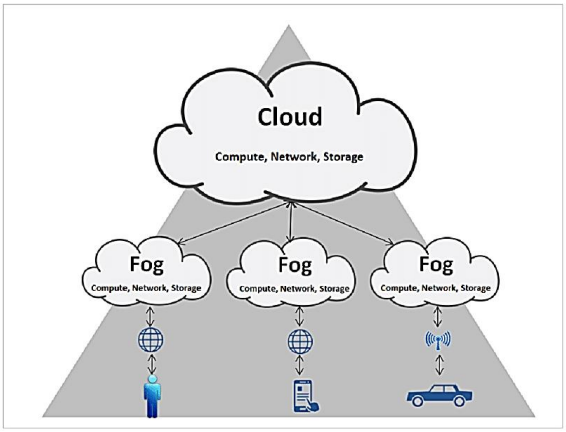

Figure 1. Hierarchal architecture of fog computing [10].

Consequently, efficient resource allocation is required to achieve highest system throughput and maximum profit. Resource allocation is defined as a systematic approach to allocate available resources to the clients over the internet [2].

The main contribution of this paper is to assign tasks of customers to the available resources of fog computing environments in a prioritized fashion to minimize the completion time, minimize total cost, and maximize resource utilization based on deadline constraint.

This paper is organized as follow: Related work is presented in Section 2. Problem definition is illustrated in Section 3. Our proposed algorithm is descriped in Section 4. Experimental results are presented and discussed in Section 5. Finally, both conclusion & future work are presented in Section 6.

2. RELATED WORK

Task scheduling and resource allocation are important issues in fog computing. Efficient task scheduling and resource allocation algorithm will help to increase the overall performance of the system. Recently, much research has discussed these issues. In [2], they proposed a three-layered architecture, and designed an efficient algorithm called efficient resource allocation (ERA) for resource provisioning in fog computing. The architecture is based on a system model where a fog layer is used between the end-user clients and the cloud datacenter. In [19], the authors proposed a priority-based task scheduling algorithm in fog computing. Their algorithm enhanced the ERA algorithm with priority scheme to reduce both the average response time and the total cost. In [20] the authors proposed an algorithm for load balancing in cloud environment called dynamic resource allocation method (DRAM). DRAM tends to minimize the load-balance variance, which is relevant to the resource utilization of each computing node, and the average resource utilization. In [21], they proposed a scheduling algorithm called Cost-Makespan aware Scheduling (CMaS) heuristic to achieve the balance between the performance of application execution and the mandatory cost for the use of cloud resources. Additionally, an efficient task reassignment strategy is also proposed to refine the output schedules of the CMaS algorithm to satisfy the user-defined deadline constraints. In [22] it is proposed that a new fog computing architecture, which is divided into three layers. Then, a systematic two-level resource scheduling model is presented. Finally, a novel resource scheduling scheme was proposed using an improved non-dominated sorting genetic algorithm II (NSGA-II) with the aim to reduce the service latency and improve the overall stability of task execution. In [23], they proposed a model to effectively schedule the user tasks on the fog computing resources by combining the VM allocation and VM selection methods in the perfect arrangement. Various methods associated with VM allocation and VM selection are evaluated and combined in a suitable combination to discover the best task scheduling combination for the effective and optimized user data processing. In [24] the authors aimed to provide an easy and concise view of the High-Performance Computing (HPC) algorithms. Firstly, they presented the classification of scheduling algorithms based on multiple factors like fairness, waiting time, throughput, overhead, etc. Secondly, the forecasting has been done on HPC applications to predict the growth rate for 2020 and beyond. The authors in [25] proposed a task scheduling strategy based on a hybrid heuristic (HH) algorithm that mainly solves the problem of terminal devices with limited computing resources and high energy consumption and makes the scheme feasible for real-time and efficient processing tasks of terminal

devices. HH algorithm combines the advantages of improved particle swarm optimization (IPSO) and improved ant colony optimization (IACO) to search for the optimal solution for task scheduling problem in smart production lines with fog computing. In [26], it is proposed a multi-cloud to multi-fog architecture and design two kinds of service models by employing containers to reduce the service delay improve the resource utilization of fog nodes and. Based on these models they presented a task scheduling algorithm for energy balancing which uses a dynamic threshold strategy to schedule requests in real time. In [27], they proposed a new orchestration of Consumer to Fog to Cloud (C2F2C) based framework for efficiently managing the resources in residential buildings. It consists of three layers. Cloud layer which deals with on-demand delivery of the consumer's demands. Fog layer that is responsible for Resource management. Consumer layer which is based on the residential users and their electricity demands from the six regions of the world. These regions are categorized on the bases of the continents. In [28], the authors used a framework, including three parallel algorithms, namely, offloading, buffering, and resource allocation, to improve resource allocation balance, throughput, and task completion ratio. They considered a fog queuing system with limited infrastructure resources to accommodate real-time tasks with heterogeneities in task types and execution deadlines. In [29] highlighted key features of iFogSim along with providing instructions to install it and simulate a Fog environment. Also, they demonstrated how to implement custom application placement in iFogSim simulated Fog environment along with an IoT-enabled smart healthcare case study. in [30], it is designed novel resource allocation algorithms for the Social Internet of Things (SFIoT) system. They adopt the basic concept of two game models: voting and bargaining games to formulate the interaction among mobile devices and FC operator. Bicooperative voting game (BVG) approach is responsible for control decisions for the resource allocation method, and Nash bargaining solution (NBS) is responsible for distributing the computation resource to different application tasks. The author in [31] investigated the computation resource allocation and task assignment problem in VFC from a contract matching integration perspective. A contract-based incentive mechanism was proposed to motivate vehicles to share their resources, and a pricing-based stable matching algorithm was developed to address the task assignment problem. In [32] it is considered the resource allocation and task scheduling problem under fog system to minimize total tardiness of the tasks and meet the hard deadlines. A deadline-aware estimation of distributed algorithm (dEDA) with a repair procedure and local search is adopted to determine the task processing order and computing node allocation. In [33], the authors proposed an efficient centralized secure architecture for healthcare system deployed in Cloud environment. Fog Computing environment was used to run the framework. First, health data is collected from sensors and sent to the near edge devices. Finally, devices transfer the data to the cloud for seamless access by healthcare professionals. The main focus of this work is the security as Authentication and Authorization of all the devices. The proposed system uses asynchronous communication between the applications and data servers deployed in the cloud environment. In [34] they investigated the research challenges in Fog Computing. It promoted a lot of research in the area of Fog Computing application.

