Comparative Analysis of Cloud and Fog Environment Based on Network Usage and Cost of Execution Using iFogSim

Mohammad Saqibul Alam
Dept. of Computer Science and Engineering
BRAC University
Dhaka, Bangladesh
mohammad.saqibul.alam@g.bracu.ac.bd

Sadia Jebunnesa Jabin
Dept. of Computer Science and Engineering
BRAC University
Dhaka, Bangladesh
sadia.jebunnesa.jabin@g.bracu.ac.bd

Ajmain Alam
Dept. of Computer Science and Engineering
BRAC University
Dhaka, Bangladesh
ajmain.alam@g.bracu.ac.bd

Muhammad Iqbal Hossain
Dept. of Computer Science and Engineering
BRAC University
Dhaka, Bangladesh
iqbal.hossain@bracu.ac.bd

2021 International Conference on Decision Aid Sciences and Application (DASA) | 978-1-6654-1634-4/21/$31.00 ©2021 IEEE | DOI: 10.1109/DASA53625.2021.9682295

***Abstract*—The number of IoT devices around the world is increasing exponentially with the growth of technology. But the resources we have at hand may prove difficult to accommodate all of them. To make efficient use of all the resources, we need to ensure that we get maximum output while decreasing network congestion. In this paper, we deal with such an IoT device: CCTV. In our digitised world, 24/7 surveillance has become prevalent through the use of CCTVs and the use of CCTVs will only keep growing. To better use the data from the CCTVs, we run it through video processing and analysis to extract the data we need. This computation is done on a localhost machine which requires the machine to have high specifications to be able to perform such rigorous tasks. This means that a certain machine is dedicated to a certain number of CCTVs. In our paper, we have decided to introduce fog computing to the scenario and compare the results versus the already existing architecture using iFogSim as the simulation medium. The CCTVs will send the footage to the Fog Servers instead of the local machine, where the necessary processing and analysis will be done and the results will be sent to the local machine. This, not only helps in reducing the cost of execution and total network usage from a traditional setup, allows multiple users to reuse the same computational resources in the fog server.**

***Index Terms*—Fog computing; Edge computing; IoT.**

I. Introduction

FOG computing is a decentralized network structure that connects cloud computing to the Internet of Things (IoT). It agrees with the concept that almost all IoT devices that are used by people on a regularly, will be interconnected with each other. IoT devices include smartphones, smart home appliances, smartwatches, wearable health monitoring systems, biometric cybersecurity scanners, fire detection and alarm systems, smart door lock and burglar alarm system, lighting management systems, smart bicycles, fitness trackers etc. But the sensors in IoT devices lack storage capability, resources to carry out the computational workload and analyze data. With Cloud storage in distant geolocation, it is difficult to do the required task in a reasonable timeframe. Fog computing puts one and one together. It allows data transmission between IoT devices and cloud services to be processed faster by bringing them closer to one another. Concurrently, it determines whether the information is to be stored in the cloud or the local hosts. Any device with computing and storage capability along with network connectivity can act as a fog node.

In our research, we hope to offload the computation needed to process and analyse the video footage from multiple CCTVs to a Fog Server instead of processing them locally. This will, hopefully, lead to a decrease in the cost of execution compared to a traditional Cloud setup.

As the world is being digitised, the prevalence and necessity of CCTVs has become an important factor in ensuring constant surveillance to the public eye. But with the increasing number of CCTVs, there is also a demand for video processing and analysis. Normally, said processing would be done on a local machine (a high-specification computer). We are to see if offloading the computation to a Fog server would help the situation in any way. As per our research, we have not found any papers comparing the differences between a Fog setup and a Cloud setup, so we hope to fill in that gap.

