Cost-effective resource and task scheduling in fog nodes

Ali Hussein Shamman, Hussein Ali Alasadi, Hussein Ali Ameen, Zaid Ibrahim Rasol, Hassan Muwafaq Gheni
Department of Computer Engineering Techniques, Al-Mustaqbal University College, Babil, Iraq

Article Info

Article history:

Received Oct 20, 2021
Revised Apr 9, 2022
Accepted May 19, 2022

Keywords:

Cloud
Execution time
Fog
Internet of things
Task scheduling

ABSTRACT

Cloud services are the cutting edge technology, however the growing demand for the internet of things has certain limitations which are high latency expectation and high cost of cloud resources, and this is caused by long-distance between application and cloud. Fog computing is a distributed extension of the cloud, which provide storage and computation at the network level. It consists of an internet of things (IoT) application, a fog control node, and a fog access node. This research works towards minimizing the cloud cost in scheduling. For this purpose, a cost-effective task and user scheduling algorithm are performed. The first task scheduling model is composed based on composers' roles after that task scheduling algorithm is performed to handle the various task at the fog access node in an optimized manner. Finally, the reallocation mechanism reduces the time and service delay. For the analysis purpose extensive simulation is carried out and performance statistics were compared with other existing algorithms. It was observed that the proposed algorithm provides highly cost-optimized user and task scheduling with better performance statistics and reduces the delay in the task by providing optimization in the concurrent task at the fog node.

Corresponding Author:

Hussein Ali Ameen
Department of Computer Engineering Techniques, Al-Mustaqbal University College
Babil 51001, Iraq
Email: hussain.a.ameen@gmail.com

1. INTRODUCTION

The emergence of cloud computing has expanded dramatically and is becoming increasingly attractive and a center of attraction for the industry as well as academia purpose as it provides services over the internet reduces user requirements for future planning and provides a flexible computing model and business purposes [1]-[3]. With current technological advancements, such as the internet of things (IoT), cyber-physical systems (CPS), and blockchain that are applied to an industrial area, various studies are showing that the IoT market will grow rapidly. It is estimated that the development of the number of IoT devices used in 2025 will reach 64 billion [4]. Using the internet of things internet connectivity is provided beyond mobile phones and laptops towards every possible thing or device that humans use in their delayed routine [3], [5]. Because of that, the need for cloud computing is increasing. Cloud computing is a computing model that makes it possible to access information or data through the internet network comfortably and on-demand to a collection of computing resources that can be configured together [4], [6]. Cloud computing is another stream of internet computing, in which all data is stored on the cloud or internet. These data are accessed by the user or client application by the network layer using the internet. Scalability and high-performance capability provide cost and time affecting benefits to many mobile phones and computers by increasing the data storage capacity and problem-solving at the server end [3], [7]. As the IoT device application increases the service request and response also increases, which is time, cost, and resource in-efficient concerning the cloud. This is caused by the distance between IoT devices and the cloud [8]. Reliable and persistent service of the internet was a challenge in internet transmission. However, it was scaled by the application of router, gateway, and workstation. The problem with IoT devices and the cloud is resolved by using an intermediate between them. The fog server, store the frequently accessed data locally, and hence when the user application requests data, it acts as a cache or proxies. Using this method, frequently used and simple problems can be handled on the fog node, while the updated and the complex problem can be directed towards the cloud for processing. These reduce the latency between service requests from the device to the cloud and handle the large-scale application in a time and resource-effective manner as shown in Figure 1. This clearly explains that fog nodes are proxies of the cloud and do not intend to replace cloud storage and services [9], [10].

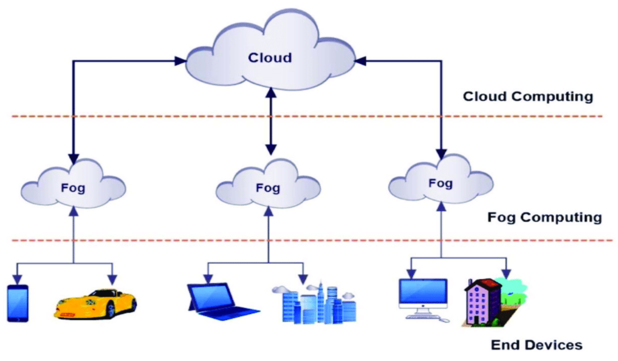


Figure 1. Fog computing model [11]

Large scale applications are comprised of many dependent or depended modules which perform the task in parallel or sequential direction [12], [13]. A similar problem is present in the heterogeneous distributed system. In which many systems act as a module for one main application and use distributed computational power the time complexity of problem-solving is increased [14]. Many types of research have proven the reduce time by applying scheduling algorithms however the resource allocation for this system is high, which is not a concern in the distributed system. But in the case of cloud storage, the cost of cloud services is very high, which should be considered while scheduling tasks in for computing model [15]. The target of Task scheduling in fog computing is minimizing the time by optimizing cloud resources as shown in Figure 2. In this research, a distributed computing technology is considered in which fog is integrated with cloud computation for workflow-based applications. Fog nodes are placed at the router and gateway to extend the cloud node. Cost makespan time and resource scheduling (CMTRS) Algorithm is proposed in this research. The reallocation mechanism reduces the delay in the task by providing optimization in the concurrent task at the fog node. This paper is organized as follows. Section 2 describes the literature survey. In this section, various ongoing and previous research based on task scheduling is discussed. Section 3 describes the problem statement. It contains a detailed problem formulation. Section 4 represents the architecture of the system and proposed algorithm. Section 5 presents the experimental setup and results from analysis and section 6 presents the conclusion.