All previous studies concluded that resource provisioning and allocation is the most important issue in fog computing that can affects the processing time of tasks because improper resource allocation can lead to degrading the performance of the system. However, the previous studies rarely considered QoS parameter such as task deadline which is essential for real time tasks. This paper differs from previous studies in its contribution which is minimizing the completion time and maximizing resource utilization simultaneously under the deadline constraint to satisfy the demands of the user and improve QoS.

3. PROBLEM DEFINITION

3.1 Fog Computing Architecture

Fog computing adds an extra fog layer between cloud and end devices (end users). As shown in Figure 2, the system model consists of three layers: cloud layer, fog layer and client layer. Cloud layer (top layer) consists of a set of cloud data centers. Client layer (bottom layer) consists of end devices, which send requests to the upper layers for application

execution. Fog layer (middle layer) consists of a set of fog nodes or fog servers. Each fog node consists of several virtual machines (VMs). Each VM contains various physical resources including CPU, memory, storage, and bandwidth. Our proposed algorithm is implemented at the fog layer. In the fog layer, there is a fog device called fog broker that acts as a resource manager and task scheduler. Fog broker is responsible for receiving requests or tasks from users, managing the available resources, and generating the most suitable schedule to specify which task will be executed on which resource.

3.2 Problem Statement

In fog computing, scheduling means assigning the available resources to user requests or tasks in a specified order to satisfy user requirements and quality of service (QoS) needed. One of the most important parameters of QoS is deadline. In our problem, we focused on allocating resources to tasks considering user-defined deadline constraints.

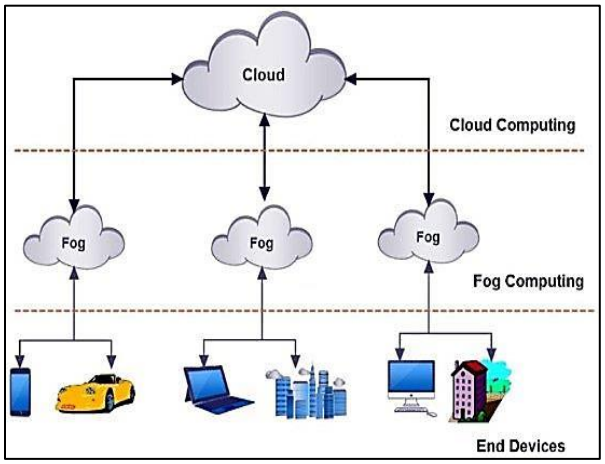

Figure 2. Fog computing architecture [36].

Fog layer consists of *S* number of fog servers denoted as S_1, S_2, …, S_s. Each fog server consists of *M* number of VMs denoted as p_1, p_2, …, p_m. Each VM has its own resources and its own speed (denoted by sp_i. sp_i is measured by the number of millions of instructions per second (MIPS). Let $N = \{T_1, T_2, \ldots, T_n\}$ represents the number of independent tasks to be executed in fog. The problem can be stated as follows: a set of *N* independent tasks will be executed on *M* virtual machines considering the deadline constraint *d*. We aim to minimize the completion time of the task, minimize the total cost of resource usage as well as maximize the resource utilization under deadline constraint defined by the user for each task.

4. PROPOSED ALGORITHM

Each task has different properties such as deadline, length, and execution time. In task scheduling, deadline is considered one of the most important parameters for task execution, which affects QoS of the system. The focus of DRAM [20] algorithm was on minimizing the load balance variance and maximizing resource utilization without considering the effectiveness of the makespan and deadline parameters. DARA algorithm aims to maximize resource utilization and makespan taking into consideration deadline. Users submit tasks with deadline constraint for each task to the fog broker. Then, the broker will assign these tasks to the available resources according to the proposed algorithm.

The execution time of a task can be calculated by Equation 1[35].

$$ET = \frac{r}{sp_i} \quad (1)$$

Where sp_i is the VM's speed, r is the task's length. On each VM, EET (Expected Execution Time) of the task is calculated by using Eq. 1 and compared with the deadline constraint. The VM which meet the deadline constraint will be labeled as a valid VM. Otherwise, it is labeled invalid VM as illustrated in Eq. 2.

$$VM`s\ state = \begin{cases} Valid & if\ EET \leq deadline \\ Invalid & otherwise \end{cases} \quad (2)$$

Then, the task will be assigned to one of the valid VMs, which provides the least and enough requirements based on the task type. For example: In case of memory task, the task will be assigned to a valid VM that has the least and enough memory for executing the task. In case of storage task, the task will be assigned to a valid VM that introduce the least and enough storage for the task. The same approach is applied for other types of tasks.

Finally, a factor called deadline is violated (*DIV*) will be used to express whether the task can be executed before its deadline or not. *DIV* is a binary factor which has two values 1, 0. *DIV(T)* = 1, if the deadline of a task *T* is violated. In this case, the task

will be migrated to another fog server. *DIV(T)* = 0, if the deadline of task *T* is achieved the task will be assigned to a specific VM and removed from the tasks ready queue.

4.1 Steps of Proposed Algorithm

Input: set of unmapped independent tasks with deadline.
Output: minimize the makespan and the total cost based on deadline constraint.

1- Categorize tasks into groups based on the task type.
2- In each group, sort tasks in descending order.
3- Assign priority to each group based on the availability of VMs
4- Select the highest priority group
5- For $j = 1$ to n //n is the number of tasks in each group
6- //check VM state:
For $i = 1$ to m
Calculate the expected execution time of task (EET)
If $EET_T \leq$ task deadline
VM state = valid
Count ++
Else
VM state = invalid
End for
7- If count > 0
DIV = 0
Find a specific successful VM that provide the least and enough requirements for the task based on the task type.
Else
DIV = 1
Migrate task to another fog server
8- Map task to a specific VM.
End for //all tasks are mapped.

5. SIMULATION AND EXPERIMENTAL RESULTS

5.1 Simulation Environment

A simulation environment that simulates fog environment has been built to evaluate the performance of DARA algorithm. Visual C# .NET 4.0 is used to build the simulator on machine with: Intel(R) Core(TM) i3 CPU M 350 @2.27GHz, RAM of 8.00 GB, and the operating system is window 10, 64-bit.