While using Cloud has provided us with innumerable facilities, with the ever increasing number of users and IoT devices, it gives rise to a variety of issues which include latency and bandwidth management, power management etc. For our paper, we are focusing on lowering the network usage and cost of execution in a CCTV footage analysis architecture. In a Cloud setup, the analysis of video footage from CCTV has to be done on a local machine(s). To meet the demands necessary for video processing and analysing, the local machine has

to have high specifications. Imagine there is a vendor who uses 2 CCTVs for his store and another vendor who uses a single CCTV. In the traditional approach, each of the vendors would require two separate high-specification PCs to carry out any video analysing and processing, if they chose to. This introduces a lot of wasted resources that could rather have been reused by multiple users.

The purpose of our research is to decrease the overall cost of execution and the total network usage in a Cloud setup by implementing a Fog setup. This would allow users to offload the necessary video processing and analysis to the Fog servers and reduce the need to have a host machine with high specifications. This would also let multiple clusters of CCTVs reuse the resources in a single Fog Server at different time intervals. Carrying out the analysis practically would require a lot of resources that we do not have access to. So, for this research, we are working within the boundaries of a simulating tool iFogSim. We hope this will also benefit other areas where offloading computational tasks from end devices to Fog servers is possible.

The paper consists of 5 sections: Introduction, Related Work, Simulation Tool, Work and Analysis and finally the Conclusion. In the Related Works section, we have touched up on previous research and work that have helped us with our analysis. In the Simulation Tool part, we talk about our approach to solving the problem at hand. We are using a simulation tool since we do not have access to the practical hardware required to carry out the research. We go into depth about how the simulation environment is set up and the specifications of the hardwares. The Work and Analysis section explains the results that we achieved from our simulation in the previous section. We end the paper with the Conclusion where we discuss some of the constraints we had to face and how this line of work could be updated in the future.

II. Related Work

Puliafito, Mingozzi, and Anastasi in their paper [1] mentions the issue of mobility support on mobile nodes in a fog environment. Implementing mobility support results in service availability, efficient bandwidth consumption, lower latency and higher responsiveness as all the resource-intensive computing is done on the fog server and IoT devices only send and receive data to and from the server. For flexible mobility, migration must be done on the topologically closest fog nodes considering migration cost and existing workload. CPU and bandwidth consumption should be limited and downtime should be minimized. To achieve service migration, techniques, such as virtualization and migration, methods to find a set of possible target fog nodes and to be able to adapt with the change of IP addresses when the end-users are mobile and moving, are needed. In a stateless fog service, migrate the targeted fog nodes and redirect them to the mobile nodes. In [2] and [3], the authors introduced the idea of FMC and FME which helps with user mobility but only in cellular networks. In [4], FMC is refined to help users connect to different networks but increases the threat of service as IP keeps changing when the users change location. In [5], the writers model and predict the user mobility through a 1D mobility pattern and model the service migration process as a distance-based MDP. In [6], Wang. et al. introduced a newer algorithm to reduce the complexity from O(N3) to O(N2) by changing to a 2D random walk mobility model. In [7], the authors present SENGUE that performs migration by collecting key parameters, analysing and predicting QoS violations, and determining the optimal node executing MDP.

Kenitar, Arioua, Younes, Radi and Salhaoui [8] propose using the network coordinate system NC to imitate the high latency networks. Real-time bandwidth vulnerable applications in densely distributed IoT devices. By considering the energy utilization equation as a vector 1D and bandwidth optimization equation as a sum of 1D vectors, 2 pairs of equations were developed for both fog and cloud. Energy transmission of the unit packet between end nodes (α,β) and total data size (DN, DC) for a request (k) were taken as parameters for the energy consumption equation of fog and cloud respectively. ϕf(t) = α.$\sum$(DN(k) DC(k)) and ϕcld(t) = (α+β).$\sum$(DN(k)). The delays in unit byte data transmission (μ,θ) from end nodes to the fog and cloud, storage (s) and processing (p) were taken into consideration for the latency equation for fog and cloud respectively : ρf(t) = μ.$\sum$((DNs(k) + DNp(k))(DCs(k)+ DCp(k))) and ρcld(t) = (ν+μ).$\sum$((DNs(k) + DNp(k). The proposed conceptual model contains physical, middleware, and cloud packaging. All the nodes are presumed active and data is sent to a fog enabled device. However, the bandwidth is lower than traditional cloud architecture. The developed algorithm works by importing data, computing the bandwidth transmission of data per unit, and computing the data that needs to be forwarded. In the same manner, the amount of energy consumption is determined. The size of the data and the number of nodes were adapted as variables to check the effectiveness in respect of bandwidth and the energy utilization in fog and cloud architecture. The results indicate a decrease in transmission of latency and energy optimization scenarios where fog architecture was used.