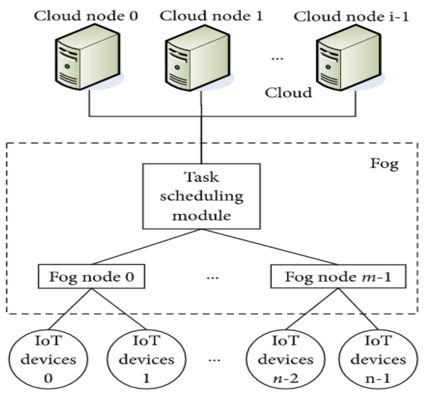


Figure 2. Task scheduling model in fog computing [16]

2. LITERATURE SURVEY

A heterogeneous distributed system acts as different modules for one main application present at various locations and working simultaneously [14], [15]. Many types of research have proven the reduce time by applying scheduling algorithms however the resource allocation for this system is high, which is not a concern in the distributed system [17]. Topcuoglu proposed an algorithm for heterogonous early finish time which is based on the heterogonous system. It consists of two phases, task prioritization, and processor selection [18]. Barbosa proposed a predicted earliest finish time algorithm. This algorithm takes count of processors and tasks to maintain the optimum cost table [19]. Wang also proposed a mechanism for the time-efficient heterogonous system which works in the workflow module. In this insertion-based optimization method is used. However, these techniques were analyzed for heterogonous systems and it was only makespan effective and not cost-effective [14]. Panda proposed a task scheduling algorithm based on profit, this reduces both time and utilization of the cloud services. However, it doesn't consider cloud resource costs [13]. Den applies task scheduling algorithms on public, private, and hybrid clouds. Task priority is mainly based on the deadline constraint available for each application. The limitation of this algorithm was specific service orientation towards the public and private cloud. New concept of sub deadline was proposed and assigned to all tasks. Based on the deadline it is migrated from public to private cloud. But this research fails to address the trade-off of cost and make span [20]. Many recent types of research have been working towards the network and its edge, like routers and gateway [21]. Bonomi studied the application and challenges of fog nodes in cloud integration. In this research, the IoT and fog computing integration were also discussed [22]. Alsaffar proposed architecture for the allocation of resources and delegating the IoT services. Using service request parameter, size, time, and virtual machine capacity [23]. Souza explores the problem related to the quality of services when the cloud is integrated with the cloud [24]. Different researchers worked on simplifying the scheduling issue for the task in fog computing. Zeng in his research proposed a design for the management of resource and time utilization in fog computing [25]. Nan proposed a different strategy that works on average time, adaptive algorithm is used for decisions in the three-tier architecture of fog computing [26]. However, the workflow model applications were not considered in fog-based research. In the proposed research, workflow model application is considered for fog cloud integration. A heuristic algorithm is proposed to minimize the trade-off makespan and resource cost considering the workflow model and deadline constraints.

3. PROBLEM ANALYSIS

In the case of fog nodes, the processing speed is usually less than cloud nodes as cloud services have high RAM, storage, and processing units. However, redirecting a task to a cloud node increases the resource cost as cloud resources are expensive. Hence an unmanaged task assignment strategy reduces the performance of fog-cloud infrastructure.

3.1. Challenges in scheduling the task

Scheduling tasks in an application pool to minimize the time complexity is a challenge in this fog computation. Large-scale applications are comprised of many dependent or independent modules which perform the task in parallel or sequential direction. For parallel tasks there is no dependency on other solutions however in the case of the sequential task, few tasks need input from the previous task for processing. These type of application processing are based on workflows and it is represented in a directed graph, in which nodes presents the task and the edges represent the constraint of precedence. In such an application, steps are based on either user input or the previous step's output. This type of sequential task increases the time requirement and thereby increases the resource requirement. Hence it is very much essential to handle the sequential tasks in a way to minimize the resource and time allocation for the entire application. Task scheduling algorithms are meant to solve this issue. These algorithms assign a task to the processor in such a fashion that the entire time required for processing all tasks gets reduced or minimized.

3.2. Analysis of task scheduling model

Figure 3 shows the example of a large-scale application with a workflow model. In such a task, instead of one single module, the task is divided into several modules to minimize the dependency on a single module. However, when it comes to a dependent task, parallel programming may get affected because of the dependency of one module on the output of another or user-defined inputs. Hence, To maintain the time complexity of the application, various processors are assigned to the task. In this case, the resource cost gets increased which is acceptable in a few application infrastructures but in the case of cloud computing, resource cost is expensive which is why limited processor count is fixed to reduce the cost of cloud usage.

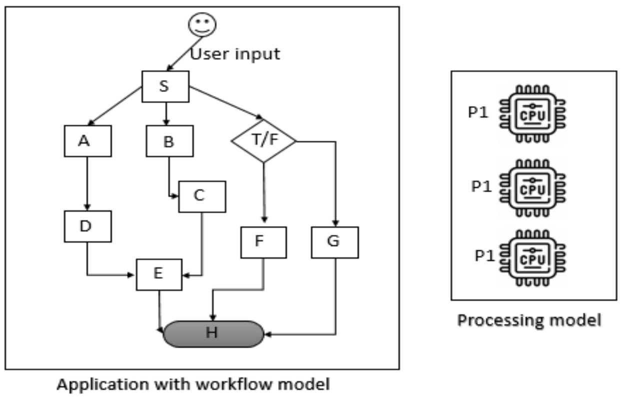


Figure 3. Workflow model, T: Task

To handle both of these issues simultaneously an adaptive method is required to assign a task to the processor in a way that will minimize the time required to produce output. Figure 4, illustrates an example of the task scheduling problem discussed in Figure 3. The scheduling algorithm assigns a task to processors. From Figure 4, it is observed that depending on the priority and the availability of the processor, this algorithm assigns a task. Using this technique, the parallel execution time for all task get reduced to a great extent.

