A Cache-Based Approach Towards Improved Scheduling in Fog Computing

[1]Osama Amir Khan, [2]Saif U. R. Malik, [1]Faizan M. Baig, [3]Saif Ul Islam, [4]Haris Pervaiz, [5]Hassan Malik, [6]Syed Hassan Ahmed

***Abstract*—** Fog computing is a promising technique to reduce the latency and power consumption issues of the Internet of Thing (IoT) ecosystem by enabling storage and computational resource close to the end-user devices with additional benefits such as improved execution time and processing. However, with an increase in IoT devices, the resource allocation and job scheduling become a complicated and cumbersome task due to limited and heterogeneous resources along with the locality restriction in such computing environment. Therefore, this paper proposes a Cache-based Approach (CBA) for efficient resource allocation in fog computing environment, while maintaining the quality of service. The proposed algorithm is realized using iFogSim simulator and a comprehensive comparison is presented with the traditional First Come First Served and Shortest Job First policies. The performance evaluation revealed that with the proposed scheme the execution time, latency, processing delays and power consumption decreased by 38%, 11.1%, 6% and 17.8%, respectively as compared to the traditional schemes.

Keywords— Cache, Job scheduling, Fog computing, Cloud computing, QoS, IoT, QoE.

1. Introduction

Internet of Things (IoT) envisioned to enable an ecosystem where physical things embedded with sensors and communication technologies can exchange data among each other and with the network to enables services such as smart homes, smart city, smart healthcare, connected vehicles, etc [1, 2]. The basic IoT architecture constitute of four components namely Things, Gateway, Communication Network and Cloud Platform [2]. Things are commonly referred as IoT end devices enabled with sensor or actuator for data collection. The collected data from these devices are then forwarded to Gateways, now a day, named as Fog Nodes. They act as an intermediate node between IoT devices and Cloud Platform providing the needed connectivity, scalability, security and manageability. However, communication network refers to both wired and wireless communication technology needed for the actual transfer of information among these components. Lastly, the Cloud Platform can be regarded as main data collection center constitute of large number of servers and storage capabilities that are linked together. The key functionality of Cloud Platform is to process large amount of data using cloud computing techniques and provide meaningful information that can be used to support IoT application and services.

However, the rapid advancement in IoT ecosystem is expected to have billions of connected devices in future with more stringent application requirements in terms of latency, processing delay, power consumption and execution time, etc. To meet such requirements, new computing technologies such as edge or fog computing are introduced to partially or completely process the data at the device or gateway level [3]. Fog computing inherits the reduced functionalities of cloud computing techniques that can be executed on resource constraint devices such as gateways. Fog computing can be viewed as geographically

[1] Osama A. Khan, Faizan M. Baig is with the Department of Computer Science, COMSATS University, Islamabad, Pakistan. email: *osamaamirkhan@gmail.com*,

[2] Saif U. R. Malik is with Cybernetica AS, Estonia (email: *saif.rehmanmalik@cyber.ee*).

[3] Saif ul Islam is with Dr. A. Q. Khan Institute of Computer Science and Information technology, Rawalpindi, Pakistan. (email: *saif@kicsit.edu.pk*)

[4] Haris Pervaiz is with School of Computing and Communications (SCC), Lancaster University, UK. (emails: *h.b.pervaiz@lancaster.ac.uk*).

[5] Hassan Malik is with Thomas Johann Seebeck Department of Electronics, Tallinn University of Technology, Estonia. (email: *hassan.malik@taltech.ee*).

[6] Syed Hassan Ahmed is with Department of Computer Science, Georgia Southern University, Statesboro, GA 30460, USA (email: *sh.ahmed@ieee.org*).

distributed computing paradigm, having heterogeneous devices at the edge of the network that are connected collaboratively to give elastic computation, communication, and storage services [4]. Fog computing not only reduce the execution time and latency by provide fast data processing but also reduce the burden on communication network and power required by devices for long communication. This will allow the communication networks to serve more devices as well.

Despite having promising features, Fog computing still faces high latency issues, as reported in [5] due to the lack of efficient resource and job scheduling algorithm. There are several factors that makes a resource allocation in a Fog computing environment a challenging task, such as resource scarcity, heterogeneity, geographic restrictions, and varying resource demands [6]. The goals of scheduling and resource allocation is to increase the efficiency of the use of resources, satisfy the Quality of Service (QoS) requirements, meanwhile maximizing the profit of both fog nodes and user devices [7].

Therefore, this paper presents a Cache Based Approach (CBA) for optimal resource and job scheduling in fog computing environment to achieve reduction in execution time, latency, internal processing delay and power consumption. Caching is one of the promising technology for speed up data retrieval time, reduces the number of path lengths, and improves system efficiency. Caching strategies exploit storage capacity to absorb traffic by replicating the most popular content closer to the node and enable low processing cost and remove a single point-of-failure [8-10]. Therefore, the integration of caching in fog computing will enable fog nodes to identify the demand of the user and pro-actively select the most suitable contents to cache in geo-distributed nodes and improve the resource utilization. The main contribution of this paper can be summarized as follows:

- **Firstly**, the resource allocation in fog computing is modelled as delay minimization problem with constraints of available resources and number of jobs.
- **Secondly**, a CBA based scheduling is proposed to address the formulated problem and to improve the efficiency of resource utilization while maintaining the QoS.
- **Thirdly**, a comprehensive comparison of the proposed scheme with the state-of-art schemes such as First Come First Serve (FCFS) and Shortest Job First (SJF) is presented. Moreover, the corresponding performance evaluation in terms of execution time, latency, internal processing delay and power consumption are discussed in detail.

The remaining of the paper is organized as follows. Section 2 presents the related work on cache based approached for fog computing. Section 3 presents the fog computing architecture and its important components along with the interdependencies. The proposed scheduling methodology is presented in Section 4. Section 5 presents the performance evaluation of the proposed scheme and comprehensive comparison other traditional schemes in Section 5. Finally, the conclusion is drawn in Section 6.

2. Related Work

This section presents an overview of resource allocation schemes for resource constraints fog computing devices and highlight the limitation of the proposed schemes.

In the literature, one of the promising solutions for resource allocation for fog computing is by mean of smart gateway [11]. The concept is to use a smart gateway between an end device and cloud, that is capable to pre-process data i.e., filtering, monitoring, management and resource allocation. Based on initial data processing, smart gateway can decide whether data need to be transmitted to cloud or the gateway itself can process the data. However, the resource optimization at smart gateway is still a challenging task. In this regard, in [12], an author presents QoS based resource allocation using Particle Swarm Optimization (PSO). The proposed algorithm improves the performance in terms of resource utilization and reduce delay. Moreover, in [13], a bio-inspired algorithm based on Bees is proposed to optimize the task distribution among resource of end devices, fog nodes and cloud server. The aim of the proposed algorithm is to optimize CPU execution time and memory usage. Similarly, in [14], load balancing algorithm between end device, fog node, and cloud is presented. The proposed algorithm aims to optimize execution time, resource allocation and deadlines based on heuristic approach. A threshold to control the number of jobs request is set for the fog layer. Once the threshold expires, the corresponding tasks forwarded to cloud layer for execution.