A. Simulation Tool

We are comparing the network usage in the topologies of a Cloud vs Fog. In the Cloud topology, we assume that the video processing and analysis is done on a local machine and data is sent to the Cloud purely for storage purposes. In the Fog setup, however, we offload the computational video processing to the fog servers and only the result data is sent to the localhost. We have decided to use iFogSim to achieve the simulation.

B. iFogSim

1) Introduction to iFogSim: iFogSim is a Java-based simulation toolkit used to understand latency in an end-to-end connection, congestion in a network, power consumption and other factors in a fog environment. In identifying such factors beforehand, it helps to build the proper set up in the future. According to the authors [9], iFogSim has 3 components:

- Physical components: Fog devices are an example of physical components,where the lower level fog devices are connected to sensors and actuators. Compared to a cloud setup, Fog devices play the same role as data centers where they offer computational resources, network and memory
- Logical components: These include the AppModules(Application modules) and AppEdges. AppEdges define the dependency between two modules. Going back to a cloud setup, the AppModules are mapped with VMs and AppEdges dictate the logical flow of data between VMs.
- Management components: The Controller and Module objects are the management components in iFogSim. The accessible resources of a Fog Device are recognized by the Module Mapping object as per the specifications of the AppModules. If a Fog Device is not able to meet the demands of a module, the module is forwarded to a higher level Fog Device. The Controller object initiates the AppModules on their designated Fog Devices after being placed by the Module Mapping object. After the simulation is done, the Controller object collects the results which contain the consumption of energy, usage of network and cost during the simulation.

2) Topologies of Fog and Cloud: In the Cloud paradigm, users connect to a data server over the internet, where information is stored and accessed anytime and from anywhere with an internet connection. In the Cloud architecture, there are mainly two parts: the Front End, where there is the Client's machine, and the Back End, which consists of the different components that make up a Cloud environment like Storage, Servers, Networking etc. The communication between the two parts is done over the internet.

The Fog paradigm works hand in hand with the Cloud, where the devices are arranged in a three-step hierarchy. End devices at the lower level have IoT actuators and IoT sensors connected to them. The gateway devices are the fog servers that are placed between the cloud and the end user. To make matters easy, we have considered the fog devices of the same level to be homogeneous and have maintained the same sensing frequency for all the sensors.

Figure 1 depicts the logical topology of a general fog setup. It is assumed that the ClientModule in Figure 1 is set in the end devices and the StorageModule in the cloud. The MainModule is placed in the fog server and needs a set level of computational resources to start. If other end devices require extra resources to perform tasks in a given time, they do so by requesting adjacent fog nodes.

MainModules are placed in the gateway Fog Device where the deadlines of different connected end devices and the resource availability of the Fog Device are identified. This flow of tasks is shown in Figure 2. We start by placing the MainModule in a gateway Fog Device and determining all the adjacent Fog Devices. Then the deadline based QoS and the resource requirement of the Fog Devices are implemented. Then, we determine if the current Fog Device has the resources