3.3. Goal and objective

This research attempt to schedule all the tasks in the application pool according to the availability of processing units, to reduce the resource cost at the fog-cloud level. The objectives of this research are:

a) Task management at fog and cloud node depending on the task. As fog nodes have the advantage of bandwidth and cloud node has the advantage of processing units and storage.
b) Schedule the Tasks to optimize the usage of the processing unit.
c) Create a task reassignment strategy at fog broker for user-based deadline constraints, to maintain the execution time of the task and improve system performance.

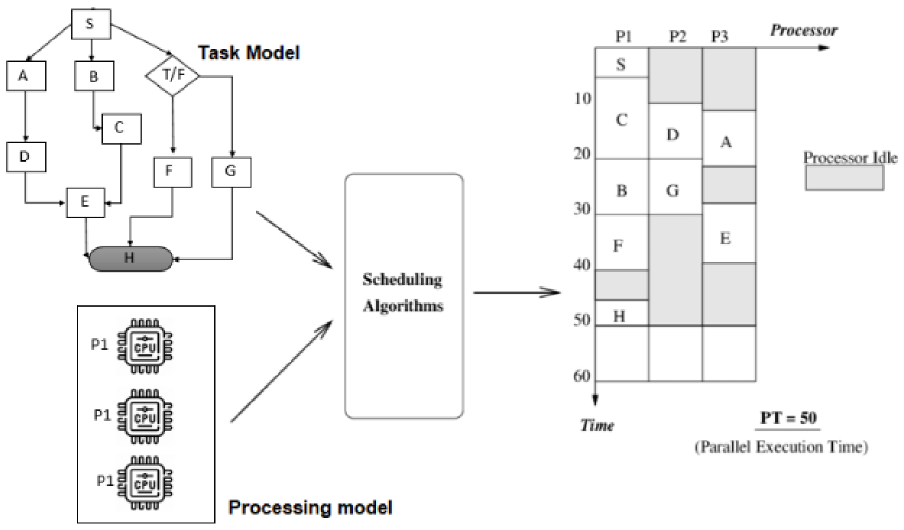


Figure 4. Task scheduling example [27]

3.4. Problem model

The problem formulation of task scheduling consists of the task model and processor model. To describe how the current system implementation assumes that tasks should be performed when interacting with the application. Figure 5 shows illustrate the problem model.

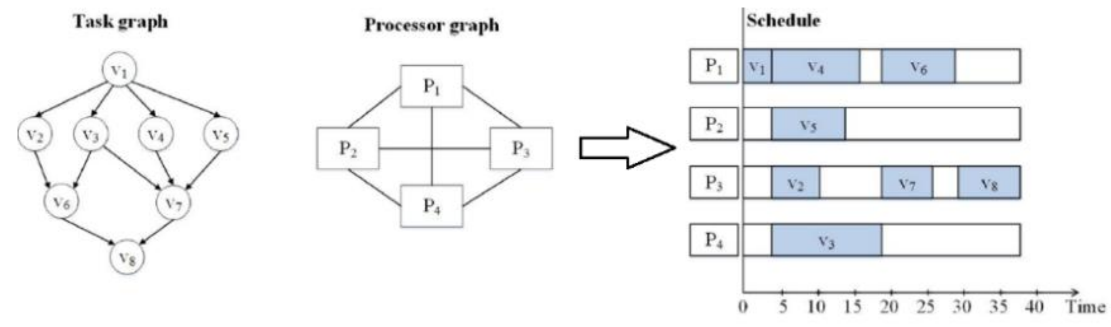


Figure 5. Illustration of problem model [28]

3.5. Task model

In the task model, the tasks model is formulated based on the workflow of the application. Like a flowchart task model is based on vertices and edges. Where vertices set are denoted by V, and it consists of v_1, v_2, $v_3 \ldots, v_n$. Edge defines the relationship between two vertices, and it is denoted as E. For example edge between v_1 and v_2 is defined as $e_{12}=(v_1, v_2)$. Weight is used to define the workload of the task, it is defined as w_1 for v_1. One task could be very small and the other could consist of large computation hence considering weight is a necessary step while scheduling. Along with inputs from the previous step, the time required for gathering data from a data source like a database and user-defined input is also considered.

3.6. Processor model

The processor model is consist of a set of processors. In which each processor is denoted by a vertices Pi, and a set of all vertices is denoted as N. These processors are connected so that the information sharing between the scheduling is carried out like processor speed and acquired bandwidth. In the case of fog nodes, the processing speed is usually less than cloud nodes as cloud services use processors with high random access memory and processing units. While the bandwidth of fog nodes is higher than cloud nodes, as they are placed locally.

4. METHOD

Fog computing is three-layered architecture [15]. This is illustrated in Figure 6. These tiers are the cloud computing layer, fog computing layer, and IoT devices. The lower tier is IoT devices. Fog computing or fog network is a middle layer. Fog computing acts as intermediate using IoT devices and cloud networks. Fog computing is based on the network edge and it provides fast access to IoT devices. It is also called a cloud which is present on the ground, as it is placed very close to the device or end application. In the case of fog computing, instead of passing all the requests to the cloud, it is directed through the fog node or fog server, which is present at the network edge locally. The fog server, store the frequently accessed data locally, and hence when the user application requests data, it acts as a cache or proxies. Using this method, frequently used and simple problems can be handled on the fog node, while the updated and the complex problem can be directed towards the cloud for processing.