The architecture of fog computing to assist systems like 5G networks to attain high performance by ensuring optimal scheduling of job is presented in [15]. In this work, three policies are considered for job scheduling. In the first policy, fog node is randomly selected to execute the job from a uniform distribution called random policy. In second policy, fog nodes provide low latency which depends on systems current state and known as latency policy. The last policy is the capacity policy which selects fog node having most extreme number of outstanding resources among the candidate nodes. The simulation demonstrated that the least latency policy gives a superior outcome because of the accessibility of resources. The authors of the paper inferred that combination of the three policies together locate the most reasonable node for the job. Therefore, utilizing a solitary strategy may not the best answer for the entire system. In [16], authors present a load balancing mechanism in fog computing in which task distribution is depend on graph partition. In this mechanism, tasks are allocated to multiple or single nodes of virtual machines depending on the requirements of the task. By using non-directional graphs, physical nodes of the fog computing are represented in this work. These physical nodes came into a lot of virtual machine nodes as indicated by the accessible fog computing resources, where the virtual machine nodes give administrations to the users by means of graphics partition. To achieve this, the entire graph is used to create a minimum spanning tree; those edges are removed from the tree that did not offer abundant resources. The result of this graph shows the load balancing partition which is fingered by fog computing. Task runtime is achieved using this mechanism. However, the limitation of this technique is for dynamic load balancing is that high performance is not achieved due to the regular repartitioning expected to deal with fog changes. Furthermore, in [17], authors proposed mechanism for task scheduling and resource allocation that is based on container. The method is proposed to decrease the delay in execution of the task. In [18] authors proposed task scheduling algorithm in fog computing that is based on priority levels. The proposed algorithm consists of two steps. First step is assigning the task to nearest fog server. In second step all the requests are process in the three-priority queue within a fog server and reallocate the task to other fog servers if the selected fog server have insufficient resources. Finally, task is sent to cloud if fog layer doesn't have any resources.

Most of the above-mentioned studies focused to increase the performance by minimizing latency and execution time and ignored the resource re-allocation. This paper, combine the concept of smart gateway and caching to address the resource allocation problem for fog computing and presents CBA scheduling. The main idea of the proposed scheme is to incorporate cache module within smart gateway that stores the job and server information. The existence of the cache module will decrease the average waiting time of the jobs, which in return will have significant impact on the overall performance of the system. The presence of the smart gateway increases the performance, minimize latency and propagation delay. Moreover, the caches record will be used to reallocate the job to the cloud and fog nodes. The main objective is to assign the job to the most optimal resource that take minimum time and power to execute the job. This will improve the response time and reduce the cost in term of processing as well.

To the best of author's knowledge, the use of the Caching for job scheduling in fog computing is under studied. In [19], a community-based caching approach (CC) to solve cache pollution and cache monopoly problems in cloud computing based high performance web services. CC performance is compared with thirteen other policies that are managed by cache, and in results its concluded that CC is better than other policies by achieving the cache-hit rate between 0.7-55%. The motivation of this research work is due to the appearance of caching as a cloud service, which supports web services with increasing user demands on its backend database servers. Similarly, in [20], a Cost Aware Cache Replacement Policy (CACRP) for fog computing is proposed. The proposed aims to minimize the cache miss cost in a hybrid memory system.

3. Fog Computing Architecture

Fog computing architecture consists of Cloud layer, Fog layer and the Terminal layer as shown in Fig. 1. Cloud layer includes multiple data storing devices and high-performance servers, and issue different application services like smart transportation, smart home, smart factory, etc. Cloud layer has huge storage and powerful computing capabilities to support extensive computing analysis and permanent storage of a huge amount of data. However, unlike traditional cloud computing architecture, not all computing and storage tasks go through the cloud. Depending on the demand load, the central modules of the cloud are managed and planned efficiently some control strategies to improve the use of cloud resources [21].

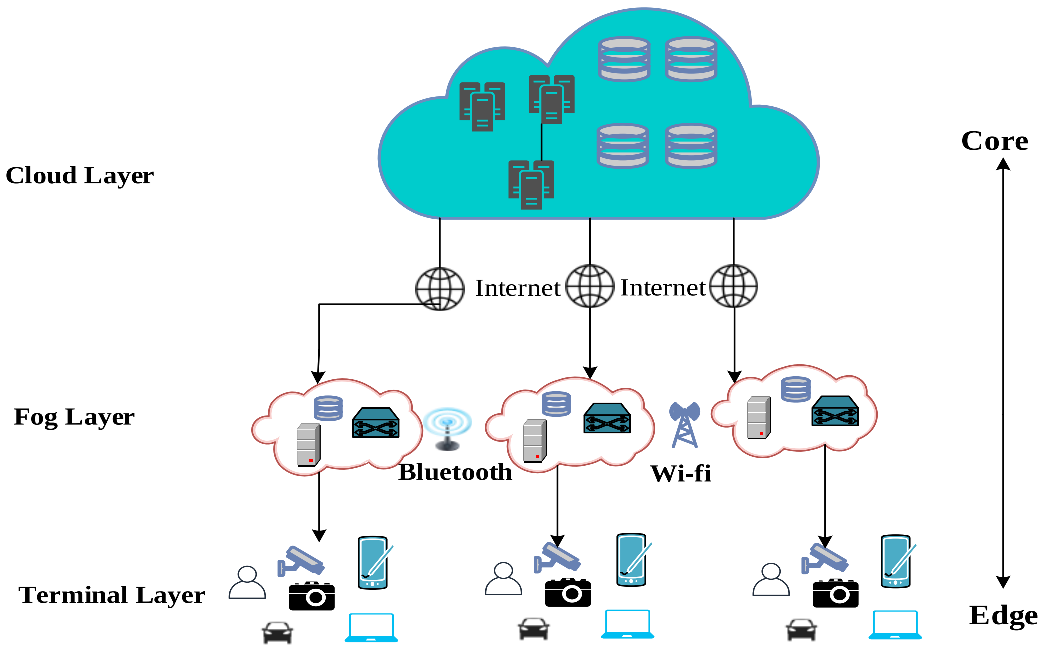

Figure 1: Fog Computing Architecture

Fog layer has numerous fog nodes, and each node includes a base station, routers, switches, gateway, access point, specific fog servers, etc. Fog node is a bridge between the cloud and end devices. Fog nodes can be static or mobile on a moving carrier. The end devices get services from fog nodes. Real time analysis and low latency can be achieved in fog layer. Also fog layer is connected by IP core network with cloud data center. Fog nodes are connected with cloud to get more storage and computing capabilities. The third layer is a terminal layer and is closer to the end user and the physical environment. It includes different IoT gadgets like smart vehicles, mobile phones, sensors, smart card and so on. These devices (sensors) are responsible for sensing the data of physical objects or events and transmit it to upper layer for processing and storage [21].

The end user can directly communicate with fog server using wireless connections that are 4G LTE devices, Wi-Fi, LPWAN technologies and Bluetooth, etc. Cloud and fog server communicate to each other through a wired and wireless connection to access more application tools and computing services or resources. Mostly data stored in fog node is on brief premise. The cloud is more suitable for long-term data storage because of the availability of more resources than fog nodes. When the data is sent to the cloud, it is then not required to store on fog nodes [22].

The issue of latency between end user device and the cloud is addressed by fog computing architecture. Fog computing extends the cloud computing by moving the storage and computation resources at the edge of the network. Fog computing offers advantages like a fast response to delay-delicate applications, data aggregation for heterogeneous devices, gives data security and protection for sensitive data, avoids pointless communication by filtering the data before sending it to the cloud and provide context-aware and location-aware services. Along with these benefits, there are several challenges that still need to be addressed. These challenges include fog-cloud collaboration, service scalability (horizontal and vertical), fog scalability, fog resource management and fog based dedicated applications [22].

There are different performance metrics on which fog computing performance is measured. In [22], researchers evaluated fog computing performance against different metrics of performance such as processing costs, processing delay, and processing power to show the gain in performance. Quality of experience (QoE) is another performance metrics to evaluate the performance of scheduling algorithm for fog computing [14].

This paper proposes an extension of the fog computing architecture by introducing smart gateway enabled with caching for efficient resource allocation and evaluate the performance in terms of execution time, processing delay, latency and power consumption. The detailed proposed architecture is presented in Fig. 2 along with the detailed methodology in Section 4.

4. Proposed Methodology

In this section, the system model for CBA based resource allocation for fog computing is presented as shown in Fig. 2. In fog computing, job scheduling problem focus at assigning sets of jobs to fog nodes located at the edge of the network in a way to minimize CPU execution time and latency. Table 2 shows the list of symbols and notations that are used throughout the paper.