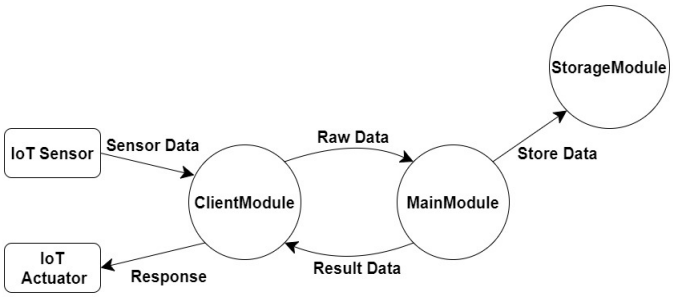

Fig. 1. The logical topology of a fog setup

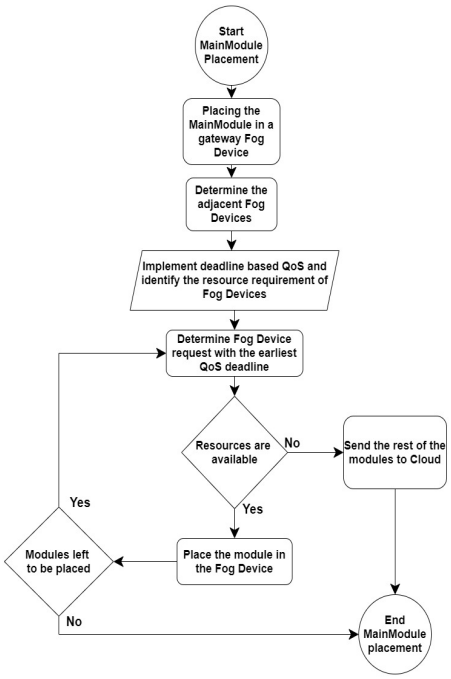

Fig. 2. Flowchart of the module placement in a fog environment

available to meet the earliest QoS deadline. If it does, then we place the module in the Fog Device and keep repeating it if there are other modules left to be placed. If, at any point, the Fog Device does not have the necessary resources available, the modules are sent to the Cloud to be processed.

3) Application in a real life scenario: To compare the results of the two approaches, we have decided to study a CCTV system, where footage from the IoT sensor is sent to the camera which is further processed either locally or on a fog server.

For the Cloud setup, as shown in Figure 3, we have considered that all the CCTVs are connected to a local host in the LAN, say an admin's PC. The footage is then analysed for facial detection or whatever the user might intend to do, in the Admin PC. After the processing is done, the result is shown on an IoT actuator (the user's monitor) and data is sent to the Cloud just for storage. We assume that the Admin PC acts as a gateway as the data to be stored in the Cloud is sent

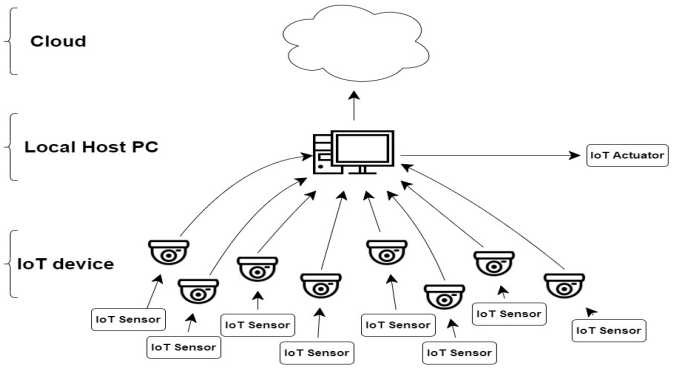

Fig. 3. The physical topology of the Cloud CCTV setup

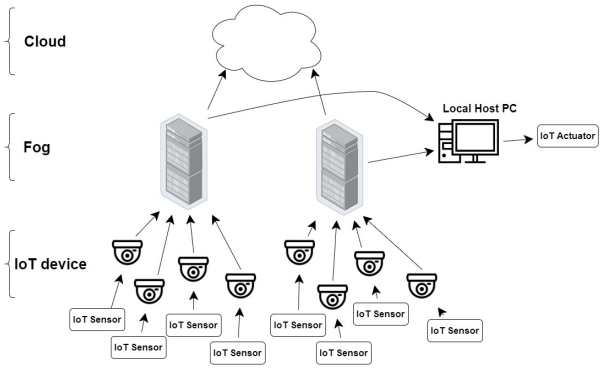

Fig. 4. The physical topology of the Fog CCTV setup

from here.