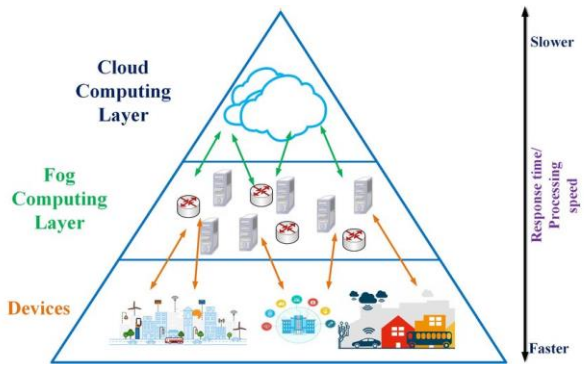


Figure 6. Three-tier architecture of fog computing [29]

The fog computing layer is placed near the IoT devices in the form of gateways or routers. It is present locally. These nodes are referred to as fog nodes which have a limited computing potential. When a user device requests some computation, the fog layer handles it. In the fog layer, fog brokers are present along with fog nodes, which are responsible for the scheduling of tasks and processor and storage management. The broker receives the request from fog nodes. Then within the internal structure the processing speed, network coverage, and bandwidth is analyzed for management. It also computes the resource cost and makespan for the task and returned it to the fog node. Figure 7 illustrates the internal structure of the fog computing layer. It consists of fog resources and a fog broker. Fog resources contain many fog nodes while fog broker contains resource collector, application receiver, data query, application scheduler, application database, and task dispatcher.

4.1. Task scheduling model

To implement the task scheduling of fog computing effectively, the task prioritizing Phase, node selection Phase, and task reassignment phase are integrated into the task scheduling process of fog computing. Figure 7 represents the task scheduling model of fog computing. The proposed algorithm creates and executes a schedule based on the set of vertices and edges in the task graph and the availability of processors in the processor pool. The executed schedule ensures that the time required for the whole set of tasks gets optimized in such a way that it will consume less time for execution and cost-effective management of cloud computing resources. The three main components of the proposed task scheduling model are task prioritizing, node selection phase, and task reassignment. In the task prioritizing phase, the priority of each task is set. The direction and position of the task in workflow and the information present in data query and application storage decide the priority of the task. In the node selection phase, depending on the priority and processor availability, a task is assigned to a node. If the necessary processing capabilities are available at fog nodes then it is processed at fog level or the task is assigned to a cloud node. The last phase of the proposed algorithm is the task reassignment phase. In this phase quality of services is ensured. This takes input from the second phase and refines it based on the user define deadline constraints.

4.1.2. Task prioritization

In the task prioritizing phase, the priority of each task is set. The direction and position of the task in workflow and the information present in data query and application storage decide the priority of the task. This is scored and placed in the upward rank. This could be based on the distance of the vertices from the output of the application. This also analyzed the computing time at each vertex. The priority of each node in the task graph is calculated by:

$$P(Vi) = \frac{wi}{\sum_{pn\in N} Pn} + max\ [\frac{Cij}{\sum_{pn\in N} bwn} + P(Vj)] \tag{1}$$

in this equation P(v_i) is the priority value of vertices present at the ith location. Wi defines the computation time for that task, C stands for communication delay between node i to node j. For the first node, the value of P(v_j) is 0, and hence distance is 0. However, this function works recursively to compute the priority value of each node, n is the count of nodes. The sorting of all tasks is carried out in non-increased order to maintain the constraint precedence of all the tasks in the application.

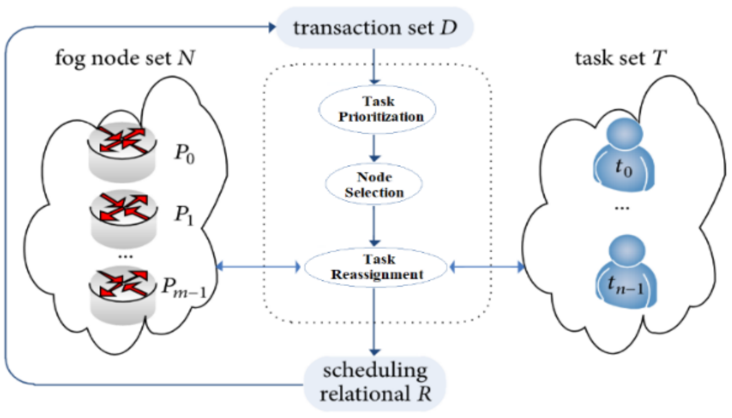


Figure 7. Fog computing layer

4.1.3. Node selection phase

In the node selection phase, depending on the priority and processor availability, a task is assigned to a node. If the necessary processing capabilities are available at fog nodes then it is processed at fog level or the task is assigned to a cloud node. Every specific task in the pool starts executing only once all its previous dependent task completes. The time required for completion of all its previous dependent tasks is defined as data transfer time. Communication time is the time required to transfer data within nodes. Ready time is based on the accessing time required for collecting all the information from the application storage. Along with this early initialization time and early task completion time is calculated. This time is analyzed and the task scheduler is prepared for the assignment of tasks to the processor. The computed time in this phase is considered the final optimized time required for the completion of all tasks in the application pool. The output of this phase is the complete schedule for task assignments. Based on the aforementioned cases, the pseudocode for node selection of the application-task workflow is described below:

```
Pseudocode:
1. Initialize the counter with the output node, calculate the priority value by formula 1
2. By non-incremental order, sorting of all the nodes using priority value in Array L
   Array L = sorted set of nodes
   Array N = array of processor nodes
3. For each task in Array L
4. For each processor in N
5. Compute Data Transfer Time, Ready time, early initialization time, and early task
completion time
6. Compute utility value
7. End Task Loop (Array L)
8. Task Assignment to the processor with max utility value
9. End processor loop (Array N)
Output: scheduler table.
```

5. EXPERIMENTAL ANALYSIS

For evaluating the performance of the proposed algorithm, its time complexity and resource cost are measured. For comparison purposes, various scheduling algorithms discussed in the literature survey were compared with the proposed system. A similar simulation environment and processing capabilities with constant network access are provided for all scheduling algorithms. The analysis result is conveyed in numerical value for comparison. The main aim of this analysis is the comparison of execution time, resource cost, and the trade-off between fog computing and cloud computing. The greedy algorithm which is developed for resource cost optimization in cloud fog environment, HEFT, Cost-Conscious heuristic algorithm is compared with the proposed method. Along with this comparison, the accuracy of our reassignment schema was evaluated by using various user-defined deadline constraints on the application for resource cost optimization. This is to maintain the quality of services of cloud-fog integration. The proposed scheduling algorithm is embedded in the fog broker component of the fog computing layer.