Table 2: List of Notations

T_{list}	List of job requested by all users	L	Length of job/tuple
T_i	Job requested by single user	R	Link bandwidth (bits/seconds)
C_{type}	Communication Type	D_c	Cloud Data center
F_{ls}	Local storage of fog server	I_{pt}	Internal processing time
T_{exec}	Execution time	S_{type}	Service type requested by IoT
T_{type}	Type of job	$T_{arrival}$	Task arrival
T_{ct}	Computational time	$P_{utility}$	Power utility
T_{pd}	Propagation delay	T_{Time}	Tuple time
Fg_{list}	Lists of fog servers	$v_{t,data}$	Volume of data
F_i	Single fog server	d	Length of physical link
F_{broker}	Broker	s	Propagation speed
T_{pt}	Total number of tuples	Ɛ	End to end delay
F_c	Fog Cache	Q_t	Queuing time
$F_{i,c}$	Computational time of selected fog server	T_d	Transmission delay

Let T_{list} denoted as a set of jobs requested by the IoT devices. The job $T_i \in T_{list}$ is a job requested by the single user and have parameters such as job type (T_{type}), computational time (T_{ct}), and propagation delay (T_{pd}). The type of the job can be small textual, bulk data, location-based, large multimedia and medical data. Depending on the T_{type} the T_{ct} of the job is computed, which is the processing time of the

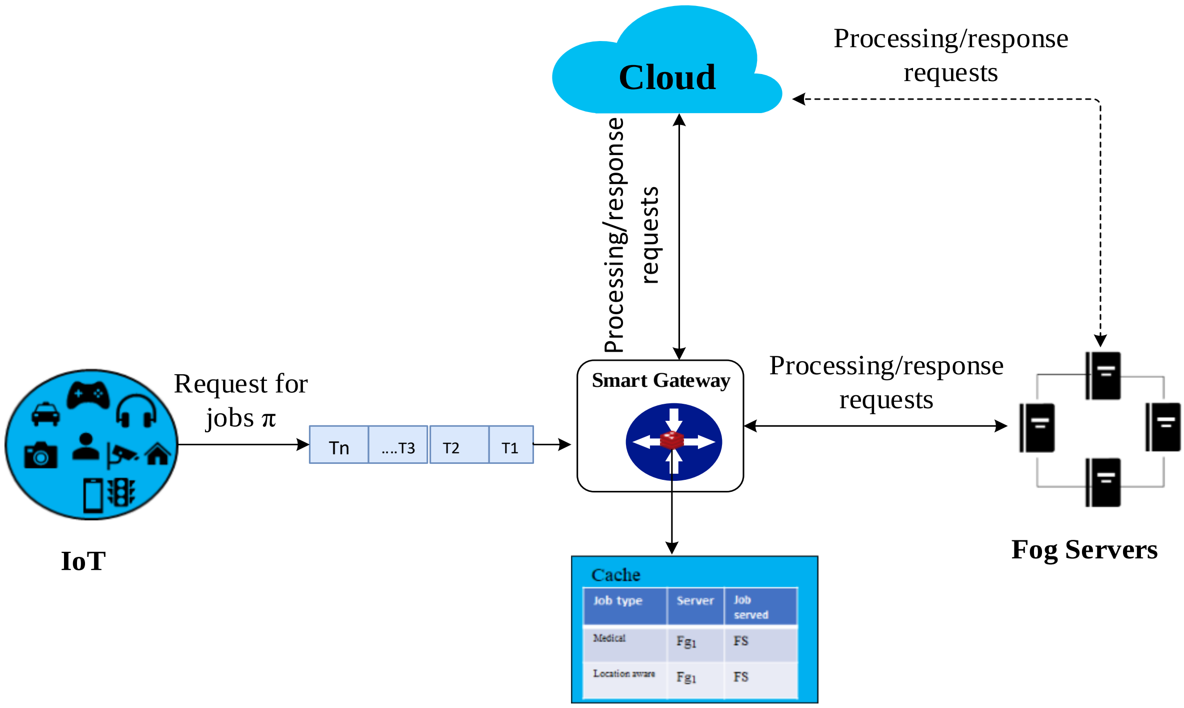

Figure 2: Cache Based Approach

job. The T_{ct} may vary based on the T_{type} of the job. The T_{pd} is the transmission delay from source to destination. As the processing capabilities of the fog are limited, the T_{type} identify if the job is executed on the fog node or on the cloud, represented as C_{type} and F_{ls} respectively. If the job is large, then the job is executed on the cloud, otherwise it is executed on an optimal fog node.

The services that are requested by the IoT devices, represented as S_{type} can be categorized as computational, storage, or a combination of both. Fog broker (F_{broker}) serves an an entity that acts as a service provider between the IoT and the fog [23]. We modeled (F_{broker}) as a global gateway that will optimally dispatch the requests between the fog and the cloud. It stores all information about fog servers and also has the information about the current job that is going to be executed. The Fg_{list} represents the list of fog nodes and F_i is an individual fog node, where $F_i \in Fg_{list}$ and $i = \{1, \ldots, n\}$ (n is the total number of nodes). We have adopted first in first out (FIFO) methodology to move the jobs from the queue to the cache (F_c). Once the job requests are in the queue, the information from the F_c is used to dispatch them to the respective fog nodes. A gueue is maintained to store the jobs, when number of jobs arrive simultaneously. Using FIFO for the jobs in queue, we select nearest fog server and cache the job type, fog server, arrival time, time to leave, and internal processing time.

Once the jobs are dispatched and received at the broker, it performs an initial check to identify if the sleceted fog server has the required power and resources to exeute the job. If the fog nodes has the resources, then its is executed on it. Otheriwise, a new node is selected. If no node is available, then the job is moved to the cloud.

The proposed algorithm implements the cache on the smart gateway. Smart gateway helps in connecting IoT devices to the network and efficient utilization of the cloud by setting up the time and type of data to be transferred over the network [24, 25]. The objective of the algorithm is to perform different tasks like preprocessing, filtration of data and reconstruct it in more useful ways, moving data to cloud to provide QoS, keeping a check on delay, execution time, energy consumption. Overall the cache is used in algorithm to expediate the scheduling process and to optimally select the fog nodes.

4.1. Cache-based Job Scheduling Mechanism

This section presents the proposed CBA scheduling that mainly focuses on designing a centralized mechanism for content delivery by introducing cache-based processing. The devised algorithm with all the functionalities is shown in Algorithm 1. Initially, when a user request for a T_i, the request will go to cache inside the smart gateway followed by the queue (line 1-6 in Algo 1) as shown in the flow diagram in Fig. 3. After receiving a job request, the proposed algorithm will execute in two steps as follow:

4.1.1 Execution of Job on Fog Nodes

When job request T_i arrived at the queue, the algorithm first search for the Fog server F_i from the list of available fog servers F_{list} that matches with the requirements of the T_i. If it finds the optimal server for the job, it will assign the job to that F_i from F_{list} (line 7-14 in Algo 1). Next, the type of job and address of the F_i to which the job is assigned is stored in the cache, to save time when a similar type of T_i is requested again (line 17-41 in Algo 1). Once done, the algorithm takes the next T_i from T_{list}, which is first checked within F_c whether this type of T_i is executed or not. If yes, then it will send the request to the specified F_i. If the F_i is already fully utilized, then it will make a new entry in F_c and repeat the same procedure (line 42-50 in Algo 1). After the execution of job on fog servers the CBA will respond the type of the job and server address to cache to save time in future execution.

4.1.2 Execution of Jobs on Cloud

If a new job request arrive and all the F_i are fully utilized, then this request is sent to the cloud data center D_c for processing. Similarly, it also depends on the size of the job, the data or compute intensive jobs are sent to the cloud instead of serving at fog servers as D_c have more processing and storage resources available (line 58-68 in Algo 1). After the processing of job CBA will respond backed to cache in order to update the cache information for the future use. The flow diagram of proposed approach is given in Fig 3.