TABLE I
SPECIFICATIONS OF HARDWARE IN THE CLOUD CCTV TOPOLOGY

Parameter	Cloud	Admin PC	CCTV
Level	0	1	2
CPU length (MIPS)	44800	3500	1500
Rate per MIPS	0.01	0	0
RAM(MB)	40000	8000	1000
Downlink Bandwidth	10000	10000	-
Uplink Bandwidth	100	10000	10000
Idle Power	16*83.25	83.4333	82.44
Busy Power	16*103	107.339	87.53

In the Fog setup, as shown in Figure 4, a fog server is dedicated for every 4 CCTVs. CCTVs are connected to the fog server where there is a higher computational capability than in the localhost. Data is processed in the server and the result is sent to the localhost (Admin Pc) and shown on an IoT actuator. Data is sent to the Cloud for storage. This allows for the Admin PC to not have high-end specifications which the data analysis and processing might require, lowering the cost in that sense.

In the logical topology of the Fog setup Figure 5 , the IoT sensor in the CCTV sends the SensorData to the CCTV module. The CCTV sends the RawData to the Fog Server module for processing. After it is done, the Fog Server sends

TABLE II
SPECIFICATIONS OF HARDWARE IN THE FOG CCTV TOPOLOGY

Parameter	Cloud	Fog Server	CCTV
Level	0	1	2
CPU length (MIPS)	44800	2800	1500
Rate per MIPS	0.01	0	0
RAM(MB)	40000	4000	1000
Downlink Bandwidth	10000	10000	-
Uplink Bandwidth	100	10000	10000
Idle Power	16*83.25	83.4333	82.44
Busy Power	16*103	107.339	87.53

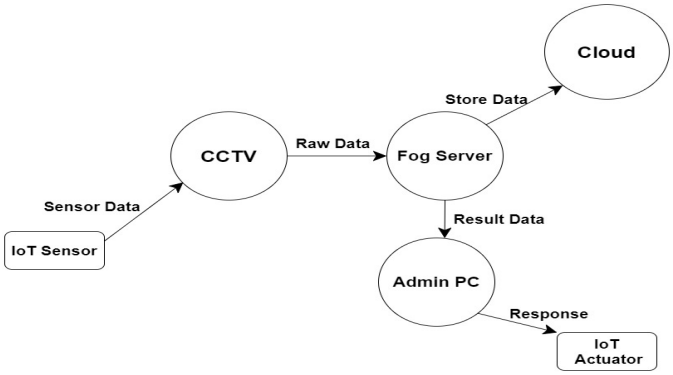

Fig. 5. The logical topology of the Fog CCTV setup

StoreData to the Cloud (for storage of data) and ResultData to the Admin PC. The Response is then shown on the IoT Actuator of the Admin PC.

In the logical topology of the Cloud setup Figure 6, The IoT sensor in the CCTV sends SensorData to the CCTV module. The CCTV module then sends the RawData to the Admin PC module where the data processing and analysing is done in the same module. After computing the ResultData, it sends StoreData to the Cloud and the Response is shown on the IoT Actuator of the Admin Pc.

III. WORK AND ANALYSIS

We wrote the code in iFogSim with the specifications given in Table I for the Cloud topology and Table II for the Fog topology. The code was run for 4, 8, 12, 16, 20 and 24 CCTVs in both the environments.

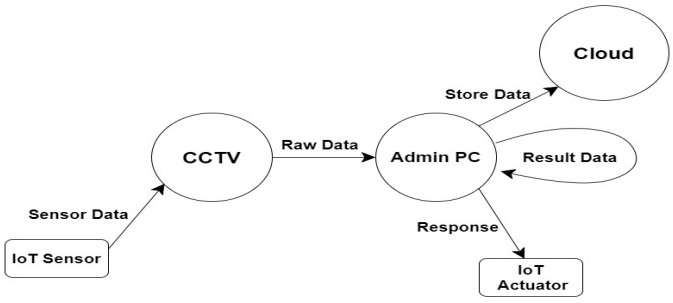

Fig. 6. The loogical topology of the Cloud CCTV setup

In the Fog environment, each fog server is allocated with 4 CCTVs. So, for 4 CCTVs, there was 1 fog server. For 8 CCTVs, there were 2 fog servers and so on.