5.1. Simulation environment

This section demonstrates the experiment setup including coding language, software development platform, operating system and computer configuration, processor, and task counts.

Algorithm development tools
- Coding language–Java
- Java SDK version–SDK 11
- Database–MySQL
- Platform–Oracle software
- Cloud integration framework–CloudSlim

System configuration
- Operating system–windows 10–64 bit
- Random Access Memory–16 GB
- Processor–Intel i7 (5th Generation)

Processor configuration at fog network
- Processor Count–15
- Processing rate–250 MIPS

- Bandwidth–1GB

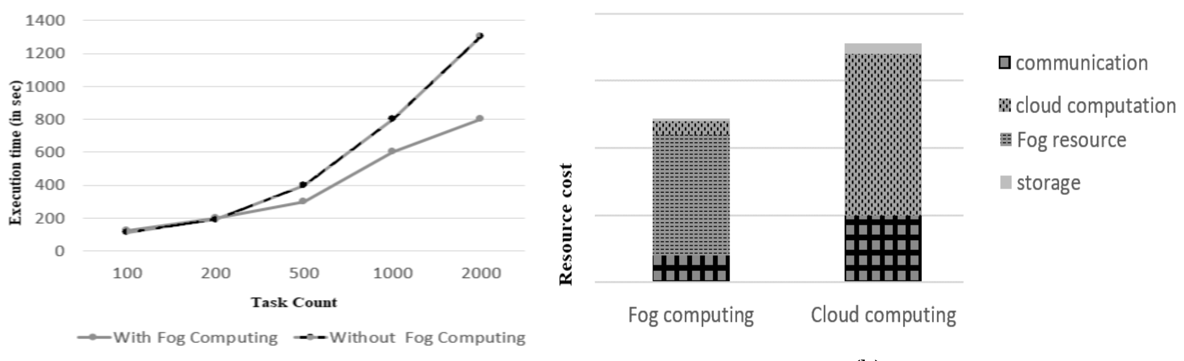
Processor configuration at cloud network
- Processor Count–25
- Processing rate–1000 MIPS
- Bandwidth–500 MB

5.2. Experimental analysis

For evaluating the performance of the proposed algorithm, its time complexity and resource cost are measured. For comparison purposes, various scheduling algorithms discussed in the literature survey were compared with the proposed system. In this research, for performance evaluation, the simulation environment is kept constant for all the algorithms. The processing rate for the fog node is less than that of the cloud node however bandwidth of the fog node is higher than the cloud node, as fog nodes are in close range distance from IoT devices. For analysis purposes, the value of cloud cost is normalized from 0 to 1.

5.3. Execution time and resource cost evaluation

Fog nodes are introduced to minimize the execution time in the IoT and cloud environment. With the application of fog nodes, only high computation calculations require cloud service, proxy computation, and cache memory stored on the fog to reduce the execution time of the task. In this section, the comparison of task completion time, communication time, and resource usage with and without the fog computing layer are illustrated. Figure 8 illustrates the proposed work contribution by representing a comparison of task completion time, communication time, and resource usage at fog clusters and on the cloud side. Figure 8(a), illustrates a task execution time comparison with and without the fog computing layer. It is observed that for a less amount task, the time required for both of the cases is very much similar, however, as the count of the task in the application pool increases, the difference between the execution time of both mechanisms also varies. With the increased amount of tasks, the dependency on cloud services increases. This causes an extra loading period for the computation of precedence-dependent tasks. In the case of the fog computing layer, the amount of execution time gets reduced as compared to the cloud as the number of tasks increases. In this case, fog computing took some extra time for communication but it is much lesser than computation which is present in the cloud computing layer. For resource cost evaluation, a simulation environment is maintained. In the same condition, four application with variable instruction set length is analyzed with and without the fog computing layer. This comparison aims to improve the quality of services of fog computing by enhancing the cost factor. This analysis used the normalized value of cloud cost, this is disused in the cloud usage charges section of this research. Along with cloud charges, extra resource cost at fog nodes is also considered for fair and real-time comparison. Figure 8(b), illustrates a Resource cost for a Workflow-based application with and without a fog computing layer.



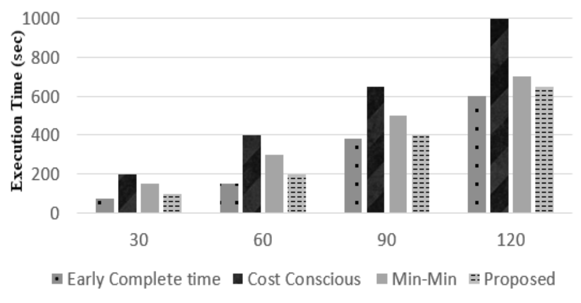
(a) (b)

Figure 8. Comparison of task completion time, communication time, and resource usage, (a) execution time with and without the fog computing and (b) resource cost by using fog and cloud computing

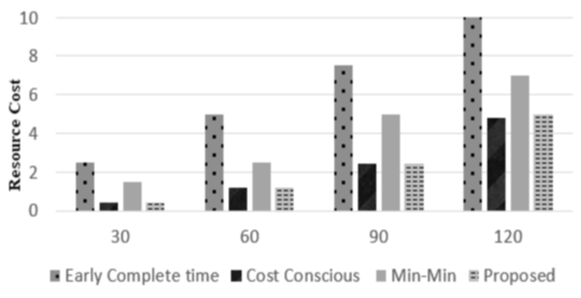
For this comparison communication cost, cloud node processing cost, fog node processing cost, and storage cost are considered. It is observed that in fog computing, storage costs get comprised by using the storage at the fog node, which is present locally. Along with storage, as the fog node also has its computation processing power it can process the small tasks at the fog layer. This reduces the dependency of applications on the cloud. As the services requests to the cloud get reduced, communication overhead towards the cloud layer also reduces which saves the network communication cost. In the case of the fog layer, nodes are available at the local station hence it consumes a very small amount of cost at the fog layer. However, the Cost required for fog node resources in fog computing was high, while in the case of cloud computing there exists no fog. Extra cost requirements for cloud processing, communication, and storage make cloud computing more expensive than fog computing.