Each fog node has different processing capabilities. We assumed that each fog node consists of two virtual machines. Each VM has its own processing power that is measured by MIPS (Millions of Instructions per Second) along with memory, capacity, and bandwidth. The characteristics of fog nodes are shown in Table 1. Each task has different attributes which are shown in Table 2.

Six data sets have been used in our experiment with variable size from 500 tasks to 3000 tasks. Each task in data sets was generated randomly in the range mentioned in Table 2. The experiment covered two types of tasks: capacity tasks (that require huge amount of storage capacity), and memory tasks (that require more memory).

Table 1. Characteristics of fog nodes

Parameter	Value
Number of fog nodes	3,4,5
Number of VMs in each node	2
Computation power of VM	[10, 200]
Storage capacity of VM	5000-10000
Memory of VM	5000-10000
Memory Usage Cost	0.01-0.03
Storage Usage Cost	0.01-0.03

Table 2. Attributes of Tasks

Attribute	Value
Number of tasks	{500, 1000, 1500, 2000, 2500, 3000}
Arrival Time	[0, 20]
Deadline	[2, 10]
Length	[5, 50]
Required Capacity	[5, 50]
Required Memory	[5, 50]

5.2 PERFORMANCE EVALUATION PARAMETERS

We used some parameters to evaluate the performance of the proposed algorithm in fog computing environment. The considered parameters are:

5.2.1 Makespan

Makespan also called "schedule length" is defined as the maximum finish time of last task executed on the VMs or the time when the last machine finishes. It begins from the time the request is received to the time that the last task is completed. To achieve higher performance, makespan should be minimized. It can be calculated by Equation 3[39].

$$\text{Makespan} = \text{Max}\,[CT(P_i)] \quad (3)$$

Where *CT* is the completion time, i ϵ VMs (1≤ i ≤ m)

5.2.2 Response Time

Response time (*RT*) is the time taken by a task to complete the execution [37]. In other words, it is the elapsed time between submission and completion time of task. It can be calculated by Equation 4 [40].

$$RT = CT_j - SB_j \quad (4)$$

Where j ϵ T (1≤ j ≤ n). *CT* is the completion time; SB is the submission time.

To calculate the average response time for all tasks on one VM, Equation 5 is applied.

$$Avg.RT = \frac{\sum_{j=1}^{n} RT}{n} \quad (5)$$

Then, the mean of total average response time of all VMs is calculated by using Equation 6.

$$\text{Mean of total Avg. RT} = \frac{\sum_{i=1}^{m} Avg.RT}{m} \quad (6)$$

Where *n* is the number of tasks in VM and *m* is the number of VMs.

5.2.3 Throughput

Throughput is the no of tasks completed per unit time. It reflects the efficiency of the scheduling algorithm. It can be calculated by equation 7 [41].

$$\text{Throughput} = \frac{n}{Makespan} \quad (7)$$

Where *n* is the total number of tasks

5.2.4 Resource utilization (RU)

An important optimization metric is maximizing resource utilization. It is defined as the resource usage of the resource units on the computing nodes. It can be calculated as follow [41]:

$$RU = \frac{\sum_{i=1}^{m} \text{Makespan}_i}{m * \text{Max\_Makespan}} \quad (8)$$

Where i ϵ VMs (1≤ i ≤ m), *max_makespan* can be expressed as:

$$\text{Max\_Makespan} = \max\{\text{Makespan}_i\} \quad (9)$$

Where i ϵ VMs (1≤ i ≤ m)

5.2.5 Load Balancing

Load balancing refers to the process of distributing a set of tasks over a set of resources, with the aim of making their overall processing more efficient. It is calculated by equation 10 [41].

$$\text{Load Balancing} = \frac{\sum_{i=1}^{m} \text{Makespan}_i}{m} \quad (10)$$

Where i ϵ VMs (1≤ i ≤ m)

5.2.6 Total Cost

To calculate the cost of processing a task T_j on a VM P_i, we must calculate the usage cost of resources included in that VM. These resources include CPU (processing), storage, and RAM. In this work, we have calculated only storage and memory cost. The cost of task "*j*" on a VM "*i*" can be expressed by equation 11 [42].

$$\text{Cost}\,(T_j^i) = C_r\,(T_j^i) + C_s\,(T_j^i) \quad (11)$$

In equation 11, each cost can be calculated as follow: RAM cost can be defined as equation 12:

$$C_r\,(T_j^i) = c_1 * \text{RAM}\,(T_j^i) \quad (12)$$

Where c_1 is the RAM usage cost per data unit in VM P_i and RAM (T_j^i) is the RAM required by task T_j . Storage cost can be defined as equation 13:

$$C_s\,(T_j^i) = c_2 * S\,(T_j^i) \quad (13)$$

Where c_2 is the storage usage cost per data unit in VM P_i and S (T_j^i) is the storage required by task T_j.

Finally, the total cost for all tasks executed on the system can be calculated as follows:

$$\text{Total cost} = \sum_{i=1}^{m} \sum_{j=1}^{n} Cost\ (T_j^i) \quad (14)$$

Where i ϵ VMs (1≤ i ≤ m), j ϵ T (1≤ j ≤ n)

5.3 EXPERIMENTAL RESULTS

In our simulation, we compared the evaluation metrics of our proposed algorithm "DARA" with those of DRAM algorithm, with varying the number of VMs from 6 to 10 VMs and varying the number of tasks from 500 to 3000 tasks.

5.3.1 Results on 6 VMs

The results are shown in figure 3, 4, 5, 6, 7, 8. The comparison results of evaluation metrics between DARA, DRAM, and MRR algorithms are listed in table 3, 4.

From figure 3, it is shown that DARA algorithm consumes lesser time to process tasks than DRAM. This means that tasks are properly allocated to the most suitable VMs that satisfy the requirements of task and complete its execution in lesser time.