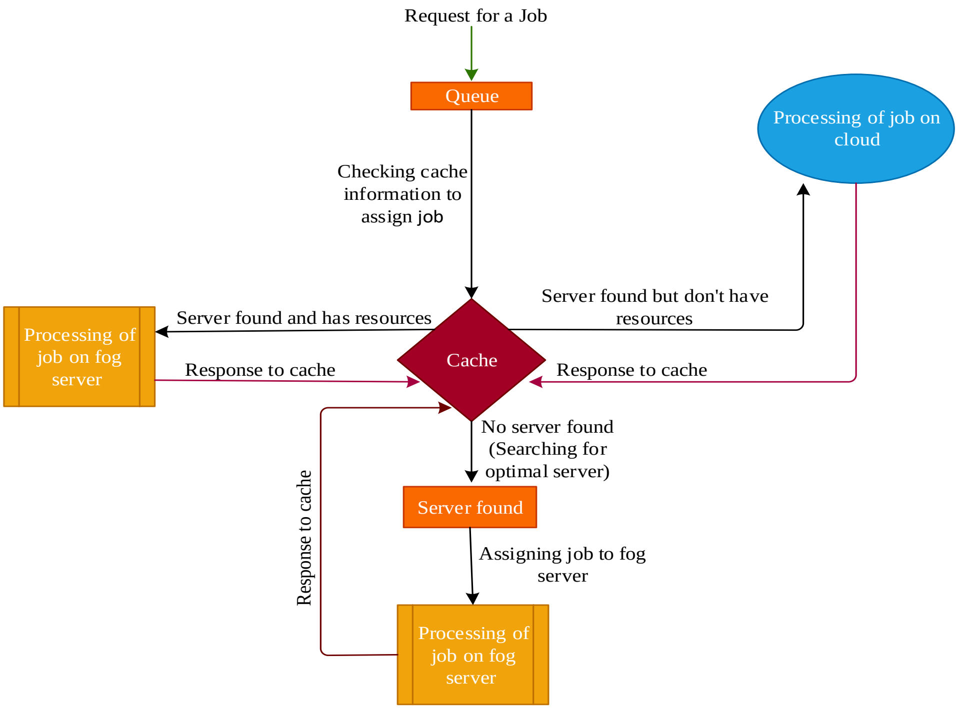

Figure 3: Flow Diagram of CBA Mechanism

Algorithm 1: CBA scheduling algorithm

Inputs are:

T_{list}, Fg_{list}, F_c, $Ct = \{true, false\}$, $St = \{S1, S2, S3\}$

for each: ($T_i \in T_{list}$) **do**
create T_{Times} class object
Set job arrival time
Add T_{Time} to T_{list}
Checking server
compatibility for a job
if $(\boldsymbol{F_i Haspower}(\,)\&\&\ \boldsymbol{Hasresources}\,(\,))$
{
then
create T_{Time} class object
set job arrival time
creating list of job arrival time
Item is not served to cloud
} end if
end for
end procedure
Procedure FOGCACHE (T_{pt}, Fg_{list})
if $F_c.D_{Type}$ == $F_{broker}.D_{Type}$).count > 0)
{
$F_{broker}.fg_i$ = $haspower()\ \&\&\ hasresources$
F_c.Where(x => x.D_{Type} ==

$F_{Broker}.T_i.D_{Type}$).OrderBy(x=> x.I_{pt}).OrderBy(x=>x.$L_i.P_t$).get();
}
Else
{
$F_{broker}.F_i$= $F_{broker}.haspower()$ && $hasresources$
}
if ($F_{broker}.F_i$ is Not null)
{create F_{Times} as fogTime
set job arrival time
F_{Name} = Fg_i
T_{Name} = T_i
List of jobs having same sources &
destination
T_i.IsServed = true;
T_i.IsServerFound = true;
Setting endtime in milliseconds
create T_{Time} class object => T_{Time}
set departure time
} Name = T_i
end if
if (FogSimulator.IsCreateCache) **then**
{ Add elements in list of F_c
Set D_{Type} = $T_i.D_{Type}$
FogServer = $F_{broker}.Fg_i$.ID
I_{pt} = $T_i.I_{pt}$
TupleGuid = T_i.ID
l_i = L_i
}
end if
%Setting Fog consumption time
Set F_{Time}.Consumption = $P_{Utility}$.Consumption($F_{Broker}.F_i$, ($ttime_{ms}$ -($InitTime_{ms}$)), $ttime_{ms}$, T_i)
$F_{Time}.Free_{Time}$ = T_{Time}. T_i Departure
$F_{broker}.Fg_i$.ReleasePower($F_{broker}.F_i$, T_i)
Else
{ Log ('missed by fog');
}
if C_t == 1) **then**
if (ServedByCloud (tuple, false, S_i, D_c)) **then**
{
Set T_i.IsReversed = true

```
Ti.IsCloudServed = true
Ti.IsServedByFC_Cloud = true
}
end if
else
{           Set Ti.IsCloudServed = false
Ti.IsReversed = true
}
end if
        Tuple = Ti
end FOGCACHE FUNCTION
```

5. Performance Evaluation

The proposed scheme is evaluated in a C# based Fog computing simulation environment. The simulation setup provides the necessary networking infrastructure, IoT and Fog nodes, and cloud data centers. The simulation platform is motivated by CloudSim [27]and iFogSim [28] simulators. It provides all the primary and advanced features to simulate IoT-based Fog computing environment. We have considered the different types of jobs generated from dumb objects [29], nodes, sensors, mobile and actuators [30] of having datatype small textual, bulk, location-based, large multimedia and medical data. In the experiments, we have considered 30,000 jobs (more details about the jobs can be obtained from [31]). There are nine heterogeneous fog servers that serve all the jobs. Fog servers and IoT devices are geographically distributed. IoT devices and fog servers are randomly deployed. The distance between fog servers is in Kilometers. Each IoT device is associated with its nearest fog server which is further linked with the cloud datacenters that contain the machines with high computational capacity and power. The proposed policy is compared with the FCFS and SJF scheduling policies. We used the following metrics to evaluate the performance of our proposed approach.

- Propagation delay/Latency
- Execution time
- Processing delay
- Power consumption

Based on the above metrics, the results of the CBA are compared with FCFS and SJF algorithms. In this section, we evaluated our Cache-based technique against FCFS and SJF algorithm to show the effectiveness of CBA for job scheduling in fog computing.The execution of T_i on F_i depends on the T_{type}, T_{ct}, and T_{pd}. So, the execution time, denoted by T_{exec}, is calculated for fog servers as:

$$T_{exec} = T_{type} + T_{ct} + T_{pd} \tag{1}$$

Eq. (1) mentioned here is commonly used in the literature such as [32, 33]. The computational time of job T_i in fog server, represented as $\mathrm{F}_{i,c}$ is calculated as:

$$F_{i,c} = \frac{v_{t,data}}{\sum_{p=1}^{P_{max}} r_{t,s}^{j}}, \tag{2}$$

where $r_{t,s}^{j}$ represents resources that are allocated by fog node for T_i during the period p, where P_{max} is the maximum time that a job can maintain, and $v_{t,data}$ represents the volume of data that need to be processed. Furthermore, the end-to-end delay is given as:

$$\mathcal{E} = Q_t + T_d + T_{pd} \qquad (3)$$

where Ɛ represents end-to-end delay, which is computed as summation of Queuing time (Q_t) i.e. time taken by job to wait in the queue until it can be executed, transmission delay (T_d), and propagational delay (T_{pd}). The T_{pd} can be calculated in the similar manner as [34]:

$$T_{pd} = \frac{d}{s} \qquad (4)$$

where *d* is the length of physical link and *s* is the propagation speed in the medium. Transmission delay can be defined as how long it takes to get all the packets into the wire in the first place. Transmission delay is calculated as follow:

$$T_d = \frac{L}{R} \qquad (5)$$

Where, L is the length of the job/tuple and R is the link bandwidth in bits per second.***Execution time*** means the time required to complete simulation, which means the scheduling of all the jobs. Three different algorithms are run on the same data set and on the same machine to evaluate the results. In Fig. 4, it is shown that our proposed model takes less time in simulation in comparison to two other state-of-the-art algorithms. Cache takes 4482.928 seconds to complete the simulation as compared with FCFS and SJF which took 6843.772 seconds and 7280 seconds, respectively. In Fig. 4, the reason for less execution time is that cache information is being used to assign the job to fog server. Jobs do not have to wait for long in the queue if all servers are busy then it starts sending jobs to the cloud for execution.