5.3. Comparison of the proposed algorithm with the scheduling algorithm

For evaluating the performance of the proposed algorithm, its time complexity and resource cost are measured. For comparison purposes, various scheduling algorithms were compared with the proposed system. For this Heterogeneous early complete-time algorithm, a Cost-Conscious algorithm and min-min algorithm were implemented. Figure 9 illustrates the proposed work contribution by representing a comparison of execution time and cost required for an early complete-time algorithm, algorithm, and min-min algorithm with the proposed algorithm. Figure 9(a) illustrates the comparison of execution time required for the Heterogeneous early complete-time algorithm, Cost-Conscious algorithm, and min-min algorithm with the proposed algorithm. The simulation environment of all of them is kept the same and performance is measured for different applications with a variable number of tasks. The execution time of the heterogeneous early complete-time algorithm, Cost-Conscious algorithm, and the min-min algorithm is compared for applications with task lengths 30, 60, 90, and 120. It was observed that the execution time for the Cost-Conscious algorithm is highest for all four cases. While the execution time for the Heterogeneous early complete-time algorithm was the least. In compassion, it is observed that the execution time for the proposed algorithm and the Heterogeneous early complete-time algorithm was very much similar, however, the proposed system always lags behind the Heterogeneous algorithm by 5 to 10%. Along with the Comparison of execution time, a resource cost comparison is also equally important. Figure 9(b) illustrates the comparison of resource cost required for the Heterogeneous early complete-time algorithm, Cost-Conscious algorithm, and min-min algorithm with the proposed algorithm. The simulation environment of all of them is kept the same and performance is measured for different applications with a variable number of tasks. This analysis used the normalized value of cloud cost, this is disused in the cloud usage charges section of this research. Along with cloud charges, extra resource cost at fog nodes is also considered for fair and real-time comparison. The resource cost of the Heterogeneous early complete-time algorithm, Cost-Conscious algorithm, and the min-min algorithm is compared for applications with task lengths 30, 60, 90, and 120. It was observed that the resource cost for the Heterogeneous early complete algorithm was highest for all four cases. While the resource cost for the Cost-Conscious algorithm was the least. In compassion, it is observed that the resource cost for the proposed algorithm and the Heterogeneous early complete-time algorithm was very much similar, however, the proposed system always lags behind the Heterogeneous algorithm by 1-2%.



(a)

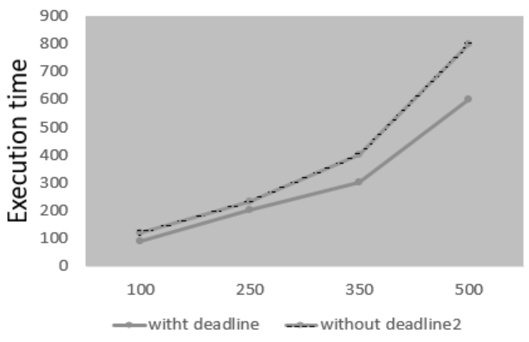


(b)

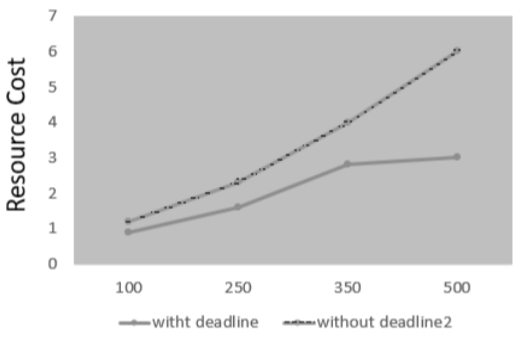
Figure 9. Comparison of execution time and cost required for an early complete-time algorithm, Cost-Conscious algorithm, and min-min algorithm with the proposed algorithm: (a) execution time comparison and (b) resource cost comparison

5.4. Analysis of user-defined deadline constraints

In this section, the performance analysis of the proposed system is compared with the proposed algorithm without considering the user-defined deadline. The user defines some deadline to ensure that the current task or application must finish with the pre-specified deadline. User-defined deadline constraints are introduced to maintain the quality of services. Figure 10 illustrates the proposed work contribution by representing a comparison of the execution time of tasks and cost for the proposed system algorithm with and without considering the user-defined deadline Figure 10(a) illustrates the comparison of the execution time of tasks for the proposed system and proposed algorithm without considering the user-defined deadline. While Figure 10(b) represents the Execution time compassion with deadline constraint, it was observed that the performance of the algorithm with deadline constraint is better. This deadline constraint puts a tight coupling between expected execution time and user-defined time and thereby ensures the quality of service of IoT devices.