Table 3. Comparison results of DARA and DRAM algorithms on 6 VMs

No. of Tasks	Total Makespan		Average Response Time		Resource Utilization	
	DARA	DRAM	DARA	DRAM	DARA	DRAM
500	1613	1748	12.500064	13.61795955	0.326203761	0.166666667
1000	2066	2981	8.067044654	11.77713178	0.302355599	0.203007939
1500	3315	3583	8.221315193	9.36502601	0.2942182	0.256954135
2000	3542	4015	6.667184918	7.879491674	0.337756446	0.292320465
2500	2902	4245	4.621646391	6.725723458	0.577188146	0.344837063
3000	4351	4896	5.707701909	6.330559601	0.479391711	0.363596133

Table 4. Comparison results of DARA and DRAM algorithms on 6 VMs

No. of Tasks	Throughput		Load Balancing		Total Cost	
	DARA	DRAM	DARA	DRAM	DARA	DRAM
500	0.309981401	0.28604119	567.5	567.5	613302.85	597935.18
1000	0.484027106	0.3354579	1171.833333	1171.833333	620397.45	604905.58
1500	0.452488688	0.418643595	1753	1753	633209.45	617677.78
2000	0.564652739	0.498132005	2301.5	2301.5	654435.05	638922.18
2500	0.861474845	0.588928151	2878.666667	2878.666667	686209.65	670737.18
3000	0.689496667	0.612745098	3444	3486.166667	728197.05	712395.58

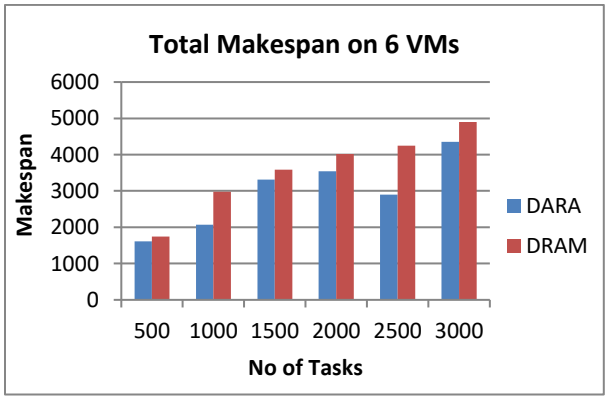

Figure 3. Comparison of Total Makespan

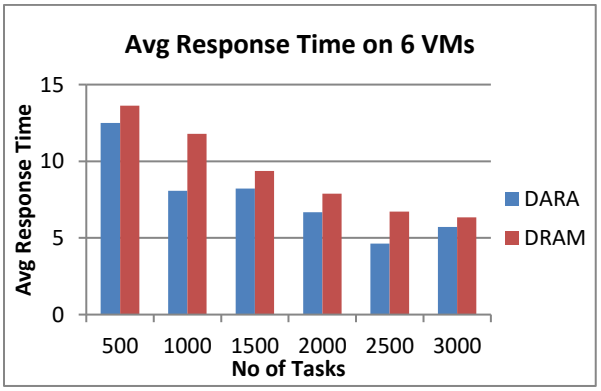

Figure 4. Comparison of Average Response Time

Figure 4 represents the average response time comparison. We can see from the figure that, DARA algorithm process tasks with lesser response time than DRAM algorithm. That makes DARA algorithm best suited for real time applications than DRAM algorithm. Resource utilization results shown in figure 5 demonstrates that, DARA algorithm provide better resource utilization than DRAM algorithm which means that, majority of resources have been allocated.

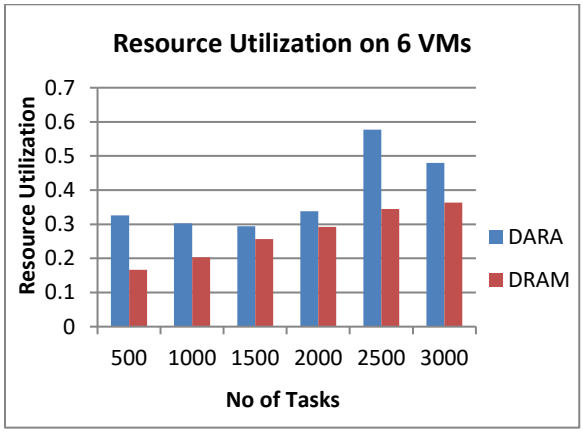

Figure 5. Comparison of Resource Utilization

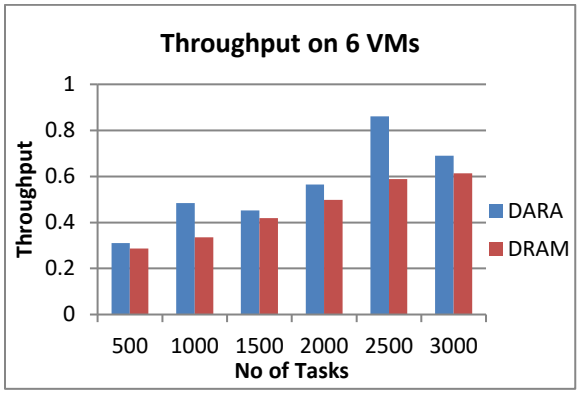

Figure 6. Comparison of Throughput

In figure 6, it is seen that DARA algorithm provides better throughput than DRAM algorithm. This means that DARA can process more tasks than DRAM in one unit time which results in improving the overall performance of the system.

Figure 7, 8 show the load balancing and total cost results. We can see that the two algorithms provide the same results or close results. This shows that, we have improved the other performance metrics while maintaining the total cost and load balancing parameters as stable as possible.

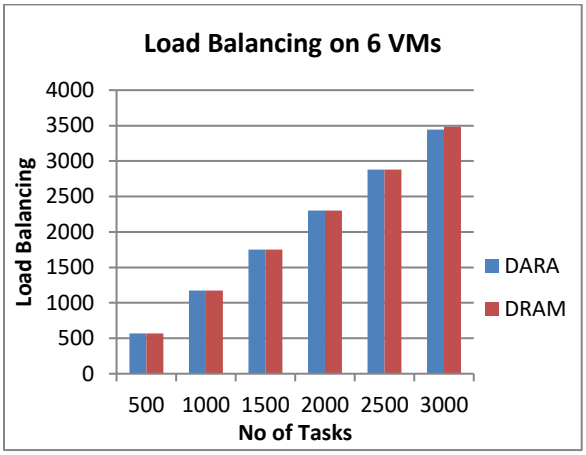

Figure 7. Comparison of Load Balancing

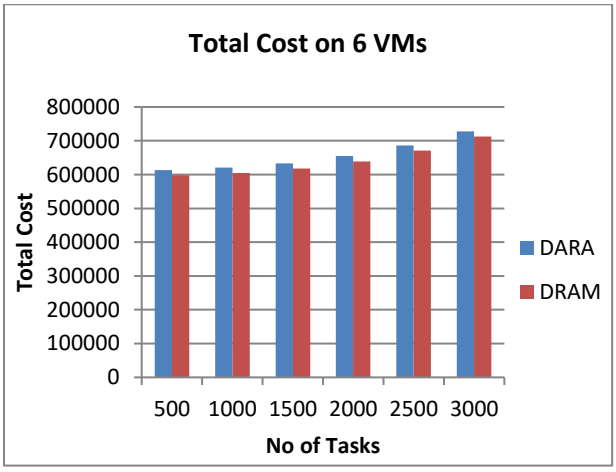

Figure 8. Comparison of Total Cost

5.3.2 Results on 8 VMs

The results are shown in figure 9, 10, 11, 12, 13, 14. The comparison results of evaluation metrics between DARA, DRAM, and MRR algorithms are listed in table 5, 6.