For the Cloud environment, however, all the computational work is done on the local host PC. So, 4,8 or 24 CCTVs, are connected to the 1 local PC.

After running the code in iFogSim, the results tell us the execution time, the Application Loop Delays, the Tuple CPU Execution Delay and other variables. An example of how the results look like are given below:

```
=========================================
============ RESULTS ====================
=========================================
EXECUTION TIME : 727
=========================================
APPLICATION LOOP DELAYS
=========================================
[IoTSensor, clientModule, adminPcModule, IoTActuator] ---> null
=========================================
TUPLE CPU EXECUTION DELAY
=========================================
IoTSensor ---> 0.12000000000080036
=========================================
cloud : Energy Consumed = 1.6448470535714285E7
g-i+ : Energy Consumed = 834332.9999999987
e-i+-0 : Energy Consumed = 826146.6843999757
e-i+-1 : Energy Consumed = 826146.6843999757
e-i+-2 : Energy Consumed = 826146.6843999757
e-i+-3 : Energy Consumed = 826146.6843999757
e-i+-4 : Energy Consumed = 826146.6843999757
e-i+-5 : Energy Consumed = 826146.6843999757
e-i+-6 : Energy Consumed = 826146.6843999757
e-i+-7 : Energy Consumed = 826146.6843999757
Cost of execution in cloud = 4435300.0
Total network usage = 5754.24
```

Fig. 7. Results of running 8 CCTVs in a Cloud topology

For our paper, we are looking at the cost of execution and the total network usage, shown in the bottom two lines of the results from running the code, and how they differ from a Fog topology to a Cloud topology.

In [10], the writers describe the total execution cost as

$$E = T_c + (C_i \times L_{time} \times R_{MIPS} \times L_u \times T_{MIPS})$$

where T_c = execution cost, C_i = CloudSim clock, L_{time} = last utilization update time, R_{MIPS} = rate per MIPS, L_u = last utilization and T_{MIPS} = total MIPS of the host.

The results of running the simulation for CCTVs numbering from 4,8,12,16,20 and 24 in a Cloud environment are given in Table III. The execution time and network usage increases with the increase in the number of CCTVs. The application loop delay stays null throughout all the simulations and the cost of execution stays constant at 4435300.

The results of simulating for CCTVs numbering from 4,8,12,16,20 and 24 in a Fog environment are given in Table IV. The execution time, cost of execution and the network usage increases with the increase in the number of CCTVs. The application loop delay stays constant at ≈ 10.

```
=========================================
============ RESULTS ====================
=========================================
EXECUTION TIME : 2487
=========================================
APPLICATION LOOP DELAYS
=========================================
[IoTSensor, clientModule, mainModule, adminPcModule, IoTActuator] ---> 9.537500000002636
=========================================
TUPLE CPU EXECUTION DELAY
=========================================
RawData ---> 1.9750000000003638
ResultData ---> 0.1312500000003638
IoTSensor ---> 0.1312500000003638
StoreData ---> 0.2403529411760032
=========================================
cloud : Energy Consumed = 1.3524429432980865E7
g-0 : Energy Consumed = 834332.9999999987
e-0-0 : Energy Consumed = 847210.9622500865
e-0-1 : Energy Consumed = 847210.9622500865
e-0-2 : Energy Consumed = 847210.9622500865
e-0-3 : Energy Consumed = 847210.9622500865
g-1 : Energy Consumed = 834332.9999999987
e-1-0 : Energy Consumed = 847210.9622500865
e-1-1 : Energy Consumed = 847210.9622500865
e-1-2 : Energy Consumed = 847210.9622500865
e-1-3 : Energy Consumed = 847210.9622500865
Cost of execution in cloud = 289824.00625007175
Total network usage = 2876.16
```