(a)



(b)

Figure 10. Comparison of the execution time of tasks and cost for the proposed system algorithm with and without considering the user-defined deadline, (a) Execution time comparison and (b) resource cost comparison

6. CONCLUSION

This research works towards minimizing the cloud cost in scheduling for IoT devices. For this purpose, a cost-effective task and user scheduling algorithm are performed in the fog broker component inside the fog computing layer. Along with the proposed scheduling algorithm, a user-defined deadline constraint-based algorithm was also proposed. For evaluating the performance of the proposed algorithm, its time complexity and resource cost are recorded. For comparison purposes performance of various scheduling algorithms including Heterogeneous early complete-time algorithm, Cost-Conscious algorithm, and min-min algorithm were implemented. It was observed that the time complexity of the Heterogeneous early complete-time algorithm is better than other algorithms, and it is very much near the proposed algorithm, however, the resource cost consumed by the early complete-time algorithm was very high. For the Cost-Conscious algorithm, resource cost dropped heavily and was approximately the same as the proposed algorithm. However, execution time increased drastically. This shows that in terms of execution time and resource cost requirement, the proposed algorithm outperforms other algorithms which we compared in this research. Finally, the re-assignment mechanisms for the proposed system is compared without applying user-defined deadline constraint. It was observed that the performance of the proposed algorithm is better with a task re-assignment mechanism to maintain the quality of services.

REFERENCES

[1] A. S. Abdalkafor, A. A. Jihad, E. T. J. I. J. O. E. E. Allawi, and C. Science, "A cloud computing scheduling and its evolutionary approaches," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 21, no. 1, pp. 489-496, 2021, doi: 10.11591/ijeecs.v21.i1.pp489-496.

[2] S. S. K. N. Prasad, "Quality and energy optimized scheduling technique for executing scientific workload in cloud computing environment," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 21, pp. 1039-1047, 2021.

[3] M. Al-Khafajiy, T. Baker, A. Waraich, O. Alfandi, and A. Hussien, "Enabling high performance fog computing through Fog-2-fog coordination model," in *2019 IEEE/ACS 16th International Conference on Computer Systems and Applications (AICCSA)*, 2019, pp. 1-6, doi: 10.1109/AICCSA47632.2019.9035353.

[4] Y. Y. F. P. M. F. Falah, S. Sukaridhoto, A. W. C. Tirie, M. C. Kriswantoro, B. D. Satria, and S. Usman, "Comparison of cloud computing providers for development of big data and internet of things application," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 22, pp. 1723-1730, 2021, doi: 10.11591/ijeecs.v22.i3.pp1723-1730.

[5] H. Hong, "From cloud computing to fog computing: unleash the power of edge and end devices," in *2017 IEEE International Conference on Cloud Computing Technology and Science (CloudCom)*, 2017, pp. 331-334, doi: 10.1109/CloudCom.2017.53.

[6] S. N. B. Yerzhan, N. Seitkulov, G. B. Ulyukova, B. B. Yergaliyeva, and D. Satybaldina, "Methods for secure cloud processing of big data," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 22, pp. 1650-1658, 2021, doi: 10.11591/ijeecs.v22.i3.pp1650-1658.

[7] N. Bansal and A. K. Singh, "Effective task scheduling algorithm in cloud computing with quality of service alert bees and grey wolf optimization," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 25, pp. 550-560, 2022, doi: 10.11591/ijeecs.v25.i1.pp550-560.

[8] A. Rabay'a, E. Schleicher, and K. Graffi, "Fog computing with P2P: enhancing fog computing bandwidth for IoT scenarios," in *2019 International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*, 2019, pp. 82-89.

[9] J. Garcia, E. Simó, X. Masip-Bruin, E. Marín-Tordera, and S. Sànchez-López, "Do we really need Cloud? Estimating the Fog computing capacities in the City of Barcelona," in *2018 IEEE/ACM International Conference on Utility and Cloud Computing Companion (UCC Companion)*, 2018, pp. 290-295, doi: 10.1109/UCC-Companion.2018.00070.

[10] R. Deng, R. Lu, C. Lai, T. H. Luan, and H. Liang, "Optimal workload allocation in fog-cloud computing toward balanced delay and power consumption," *IEEE Internet of Things Journal*, vol. 3, no. 6, pp. 1171-1181, 2016, doi: 10.1109/JIOT.2016.2565516.

[11] B. Al-Otaibi, N. Al-Nabhan, and Y. Tian, "Privacy-preserving vehicular rogue node detection scheme for fog computing," (in eng), *Sensors (Basel, Switzerland)*, vol. 19, no. 4, p. 965, 2019, doi: 10.3390/s19040965.

[12] H. Al-Zoubi, "Efficient task scheduling for applications on clouds," in *2019 6th IEEE International Conference on Cyber Security and Cloud Computing (CSCloud)/ 2019 5th IEEE International Conference on Edge Computing and Scalable Cloud (EdgeCom)*, 2019, pp. 10-13, doi: 10.1109/CSCloud/EdgeCom.2019.00012.

[13] P. Phuoc Hung and E.-N. Huh, "An adaptive procedure for task scheduling optimization in mobile cloud computing," *Mathematical Problems in Engineering*, vol. 2015, p. 969027, 2015/05/12 2015.

[14] G. Wang, Y. Wang, H. Liu, and H. Guo, "HSIP: A novel task scheduling algorithm for heterogeneous computing," *Scientific Programming*, vol. 2016, p. 3676149, 2016/03/17 2016.

[15] L. Logeswaran, H. M. N. D. Bandara, and H. S. Bhathiya, "Performance, resource, and cost aware resource provisioning in the cloud," in *2016 IEEE 9th International Conference on Cloud Computing (CLOUD)*, 2016, pp. 913-916, doi: 10.1109/CLOUD.2016.0135.

[16] L. Liu, D. Qi, N. Zhou, and Y. Wu, "A task scheduling algorithm based on classification mining in fog computing environment," *Wireless Communications and Mobile Computing*, vol. 2018, p. 2102348, 2018, doi: 10.1155/2018/2102348.

[17] M. Abedi and M. Pourkiani, "Resource allocation in combined fog-cloud scenarios by using artificial intelligence," in *2020 Fifth International Conference on Fog and Mobile Edge Computing (FMEC)*, 2020, pp. 218-222, doi: 10.1109/FMEC49853.2020.9144693.