5.3.3 Results on 10 VMs

The results are shown in figure 15, 16, 17, 18, 19, 20. The comparison results of evaluation metrics between DARA, DRAM, and MRR algorithms are listed in table 7, 8.

Table 5. Comparison results of DARA and DRAM algorithms on 8 VMs

No. of Tasks	Total Makespan		Average Response Time		Resource Utilization	
	DARA	DRAM	DARA	DRAM	DARA	DRAM
500	1100	1526	7.4820	11.5958	0.2313	0.1480
1000	1778	2005	6.4829	7.9063	0.2605	0.2262
1500	2270	2420	5.3171	6.2742	0.3794	0.2851
2000	1931	2643	3.5724	5.1750	0.4710	0.3377
2500	2488	2771	3.9312	4.3935	0.4581	0.3985
3000	2703	3222	3.6754	4.1929	0.4795	0.4092

Table 6. Comparison results of DARA and DRAM algorithms on 8 VMs

No. of Tasks	Throughput		Load Balancing		Total Cost	
	DARA	DRAM	DARA	DRAM	DARA	DRAM
500	0.4545	0.3276	425.125	425.125	268582.35	256274.17
1000	0.5624	0.4987	878.375	878.375	279034.95	266753.37
1500	0.6607	0.6198	1,313.75	1,313.75	300243.95	288011.57
2000	1.0357	0.7567	1,724.625	1,724.625	315780.75	303494.77
2500	1.0048	0.9022	1,984.25	2,157.25	335729.95	323437.07
3000	1.1098	0.9310	2,201.25	2,579.125	360727.65	348347.47