Fig. 8. Results of running 8 CCTVs in a Fog topology

TABLE III
RESULT ANALYSIS OF THE CLOUD ENVIRONMENT

Number of CCTVs	Execution time	Application loop delay	Cost of execution	Network usage
4	487	-	4435300.0	2877.12
8	727	-	4435300.0	5754.24
12	1036	-	4435300.0	8631.36
16	1210	-	4435300.0	11508.48
20	1630	-	4435300.0	14385.6
24	1742	-	4435300.0	17262.72

Figure 9 shows the comparison between the cost of execution in the Fog topology and the Cloud topology. The cost in the Cloud topology is constant at 4435300, while the cost in the fog setup is considerably lower at the beginning, increasing with the introduction of each Fog server, even so, very slightly.

Figure 10 shows the comparison between the total network usage between the Cloud CCTV topology and the Fog CCTV topology. With an increasing number of CCTVs, both the graphs increase but the Cloud setup has a much steeper

TABLE IV
RESULT ANALYSIS OF THE FOG ENVIRONMENT

No of CCTVs	Execution time	Application loop delay	Cost of execution	Network usage
4	1547	9.5375	109454.9662	1438.08
8	2487	9.5375	289824.00625	2876.16
12	3254	9.5375	409644.00625	4314.24
16	4089	9.5375	489524.00625	5752.32
20	5264	10.02	529143.7849	7192.8
24	6098	10.02	609023.7849	8631.36

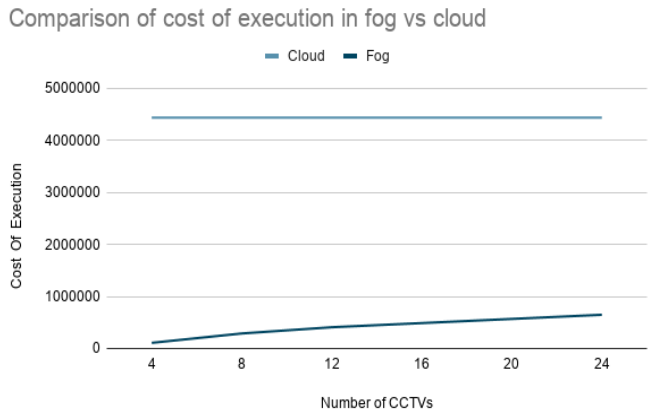

Fig. 9. Comparison of cost of execution in Fog vs Cloud

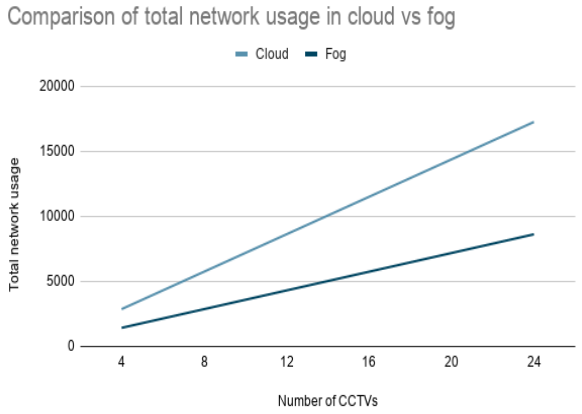

Fig. 10. Comparison of total network usage in Cloud vs Fog

increase than the Fog setup, meaning the total network usage per number of CCTVs in a Fog topology increases at a much lower rate than in a Cloud topology.

IV. Conclusion

The implications of Cloud computing in our everyday life are rising by the day. But it is starting to face problems in terms of bandwidth and latency with the increase in the number of people and IoT devices. In our paper, we have shown that offloading the computation of CCTV footage for video processing and analysis to a Fog Server, rather than doing it in a local machine, can help in decreasing the total network usage and the cost of execution in a network. We have managed to simulate this with the help of the Java-based simulation kit iFogSim, where we modelled the logical topologies of Fog and Cloud and compared the results of the two environments. From our findings, we have managed to show that implementing a Fog topology can, help in decreasing the network usage and cost of execution in a network when compared to a Cloud topology. Offloading the computational work to a fog server also introduces the concept of allowing multiple users to reuse the same fog server to carry out different complex jobs.