[18] H. Topcuoglu, S. Hariri, and W. Min-You, "Performance-effective and low-complexity task scheduling for heterogeneous computing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 13, no. 3, pp. 260-274, 2002, doi: 10.1109/71.993206.

[19] Y. Zhao, S. Cao, and L. Yan, "List scheduling algorithm based on pre-scheduling for heterogeneous computing," in *2019 IEEE Intl Conf on Parallel & Distributed Processing with Applications, Big Data & Cloud Computing, Sustainable Computing & Communications, Social Computing & Networking (ISPA/BDCloud/SocialCom/SustainCom)*, pp. 588-595 2019, doi: 10.1109/ISPA-BDCloud-SustainCom-SocialCom48970.2019.00089.

[20] N. Chopra and S. Singh, "Deadline and cost based workflow scheduling in hybrid cloud," in *2013 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, 2013, pp. 840-846, doi: 10.1109/ICACCI.2013.6637285.

[21] V. Marbukh, "Towards fog network utility maximization (FoNUM) for managing fog computing resources," in *2019 IEEE International Conference on Fog Computing (ICFC)*, 2019, pp. 195-200, doi: 10.1109/ICFC.2019.00032.

[22] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the internet of things," in *Proceedings of the first edition of the MCC workshop on Mobile cloud computing*, 2012, doi:10.1145/2342509.2342513.

[23] A. A. Alsaffar, H. P. Pham, C.-S. Hong, E.-N. Huh, and M. Aazam, "An architecture of IoT service delegation and resource allocation based on collaboration between fog and cloud computing," *Mobile Information Systems*, 2016, doi: 10.1155/2016/6123234.

[24] V. B. C. Souza, W. Ramírez, X. Masip-Bruin, E. Marín-Tordera, G. Ren, and G. Tashakor, "Handling service allocation in combined Fog-cloud scenarios," in *2016 IEEE International Conference on Communications (ICC)*, 2016, pp. 1-5, doi:10.1109/ICC.2016.7511465.

[25] D. Zeng, L. Gu, S. Guo, Z. Cheng, and S. Yu, "Joint optimization of task scheduling and image placement in fog computing supported software-defined embedded system," *IEEE Transactions on Computers*, vol. 65, no. 12, pp. 3702-3712, 2016, doi: 10.1109/TC.2016.2536019.

[26] Y. Nan, W. Li, W. Bao, F. C. Delicato, P. F. Pires, and A. Y. Zomaya, "Cost-effective processing for Delay-sensitive applications in Cloud of Things systems," in *2016 IEEE 15th International Symposium on Network Computing and Applications (NCA)*, pp. 162-169, 2016, doi: 10.1109/NCA.2016.7778612.

[27] L. Huang and M. J. Oudshoorn, "Static scheduling of conditional parallel," *Chinese Journal of Advanced Software Research*, vol. 6, no. 2, pp. 121-129, 1999.

[28] W. Chen, G. Xie, R. Li, Y. Bai, C. Fan, and K. Li, "Efficient task scheduling for budget constrained parallel applications on heterogeneous cloud computing systems," *Future Generation Computer Systems*, vol. 74, pp. 1-11, 2017, doi: 10.1016/j.future.2017.03.008.

[29] H. Rafique, M. A. Shah, S. U. Islam, T. Maqsood, S. Khan, and C. Maple, "A novel bio-inspired hybrid algorithm (NBIHA) for efficient resource management in fog computing," *IEEE Access*, vol. 7, 2019, doi: 10.1109/ACCESS.2019.2924958.

BIOGRAPHIES OF AUTHORS

Ali Hussein Shamman is Assistance Lecturer at Computer Techniques Engineering Department, Al-Mustaqbal University College, Babylon Iraq. He holds an MSc degree in Computer Engineering with a specialization in the network computer. His research areas are network, cloud, fog computing, and IoT. Currently doing Ph.D. in computer engineering. He can be contacted at email: alial-safi@mustaqbal-college.edu.iq.

Hussein Ali Alasadi is Assistance Lecturer at Computer Techniques Engineering Department, Al-Mustaqbal University College, Babylon Iraq. He holds an MSc degree in Control Engineering with a specialization in power systems. His research areas are control systems, IoT, and the cloud. system. Currently doing Ph.D. in control engineering system. He can be contacted at email: hussein.ali@mustaqbal-college.edu.iq.

Dr. Hussein Ali Ameen received his first-class B.Eng. degree in Computer Engineering in 2011 from Al-Nahrain University, Baghdad, Iraq. Then, he received his M.Sc. degree in Computer Engineering in 2015 from Al-Nahrain University, Baghdad, Iraq. He holds a Ph.D. degree in Computer Engineering in 2021 with a specialization in Artificial Intleginse systems from UTHM University, Johor, Malaysia. His research interests include RFID, computer networks, wireless communication, Vehicle to Vehicle (V2V) communication systems, and Cloud and IoT systems. He can be contacted at email: hussein_awadh@mustaqbal-college.edu.iq.

Zaid Ibrahim Rasool received his first-class B.Eng. degree in Computer Science in 2010 from Al-Mustansiriya University, Baghdad, Iraq. Then, he received her M.Sc. degree in Information Technology in 2017 from Middle East University, Amman, Jordan. Currently, he is working at Al-Mustaqbal University College, Department of Computer Engineering Techniques, Iraq. His research areas are network, cloud, fog computing, and IoT. He can be contacted at email: zaid.ibrahim@mustaqbal-college.edu.iq.

Hassan Muwafaq Gheni is an Assistance Lecturer at Computer Techniques Engineering Department, Al-Mustaqbal University College, Babylon Iraq. He holds an MSc degree in Electrical Engineering from UTHM university Malaysia. His research areas are network, cloud, fog computing, and IoT. Currently doing Ph.D. in Electrical Engineering. He can be contacted at email: hasan.muwafaq@mustaqbal-college.edu.iq.