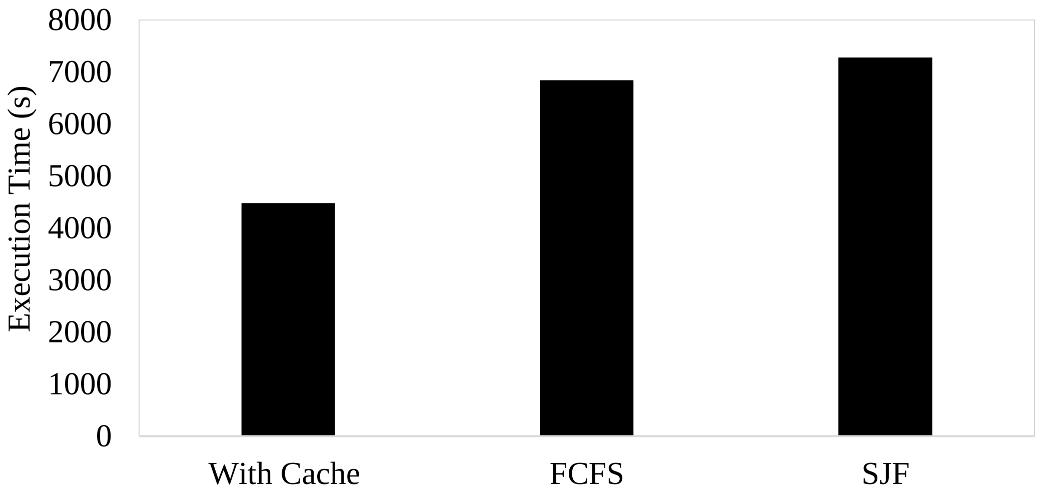

Figure 4: Execution Time

Propagation delay is the time taken by a job from the source to reach the destination. We calculated the average propagation delay of cache along with FCFS and SJF algorithm as shown in Fig. 5. Time taken by job request is calculated to reach from source to the destination server. Average propagation delay taken by our proposed model is less than the other two algorithms as shown in Fig. 5. The average propagation delay took by each job to reach to fog server while using cache approach is 380.853 milliseconds. FCFS takes 428.594 and SJF takes 381.503 milliseconds, respectively. In fact, the affective distribution of jobs to the available resources results in better bandwidth utilization that ultimately

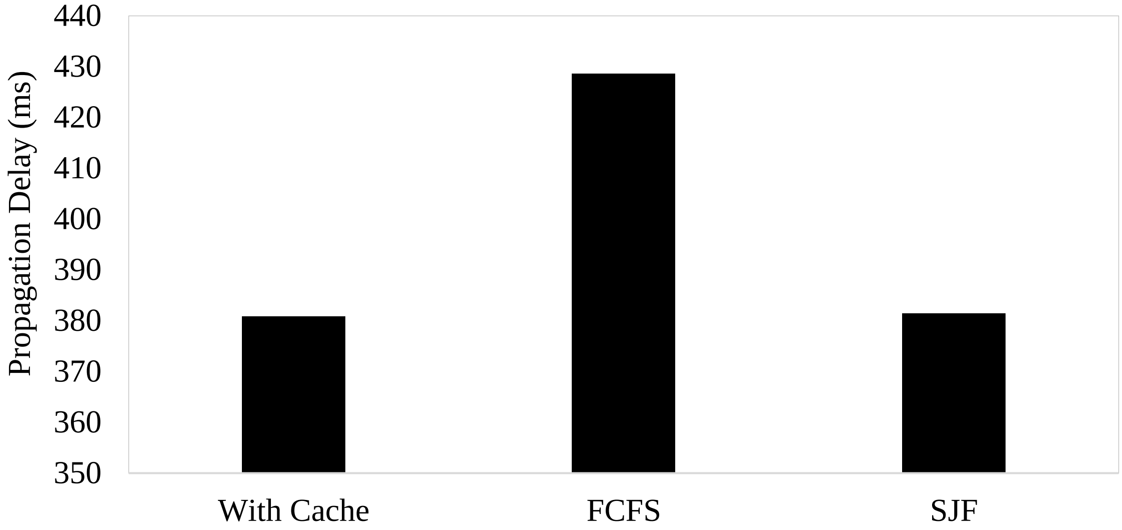

Figure 5: Propagation Delay

reduces the congestion on the network and results in lower propagation delay. Similarly, the purpose of our proposed algorithm is to utilize the fog resources as maximum as possible that also has an impact on the propagation delay – the jobs sent to cloud results in higher delays.

In our evaluation test, the delay is measured in milliseconds, and it is the time taken by a processing element to execute the job. Fig. 6 presents the average internal processing delay of the servers. The cache takes less time when compared with FCFS and SJF. Cache takes 92.567 ms whereas FCFS takes 112.648 and SJF takes 157.106 ms respectively.

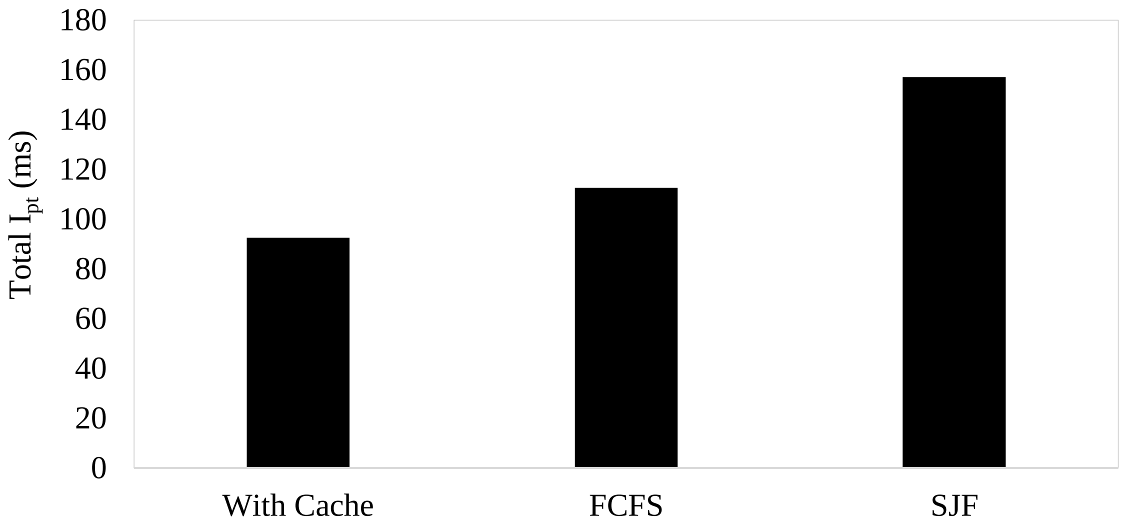

Figure 6: Total Average Internal Processing Delay

The amount of processing performed by the servers is directly proportional to the power consumed by the servers. In Fig. 7, we have depicted the power consumption of fog servers in Watts (W). The x-axis presents the servers ID and y-axis exhibits their corresponding power consumption. Fig. 7 shows the trend of power consumed in the fog resources. The power consumption has higher values throughout the time because of its two types – static and dynamic power consumption. Static power consumption refers to the power required for the working of electronic peripherals of fog servers when it is turned on and there is no load on it. Afterward, the power consumption is proportional to its utilization – called dynamic power consumption. The utilization of the fog servers depends on the millions of instructions per second (MIPS) required by the jobs at fog server for its computation and also the frequency of jobs has an impact on the utilization of fog resources. Here, we refer the fog servers to fog resources. The utilization of the fog servers depends on the scheduling policy. Fig. 7 shows that fog servers are utilized arbitrarily. It can be seen that some servers have more consumption that represents that they capture more traffic than the others. When fog resources are saturated, the incoming requests are forwarded to the cloud. Similarly, it also depends on the job size, the big jobs are sent to the cloud instead of serving at fog servers.