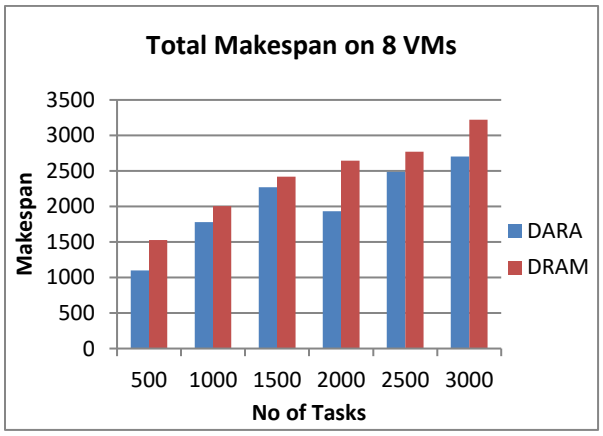

Figure 9. Comparison of Total Makespan

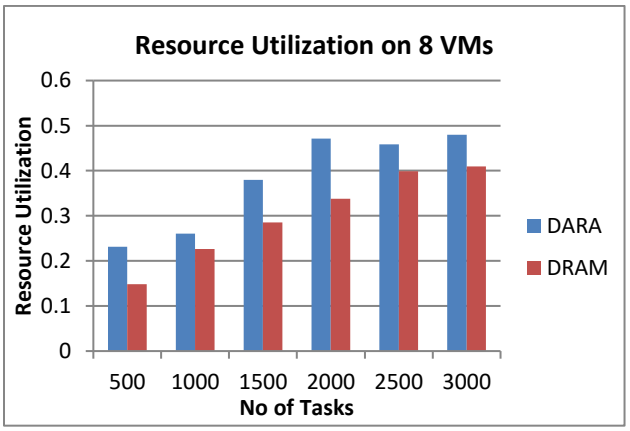

Figure 11. Comparison of Resource Utilization

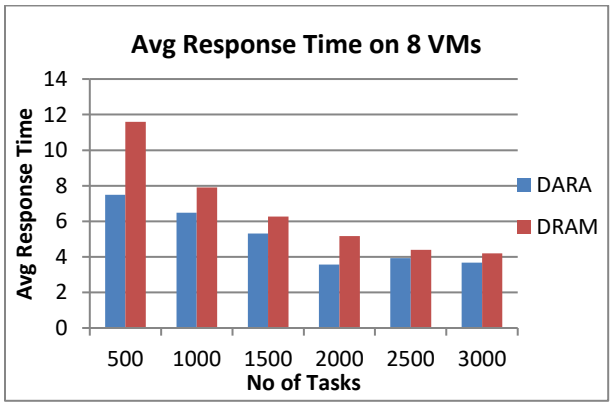

Figure 10. Comparison of Average Response Time

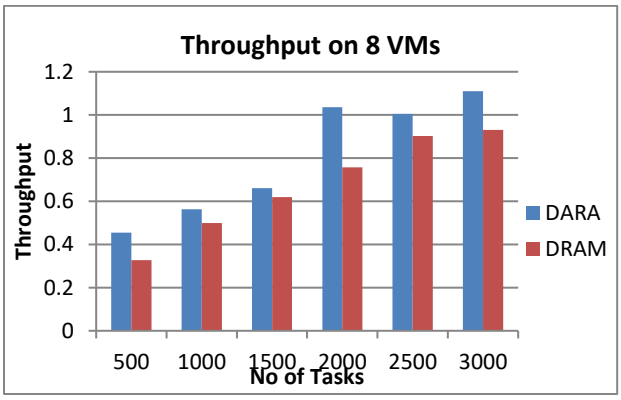

Figure 12. Comparison of Throughput

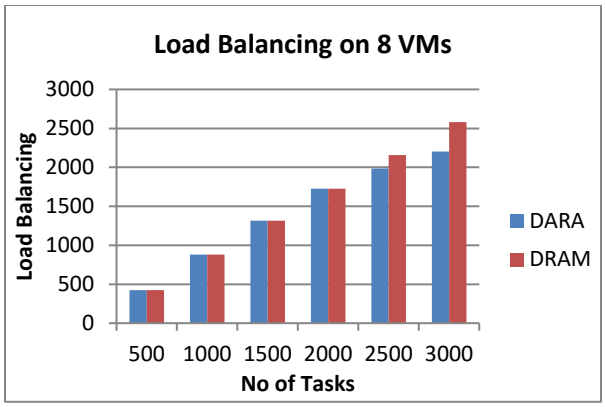

Figure 13. Comparison of Load Balancing

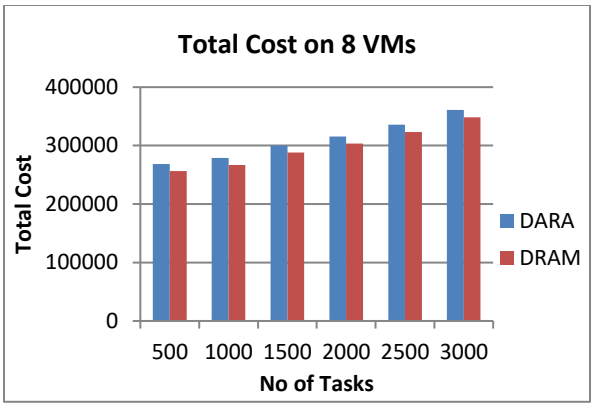

Figure 14. Comparison of Total Cost

Table 7. Comparison results of DARA and DRAM algorithms on 10 VMs

No. of Tasks	Total Makespan		Average Response Time		Resource Utilization	
	DARA	DRAM	DARA	DRAM	DARA	DRAM
500	1100	1526	7.482014849	11.59587814	0.185090909	0.118414155
1000	1778	2005	6.482926517	7.906320072	0.208436445	0.180997506
1500	2351	2420	5.299293051	6.274216353	0.264142918	0.228099174
2000	1954	2643	3.597008952	5.17506273	0.371801433	0.270185395
2500	2536	2771	4.043187311	4.393547247	0.358359621	0.318837965
3000	2796	3222	3.445009154	4.141423032	0.395028612	0.333022967

Table 8. Comparison results of DARA and DRAM algorithms on 10 VMs

No. of Tasks	Throughput		Load Balancing		Total Cost	
	DARA	DRAM	DARA	DRAM	DARA	DRAM
500	0.454545455	0.327653997	340.1	340.1	163129.27	145373.87
1000	0.562429696	0.498753117	702.7	702.7	173581.87	155853.07
1500	0.638026372	0.619834711	1030.1	1051	194790.87	177111.27
2000	1.023541453	0.756715853	1151.8	1379.7	225864.47	208077.67
2500	0.985804416	0.902201371	1403	1725.8	245786.97	228019.97
3000	1.072961373	0.931098696	1737.7	2090.1	271110.27	253300.37

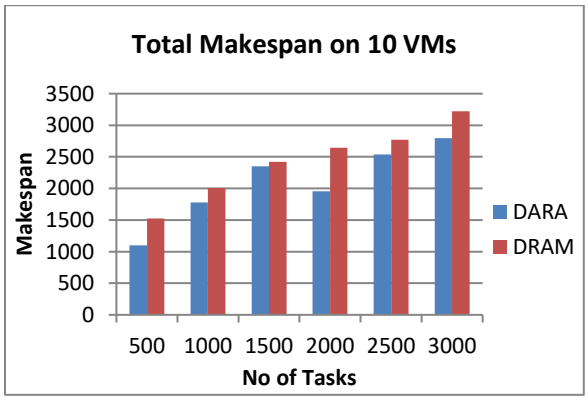

Figure 15. Comparison of Total Makespan

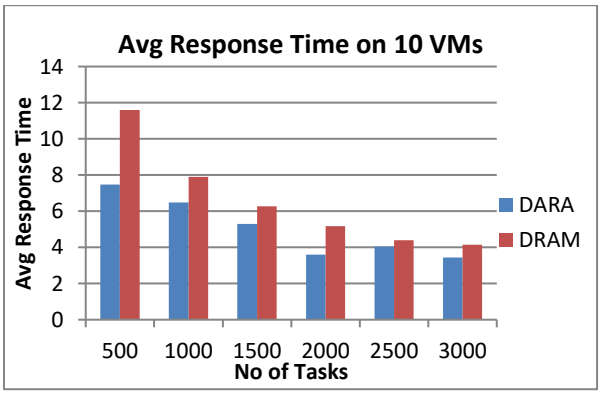

Figure 16. Comparison of Average Response Time

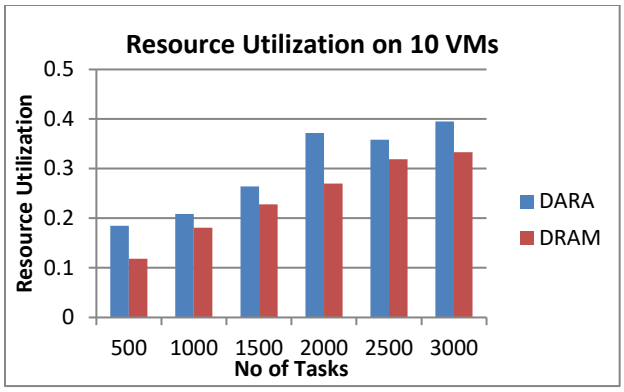

Figure 17. Comparison of Resource Utilization

DARA algorithm is based on dividing tasks according to its type resulting in minimizing the completion time and maximizing resource utilization of the system. Also, executing tasks with higher requirements first resulted in higher throughput and lesser response time. On the other side, taking the deadline of tasks into consideration improve the overall performance of the system.