There were multiple challenges that we faced during our research. One of the major challenges we faced was the lack of previous work done on this topic. Most of the papers written on the topic of fog computing were theoretical, talking about the architecture of fog computing, rather than about the application of it. Another issue that affected our work was the lack of resources, especially with the ongoing pandemic. Initially, we were hoping to allocate more than 4 CCTVs to a single Fog Server but were unable to, as our PCs simply could not run the simulation. Also due to the limitations of time and iFogSim, we were unable to showcase other performance metrics such as latency and throughput. We plan to improve on these shortcomings of our thesis and also hope our findings will help and benefit areas of research where our approach of offloading computational work to a fog server can be implemented.

Acknowledgment

Firstly, all praises to Almighty Allah for blessing us with the opportunities to complete our thesis without any major obstruction in this time of a global pandemic. Secondly, we would like to extend our heartfelt gratitude to our supervisor Dr. Muhammad Iqbal Hossain for helping us and guiding us, offering his support and advice whenever we needed it. We would like to thank our families and friends who have supported us in these trying times, offering us mental support. Finally, we express our sincere appreciation to the University of Bahrain for allowing us to publish our paper in such a reputed conference.

References

[1] C. Puliafito, E. Mingozzi, and G. Anastasi, "Fog computing for the internet of mobile things: issues and challenges," in *2017 IEEE International Conference on Smart Computing (SMARTCOMP)*, pp. 1–6, IEEE, 2017.

[2] T. Taleb, S. Dutta, A. Ksentini, M. Iqbal, and H. Flinck, "Mobile edge computing potential in making cities smarter," *IEEE Communications Magazine*, vol. 55, no. 3, pp. 38–43, 2017.

[3] T. Taleb and A. Ksentini, "Follow me cloud: interworking federated clouds and distributed mobile networks," *IEEE Network*, vol. 27, no. 5, pp. 12–19, 2013.

[4] A. Ksentini, T. Taleb, and F. Messaoudi, "A lisp-based implementation of follow me cloud," *IEEE Access*, vol. 2, pp. 1340–1347, 2014.

[5] A. Ksentini, T. Taleb, and M. Chen, "A markov decision process-based service migration procedure for follow me cloud," in *2014 IEEE International Conference on Communications (ICC)*, pp. 1350–1354, IEEE, 2014.

[6] S. Wang, R. Urgaonkar, M. Zafer, T. He, K. Chan, and K. K. Leung, "Dynamic service migration in mobile edge-clouds," in *2015 IFIP Networking Conference (IFIP Networking)*, pp. 1–9, IEEE, 2015.

[7] W. Zhang, Y. Hu, Y. Zhang, and D. Raychaudhuri, "Segue: Quality of service aware edge cloud service migration," in *2016 IEEE International Conference on Cloud Computing Technology and Science (CloudCom)*, pp. 344–351, IEEE, 2016.

[8] S. B. Kenitar, M. Arioua, A. Younes, M. Radi, and M. Salhaoui, "Comparative analysis of energy efficiency and latency of fog and cloud architectures," in *2019 International Conference on Sensing and Instrumentation in IoT Era (ISSI)*, pp. 1–5, IEEE, 2019.

[9] R. Mahmud and R. Buyya, "Modelling and simulation of fog and edge computing environments using ifogsim toolkit," *Fog and edge computing: Principles and paradigms*, pp. 1–35, 2019.

[10] S. R. Hassan, I. Ahmad, S. Ahmad, A. Alfaify, and M. Shafiq, "Remote pain monitoring using fog computing for e-healthcare: An efficient architecture," *Sensors*, vol. 20, no. 22, p. 6574, 2020.