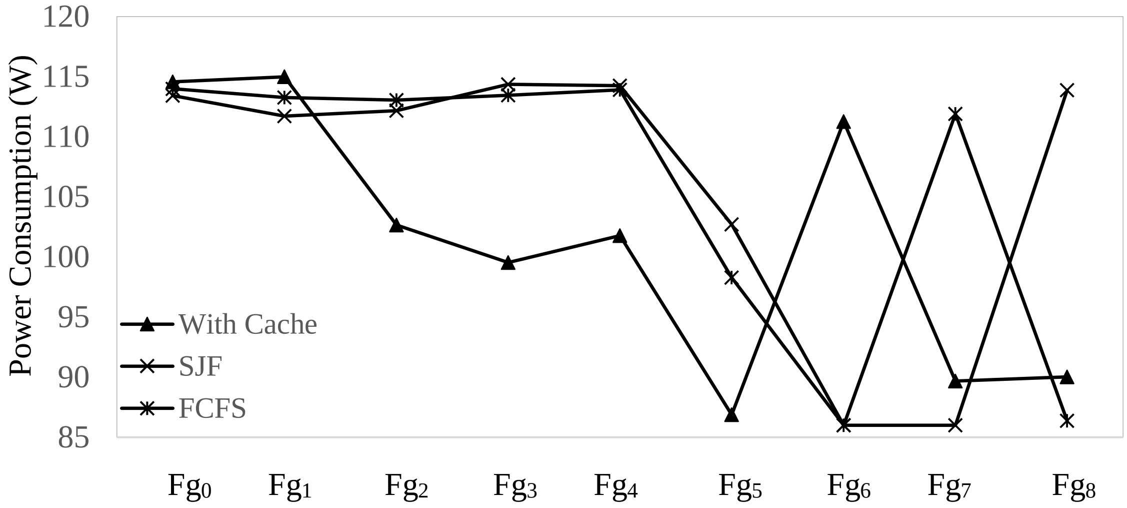

Figure 7: Server Power Consumption

Tuple execution time is the average time taken by a tuple to execute. In Fig. 8 it is shown that how tuples are executed in each second. As in Fig. 8 in the first second, only one job is executed, and in

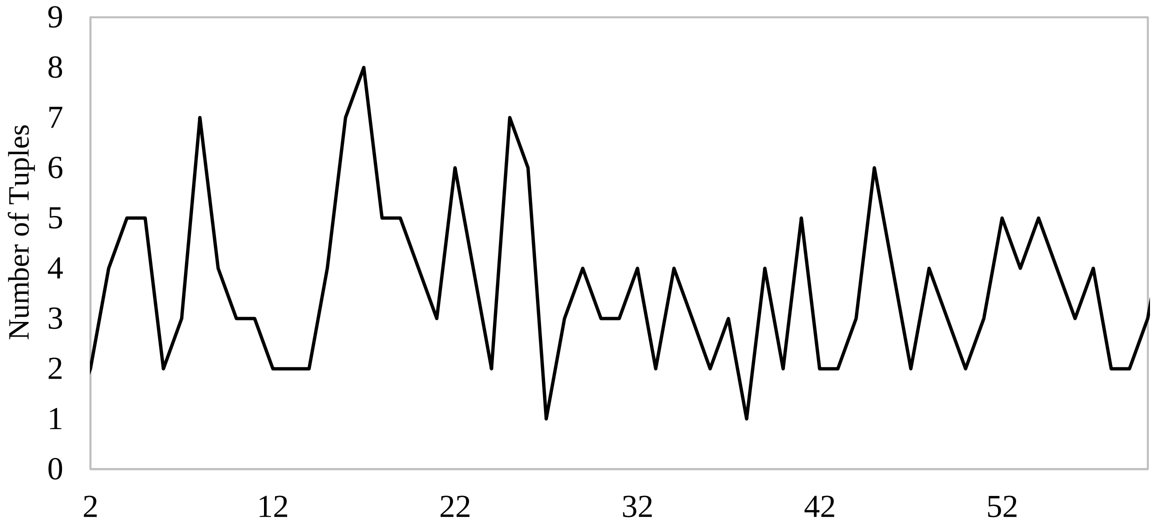

Figure 8: Tuple Execution

seventeen seconds 8 jobs are executed. Each tuple has an associated CPU and network cost to process it [32].

Fig. 9 shows the average end to end delay experienced by tuple to execute on fog and cloud using the CBA mechanism. It can be noticed that the average end-to-end tuple delay falls below as data is processed near to source and cache information is used due to which tuple does not have to wait for long in the queue. The reasons for peak values in the graph is due to the execution of jobs on the cloud and, the distance between the job source and the fog server. In Fig. 10 the time taken by each tuple to travel from sender to destination is depicted.