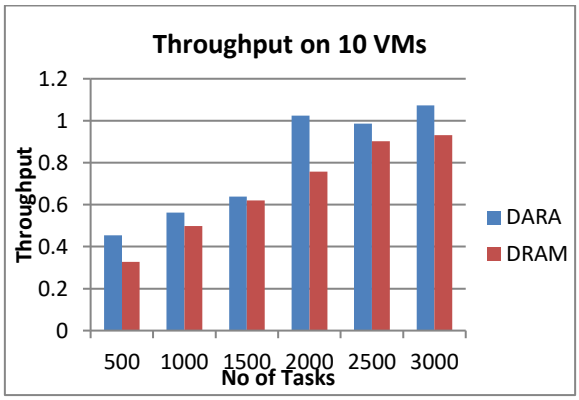

Figure 18. Comparison of Throughput

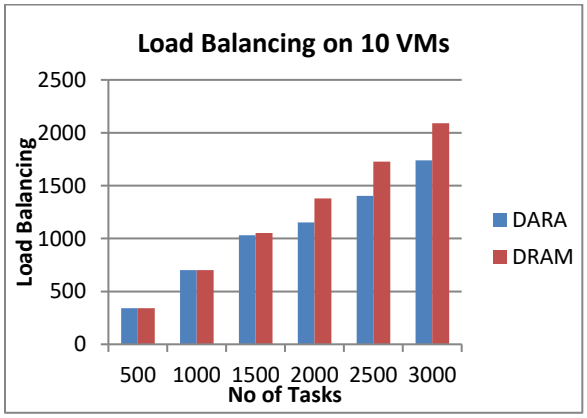

Figure 19. Comparison of Load Balancing

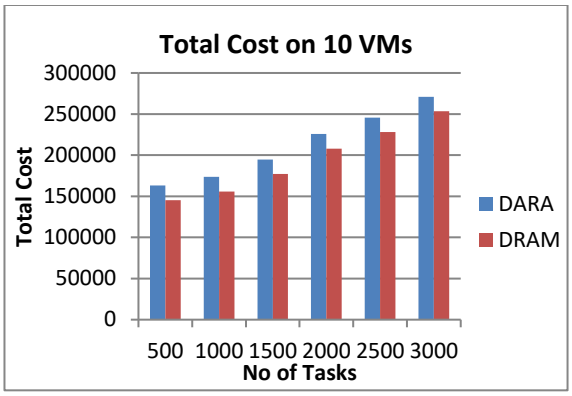

Figure 20. Comparison of Total Cost

6. CONCLUSION AND FUTURE WORK

Fog computing is an emerging computing paradigm that brings cloud services nearest to the users. Efficient resource allocation is a key issue which affects the overall performance in terms of total completion time of the application, resource utilization, and the total cost of consuming resources. In this paper, we proposed DARA algorithm that efficiently allocate the application tasks on the available resources under deadline constraint. It can be implemented in the fog layer. DARA is suitable for real-time and latency sensitive applications. Due to few resources available in our experiment, a simulator has been built to evaluate the performance of DARA algorithm against DRAM algorithm. The results showed that DARA provide better performance than DRAM in terms of makespan, resource utilization, throughput, and the average response time while maintaining the total cost of using resources and load balancing as stable as possible. From the results we can see that the total improvement ratio of makespan is approximately 16% while increasing the number of tasks and the number of VMs. In the future, we can take into consideration other QoS constraints like user defined budget to enhance the performance of the system. On the other side, we can enhance DARA algorithm to improve the results of cost and load balancing. We can also apply it on other simulators like iFogSim.

REFERENCES

[1] Kamyab Khajehei, “Role of virtualization in cloud computing”, International Journal of Advance Research in Computer Science and Management Studies, Volume 2, Issue 4, April 2014.

[2] S. Agarwal, S. Yadav, and A.Yadav, "An Efficient Architecture and Algorithm for Resource Provisioning in Fog Computing", in MCEP, 2016, doi: 10.5815/ijieeb.2016.01.06

[3] J. Xu, B. Palanisamy, H. Ludwig, and Q. Wang, "Zenith: Utility-aware Resource Allocation for Edge Computing", IEEE International Conference on Edge Computing., 2107, 10.1109/IEEE.EDGE.2017.15

[4] Monika Gupta, "Fog Computing Pushing Intelligence to the Edge", International Journal of Science Technology & Engineering `IJSTE`, Volume 3, February 2017.

[5] Bonomi, F.; Milito, R.; Zhu, J.; Addepalli, S. Fog computing and its role in the internet of things. In Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing-MCC '12, Helsinki, Finland, 17 August 2012; pp. 13–15. 2342513.

[6] Redowan Mahmud, Ramamohanarao Kotagiri and Rajkumar Buyya, "Fog Computing: A Taxonomy, Survey and Future Directions", Springer Nature Singapore Pte Ltd. 2018.

[7] Z. Ning, J. Huang, X. Wang, Vehicular fog computing: Enabling real-time traffic management for smart cities, IEEE Wireless Communications 26 (1) (2019) 87–93 (2019).

[8] S. K. Goyal, M. Singh, Adaptive and dynamic load balancing in grid using ant colony optimization, International Journal of Engineering and Technology 4 (4) (2012) 167–174 (2012).

[9] B. Donassolo, I. Fajjari, A. Legrand, P. Mertikopoulos, Fog based framework for iot service provisioning, in: 2019 16th IEEE Annual Consumer Communications & Networking Conference (CCNC), IEEE, 2019, pp. 1–6 (2019).

[10] Ranesh Kumar Naha, Dimitrios Georgakopoulos, Prem Prakash Jayaraman, and Yong Xiang," Fog Computing: Survey of Trends, Architectures, Requirements, and Research Directions", IEEE Access • August 2018.

[11] Huang CY and Xu K. Reliable realtime streaming in vehicular cloud-fog computing networks. In: 2016 IEEE/ CIC international conference on communications in China (ICCC), Chengdu, China, 27–29 July 2016, pp.1–6. New York: IEEE.

[12] Masip -Bruin X, Marn-Tordera E, Alonso A, et al. "Fog to- cloud Computing (F2C): the key technology enabler for dependable e-health services deployment", 2016 Mediterranean ad hoc networking workshop (Med-Hoc-Net), Vilanova i la Geltru´ , 20–21 June 2015, pp.1–5. New York: IEEE.

[13] Lin Y and Shen H. Leveraging fog to extend cloud gaming for thin-client MMOG with high quality of experience. In: 2015 IEEE 35th international conference on distributed computing systems, Columbus, OH, 29 June–2 July 2015, pp.734–735. New York: IEEE.

[14] Deng R, Lu R, Lai C, et al. "Optimal workload allocation in fog-cloud computing toward balanced delay and power consumption". IEEE Internet Things 2016; 3(6): 1171–1181.