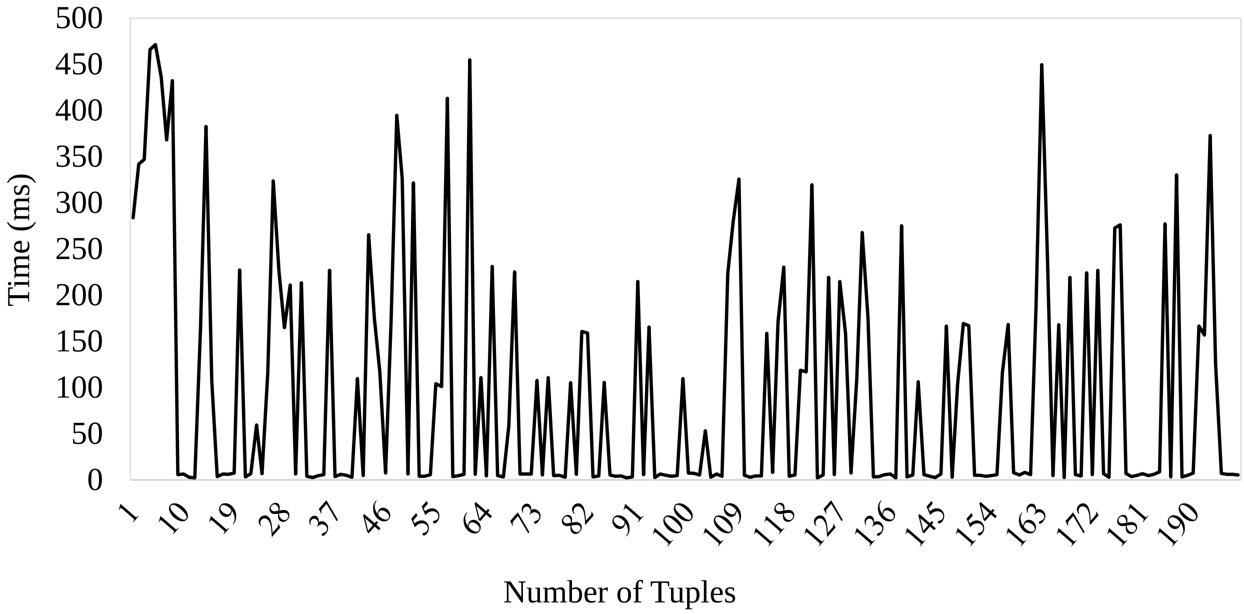

Figure 9: End to End Delay of Tuples

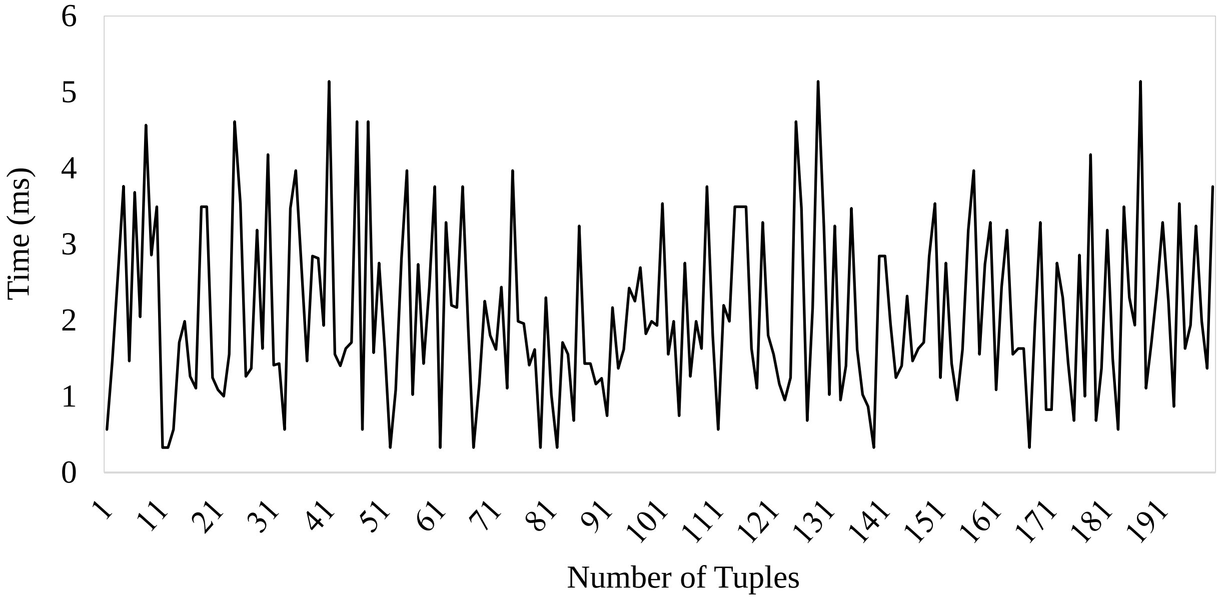

Figure 10: Network Propagation Time

Based on the above results the overall performance increased. One disadvantage of this method is that initially, server utilization is maximum which can be a future work to minimize it. The overall performance increased, latency and propagation delay decreased, and QoS achieved.

6. Conclusion and Future Work

Since the inception of IoT, Edge and Fog computing paradigm, have undergone an enormous evolution in a way that they can be used. More and more new applications, such as face recognition, augmented reality, online interactive gaming, and natural language processing are emerging and attracting the researcher to explore methods to enable computing near device level. However, such applications are generally data intensive or compute intensive, which demands high resource and energy consumption. Therefore, enable sophisticated computing algorithms at fog or edge node which are resource constraint devices is a challenging task. In this paper, the job scheduling problem in the fog computing environment for the efficient execution of tasks requested by end user devices is explored. The proposed algorithm integrate cache in the smart gateway and proposed a scheduling scheme that decrease the execution time, propagation delay and internal processing time of the jobs being requested. To handle job scheduling the proposed algorithm uses FCFS policy for a queue. The performance of the proposed algorithm is compare with the traditional FCFS and SJF policies, and the results showed that our approach is more optimized and yield reduction of execution time, latency, processing delays and power consumption by 38%, 11.1%, 6% and 17.8%, respectively as compared to the FCFS and SJF policies. In the future, the aim is to extend the proposed algorithm to minimize the server execution as it is very high at the start of the algorithm and to explore SJF policy in a CBA based resource allocation scheme for fog computing.

REFERENCES

[1]. H. Malik, N. Kandler, M. M. Alam, I. Annus, Y. Le Moullec and A. Kuusik, "Evaluation of low power wide area network technologies for smart urban drainage systems," 2018 IEEE International Conference on Environmental Engineering (EE), Milan, 2018, pp. 1-5.

[2]. M. M. Alam, H. Malik, M. I. Khan, T. Pardy, A. Kuusik and Y. Le Moullec, "A Survey on the Roles of Communication Technologies in IoT-Based Personalized Healthcare Applications," in IEEE Access, vol. 6, pp. 36611-36631, 2018.

[3]. A. Saleem et al., "FESDA: Fog-Enabled Secure Data Aggregation in Smart Grid IoT Network," in IEEE Internet of Things Journal, 2019.

[4]. S. K. Datta, C. Bonnet and J. Haerri, "Fog Computing architecture to enable consumer centric Internet of Things services," 2015 International Symposium on Consumer Electronics (ISCE), Madrid, 2015, pp. 1-2.

[5]. Luan, T.H., Gao, L., Li, Z., Xiang, Y., Wei, G., Sun, L., 2015. Fog computing: focusing on mobile users at the edge. Comput. Sci., 1–11.

[6]. S. Yi, C. Li, and Q. Li, “A survey of fog computing: Concepts, applications and issues,” in Proc. 2015 Workshop on Mobile Big Data. (MBD’15), Jun. 2015, pp. 37–4.

[7]. S. U. R. Malik, S. U. Khan, S. J. Ewen, N. Tziritas, J. Kolodziej, A. Y. Zomaya, S. A. Madani, N. Min-Allah, L. Wang, C.-Z. Xu, Q. M. Malluhi, J. E. Pecero, P. Balaji, A. Vishnu, R. Ranjan, S. Zeadally, and H. Li, "Performance Analysis of Data Intensive Cloud Systems Based On Data Management and Replication: A Survey," Distributed and Parallel Databases, vol. 34, no. 2, pp. 179-215, 2016.

[8]. U.Idachaba and F.Wang, "A Community-Based Cloud Computing Caching Service," 2015 IEEE International Congress on Big Data, NewYork,NY,2015,pp.559-566.

[9]. A. Ruia, C. J. Casey, S. Saha and A. Sprintson, "Flowcache: A cache-based approach for improving SDN scalability," 2016 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), San Francisco, CA, 2016, pp. 610-615.

[10]. B. Assila, A. Kobbane and M. El Koutbi, "A Many-To-One Matching Game Approach to Achieve Low-Latency Exploiting Fogs and Caching," 2018 9th IFIP International Conference on New Technologies, Mobility and Security (NTMS), Paris, 2018, pp. 1-2.

[11]. M. Aazam and E.-N. Huh, ‘‘Fog computing and smart gateway based communication for cloud of things,’’ in Proc. Int. Conf. Future Internet Things Cloud (FiCloud), 2014, pp. 464–470.

[12]. Krishnapriya, S., Joby, P. P., 'QoS Aware Resource Scheduling in Internet of Things-Cloud Environment', International Journal of Scientific & Engineering Research, vol. 6(4), 2015.

[13]. S. Bitam, S. Zeadally, & A.Mellouk, ”Fog Computing Job Scheduling Optimization based on Bees Swarm,” by Enterprise Information Systems, vol. 12, issue 4, p. 1-25, April 2017.

[14]. M. Verma, N. Bhardwaj, and A. Yadav, “Real Time Efficient Scheduling Algorithm for Load Balancing in Fog Computing Environment”, in MCEP, 2016, doi: 10.5815/ijitcs.2016.04.01.

[15]. K. Intharawijitr, K. Iida and H. Koga, "Analysis of fog model considering computing and communication latency in 5G cellular networks," 2016 IEEE International Conference on Pervasive Computing and Communication Workshops (PerCom Workshops), Sydney, NSW, 2016, pp. 1-4.

[16]. S. Ningning, G. Chao, A. Xingshuo and Z. Qiang, "Fog computing dynamic load balancing mechanism based on graph repartitioning," in China Communications, vol. 13, no. 3, pp. 156-164, March 2016.

[17]. L. Yin, J. Luo and H. Luo, "Tasks Scheduling and Resource Allocation in Fog Computing Based on Containers for Smart Manufacturing," in IEEE Transactions on Industrial Informatics, vol. 14, no. 10, pp. 4712-4721, Oct. 2018.

[18]. T. Choudhari , M. Moh , T.S. Moh, “Prioritized task scheduling in fog computing”, Proceedings of the ACMSE 2018 Conference, pp.1-8, March 29-31, 2018, Richmond, Kentucky.

[19]. U.Idachaba and F.Wang, "A Community-Based Cloud Computing Caching Service," 2015 IEEE International Congress on Big Data, NewYork,NY,2015,pp.559-566.

[20]. A.Jia, G.Han, H.Wang and F.Wang “Cost-aware cache replacement policy in shared last-level cache for hybrid memory based fog computing” Enterprise Information Systems, vol. 12, issue 4, p. 435-451, April 2018.

[21]. M. Aazam, S. Zeadally and K. A. Harras, "Fog Computing Architecture, Evaluation, and Future Research Directions," in IEEE Communications Magazine, vol. 56, no. 5, pp. 46-52, May 2018.

[22]. P. Hu, S.Dhelim, H.Ning, T.Qi “Survey on fog computing: architecture, key technologies, applications and open issues” Journal of Network and Computer Applications, vol. 98, pp. 27–42, Nov. 2017.

[23]. J. Gedeon, C. Meurisch, D. Bhat, M. Stein, L. Wang and M. Mühlhäuser, "Router-Based Brokering for Surrogate Discovery in Edge Computing," 2017 IEEE 37th International Conference on Distributed Computing Systems Workshops (ICDCSW), Atlanta, GA, 2017, pp. 145-150.

[24]. Dastjerdi, Amir Vahid, Harshit Gupta, Rodrigo N. Calheiros, Soumya K. Ghosh, and Rajkumar Buyya. "Fog computing: Principles, architectures, and applications." In Internet of Things, pp. 61-75. Morgan Kaufmann, 2016.

[25]. M. Aazam, P. P. Hung and E. Huh, "Smart gateway based communication for cloud of things," 2014 IEEE Ninth International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP), Singapore, 2014, pp. 1-6.

[26]. M. Aazam and E. Huh, "Fog Computing and Smart Gateway Based Communication for Cloud of Things," 2014 International Conference on Future Internet of Things and Cloud, Barcelona, 2014, pp. 464-470.

[27]. R.N. Calheiros, R. Ranjan, A. Beloglazov, R. Buyya, "CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms" in Software Practice and Experince, pp.23-50, Jan. 2011.

[28]. H. Gupta, A. Vahid Dastjerdi, S. K. Ghosh, and R. Buyya, "iFogSim: A toolkit for modeling and simulation of resource management techniques in the Internet of Things, Edge and Fog computing environments," Softw. - Pract. Exp., vol. 47, no. 9, pp. 1275–1296, 2017.

[29]. M. Aazam and E. Huh, "Fog Computing Micro Datacenter Based Dynamic Resource Estimation and Pricing Model for IoT," 2015 IEEE 29th International Conference on Advanced Information Networking and Applications, Gwangiu, 2015, pp. 687-694.

[30]. W. Lee, K. Nam, H. Roh and S. Kim, "A gateway-based fog computing architecture for wireless sensors and actuator networks," 2016 18th International Conference on Advanced Communication Technology (ICACT), Pyeongchang, 2016, pp. 1-1.

[31]. Iot-compute-dataset,https://github.com/saifulislamPhD/IoT-647Compute Dataset, 2019. Retrieved on (2019-11-22).

[32]. Y. Xiao and M. Krunz, "Qoe and power efficiency tradeoff for fog computing networks with fog node cooperation." in Proc. IEEE Conf. Comput. Commun., May 2017, pp. 1–9.

[33]. A. Pang, W. Chung, T. Chiu and J. Zhang, "Latency-Driven Cooperative Task Computing in Multi-user Fog-Radio Access Networks," 2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS), Atlanta, GA, 2017, pp. 615-624.

[34] S. Akbar, Saif U. R. Malik, S. U. Khan, R. Choo, A. Anjum, N. Ahmad, "A Game-based Thermal-aware Resource Allocation Strategy for Data Centers," IEEE Transaction on Cloud Computing, Feb. 2019.

Osama A. Khan did his BS., in software engineering degree from National University of Modern Languages, Islamabad, Pakistan in 2015 and Now, doing his MS., in software engineering from COMSATS University, Islamabad, Pakistan. His research intrests includes cloud computing and BlockChain.

Dr. Saif U. R. Malik did his Ph.D. in 2014 from Department of Electrical and Computer Engineering, North Dakota State University, USA. He worked as an Assistant Professor at COMSATS University, Islamabad Pakistan since 2014. Currently, he is a Senior Researcher at Cybernetica, AS Estonia. His areas of expertise include the application of Formal Methods in Large Scale Computing Systems, Distributed Computing, Data Centres, Security and Routing Protocols, and IoT. His research work appears in several reputable journals and transactions. He is also serving as a reviewer and TPC of many well reputed Journals and transactions.

Faizan Murtaza did his BS., in computer science degree from Fedral Urdu University of Art & Technology, Islamabad, Pakistan in 2013 and Now, did his MS., in software engineering from COMSATS University, Islamabad, Pakistan in 2018. He is currently working as Sr. Software Engineer at Enabling systems private limited. His research intrests includes cloud computing and BlockChain.

Saif ul Islam received his Ph.D. in Computer Science at the University Toulouse III Paul Sabatier, France in 2015. He is Assistant Professor at the Department of Computer Science, Dr. A. Q. Khan Institute of Computer Science and Information Technology, Rawalpindi, Pakistan. Previously, he served as Assistant Professor for three years at the COMSATS University, Islamabad, Pakistan. He has been part of the European Union-funded research projects during his Ph.D. He was a focal person of a research team at COMSATS working in O2 project in collaboration with CERN Switzerland. His research interests include resource and energy management in large-scale distributed systems (Edge/Fog, Cloud, Content Distribution Network (CDN)) and the Internet of Things (IoT).

Haris Pervaiz received the M.Sc. degree in information security from the Royal Holloway University of London, Egham, U.K., in 2005, and the Ph.D. degree from the School of Computing and Communication, Lancaster University, Lancaster, U.K., in 2016. He is currently working as Lecturer at InfoLab21, Lancaster University, UK. He was a Research Fellow with the 5G Innovation Centre, University of Surrey, Guildford, U.K (2017-2018) and an EPSRC Doctoral Prize Fellow with the School of Computing and Communication, Lancaster University (2016-2017). His current research interests include green heterogeneous wireless communications and networking, 5G and beyond, millimeter wave communication, and energy and spectral efficiency. He has been actively involved in projects, such as CROWN,CogGREEN, TWEETHER, and Energy proportional EnodeB for LTE-Advanced and Beyond and the DARE project, a ESPRC funded project. Dr. Pervaiz is an Associate Editor of the IEEE Access, Editorial Board of Wiley's ETT and an Associate Editor of Internet Technology Letters (Wiley).

Hassan Malik received the B.E. degree in information and communication systems from the National University of Sciences and Technologies, Pakistan, in 2009, and the M.Sc. degree in wireless communication engineering from the University of Oulu, Finland, in 2012, and the Ph.D. degree in electronic engineering from the Institute for Communication Systems, University of Surrey, U.K, in 2017. He was a Research Assistant with the Centre for Wireless Communication, Oulu, Finland, from 2011 to 2013. He is currently a Researcher with the Thomas Johann Seebeck Department of Electronics, Tallinn University of Technology since July 2017.

Syed Hassan Ahmed [S’13, M’17, SM’18] completed his B.S. from KUST, Pakistan and his M.S./Ph.D. from Kyungpook National University, South Korea, both in computer science, in 2012 and 2017, respectively. Later, he was a postdoctoral researcher with the University of Central Florida, Orlando. Currently, he is on the faculty of the Computer Science Department of Georgia Southern University at Statesboro, where his research interests include sensor and ad hoc networks, cyber-physical systems, vehicular communications, and future Internet.