[15] K. Chronaki, A. Rico, M. Casas et al., "Task scheduling techniques for asymmetric multi-core systems," IEEE Transactions on Parallel and Distributed Systems, vol. 28, no. 7, pp. 2074–2087, 2017.

[16] G. Lucarelli, F. Mendonca, and D. Trystram, "A new on-line method for scheduling independent tasks," in Proceedings of the 17th IEEE/ACMInternational Symposium on Cluster, Cloud and Grid Computing, (CCGRID '17), pp. 140–149, Spain, May 2017.

[17] J. Wu and X.-J. Hong, "Energy-Efficient Task Scheduling and Synchronization for Multicore Real-Time Systems," in Proceedings of the IEEE 3rd international conference on big data security on cloud, pp. 179–184, China, May 2017.

[18] T. Wauters, B. Volckaert, and F. De Turck, "Fog Computing: Enabling the Management and Orchestration of Smart City Applications in," 2018.

[19] Tejaswini Choudhari, Melody Moh, and Teng-Sheng, "Prioritized Task Scheduling in Fog Computing", ACMSE 18 Proceeding of the ACMSE Conference, Article No.22, 2018.

[20] Xiaolong Xu , Shucun Fu, Qing Cai, Wei Tian, Wenjie Liu ,Wanchun Dou , Xingming Sun , and Alex X. Liu, "Dynamic Resource Allocation for Load Balancing in Fog Environment", Hindawi, Wireless Communications and Mobile Computing, Volume 2018, Article ID 6421607, 15 pages.

[21] Xuan-Qui Pham, Nguyen Doan Man, Nguyen Dao Tan Tri, Ngo Quang Thai and Eui-Nam Huh, "A cost- and performance-effective approach for task scheduling based on collaboration between cloud and fog computing", International Journal of Distributed Sensor Networks 2017, Vol. 13(11).

[22] Yan Sun, Fuhong Lin, and Haitao Xu, "Multi-objective Optimization of Resource Scheduling in Fog Computing Using an Improved NSGA-II", Wireless Pers Commun (2018), Springer Science + Business Media, LLC, part of Springer Nature 2018.

[23] Simar Preet Singh and Anand Nayyar, "Dynamic Task Scheduling using Balanced VM Allocation Policy for Fog Computing Platforms", Scalable Computing: Practice and Experience, Volume 20, Number 2, pp. 433–456, May 2019.

[24] S. Razzaq, A.Wahid, F. Khan, N. ul Amin, M. A. Shah, A. Akhunzada, I. Ali, Scheduling algorithms for high-performance computing: An application perspective of fog computing, in: Recent Trends and Advances in Wireless and IoT-enabled Networks, Springer, 2019, pp. 107–117 (2019).

[25] J. Wang, D. Li, Task scheduling based on a hybrid heuristic algorithm for smart production line with fog computing, Sensors 19 (5) (2019) 1023 (2019).

[26] J. Luo, L. Yin, J. Hu, C. Wang, X. Liu, X. Fan, H. Luo, Container-based fog computing architecture and energy-balancing scheduling algorithm for energy iot, Future Generation Computer Systems (2019).

[27] S. Javaid, N. Javaid, T. Saba, Z. Wadud, A. Rehman, A. Haseeb, Intelligent resource allocation in residential buildings using consumer to fog to cloud based framework, Energies 12 (5) (2019) 815 (2019).

[28] L. Li, Q. Guan, L. Jin, M. Guo, Resource allocation and task offloading for heterogeneous real-time tasks with uncertain duration time in a fog queueing system, IEEE Access 7 (2019) 9912–9925 (2019).

[29] R. Mahmud, R. Buyya, Modeling and simulation of fog and edge computing environments using ifogsim toolkit, Fog and Edge Computing: Principles and Paradigms (2019) 433–465 (2019).

[30] S. Kim, Novel resource allocation algorithms for the social internet of things-based fog computing radigm, Wireless Communications and Mobile Computing 2019 (2019).

[31] Z. Zhou, P. Liu, J. Feng, Y. Zhang, S. Mumtaz, J. Rodriguez, Computation resource allocation and task assignment optimization in vehicular fog computing: A contract-matching approach, IEEE Transactions on Vehicular Technology (2019).

[32] Chu-ge Wu, Ling Wang, "A Deadline-Aware Estimation of Distribution Algorithm for Resource Scheduling in Fog Computing Systems", Apr 2019, IEEE Congress on Evolutionary Computation (CEC)

[33] Chandu Thota, Gunasekaran Manogaran, Revathi Sundarasekar, Varatharajan R, and Priyan M. K., "Centralized Fog Computing Security Platform for IoT and Cloud in Healthcare System", 2018, IGI Global.

[34] S.Balamurugan , L.Jeevitha, A.Anupriya, and Dr.R.Gokul Kruba Shanker, "Fog Computing: Synergizing Cloud, Big Data and IoT- Strengths, Weaknesses, Opportunities and Threats (SWOT) Analysis", International Research Journal of Engineering and Technology (IRJET), Volume: 03 Issue: 10 | Oct -2016.

[35] Latiff, M. S. A., Syed, H. H. M., & Abdullahi, M. (2016). Fault tolerance aware scheduling technique for cloud computing environment using dynamic clustering algorithm. Neural Computing & Applications.

[36] Madni, S. H. H., Muhammad, S. A. L., & Coulibaly, Y. (2016). An appraisal of meta-heuristic resource allocation techniques for IaaS cloud. Indian Journal of Science and Technology, 9(4), 1–14. doi:10.17485/ijst/2016/ v9i4/80561.

[37] Haidri, R.A., Katti, C.P. & Saxena, P. C. (2014). A load balancing strategy for Cloud Computing environment. In Proceedings of the 2014 International Conference on Signal Propagation and Computer Technology (ICSPCT) (pp. 636-641). IEEE.

[38] Panda, S.K., Gupta, I., Jana, P.K.: Task scheduling algorithms for multi-cloud systems: allocation-aware approach. Inf. Syst. Front. 1–19, 2017.

[39] Mokhtar A. Alworafi, Atyaf Dhari , Asma A. Al-Hashmi , Suresha , A. Basit Darem, "Cost-Aware Task Scheduling in Cloud Computing Environment", I. J. Computer Network and Information Security, 2017,5,52-59.