International Journal of Grid and Distributed Computing
Vol. 11, No. 5 (2018), pp.29-44
http://dx.doi.org/10.14257/ijgdc.2018.11.5.03

Refinement of Resource Management in Fog Computing Aspect of QoS

Debabrata Sarddar[1], Sanjit Barman[2], Priyajit Sen[3] and Rajat Pandit[4]

[1,2,3]*Department of CSE, Univerasity of Kalyani, Kalyani, Nadia, West Bengal, India*
[4]*Department of Computer Science, West Bengal State University, Barasat, West Bengal, India*

Abstract

The new era of Internet of Everything (IoE) is progressively making items online, but centralized cloud data processing does not scale to desires of the environment. The performance degradation can be seen in the cloud applications which are deadline like health monitoring and which need lesser response time and delay (instigated by transfer of data to cloud and then cloud to application). Fog computing appears as a solution to reduce the network congestion and latency as in this approach cloud is boosted to the area of the network. Fog computing is a model for handling a virtual and extremely distributed environment which delivers network and compute services between cloud data centres and users. In this paper, a Refinement of Resource Management in Fog Computing Aspect of QoS is proposed for efficient management of resources which considers execution time.

Keywords*: Fog Computing, Cloud Computing, Big Data, Quality of Service, Internet of Things, Resource Management, Energy efficiency, Edge computing*

1. Introduction

Internet of Things (IoT) environment consists of heterogeneous networks. Data from loosely connected different IoT devices is collected and processed to make smarter decision in a particular time period using predictive analysis or pattern detection techniques [1]. Data collected from IoT devices is of two types: Big data and Little data. Big data is that data which is stored in a centralized cloud repository while little data is transient data collected continuously by IoT devices. The data is coming in large variety and volume from devices through IoT sensors and satellites (GPS systems). As a result of regular capturing and collection of datasets, these data sets grow with the velocity of 110 MB/minute or more [2]. Big data processing is done at infrastructure level by cloud which is computing platform with high scalability based on the changing requirement of IoT application these platforms can be conFigd using pay per use mode to reduce cost to develop an IoT application. The requirement of Big data processing can be matched to store the data in cloud repository [3]. Both type of data are being used in IoT environments of smart cities for efficient decision making and real-time analytics. On-demand highly scalable cloud platforms can process volume of data with large magnitude. Cloud data processing cannot meet the requirements of an IoT application when low latency is required and sources of data are distributed across different sites. To solve this problem, there is need of an alternative paradigm which is called Fog computing. The architecture of Fog computing has three main layers: Cloud and Big data, Fog Computing and IoT. To make a shift from cloud computing to edge network, Fog computing is introduced by Cisco [4]. It is also called Edge computing or fogging, which provides the function of networking, compute and storage service between end devices and cloud computing data centers [5]. Further, Fog computing offers support to developing Internet of Everything

Received (January 4, 2018), Review Result (March 1, 2018), Accepted (March 6, 2018)

ISSN: 2005-4262 IJGDC

(IoE) applications which require predictable/ real-time latency.

In this new paradigm, application components (dedicated Fog devices or routers and smart gateways are running between cloud and sensors on both edge devices and cloud. To fulfil the IoT application requirements like wide geographical distribution and low latency, Fog computing supports distributed data analytics, cloud integration, interface heterogeneity, communication protocols and computing resources. Following are the advantages of Fog computing as studied from literatures [1-10].

a. Applicable to IoT Tasks and IoT queries:

Due to raising the IoT devices, request to these devices can be executed without using global information available in cloud. For example, Edomondo application is used to find the location of people playing same spot. So these types of requests with local nature are processed directly in fog without cloud infrastructure.

b. Scalability:

End devices are sending raw data continuously to cloud which makes cloud bottleneck. To solve this problem, fog computing is able to process the received data nearer to the data source itself which improves the scalability to process more number of requests without putting burden of processing on cloud.

c. Reduction of Network Traffic:

Currently 25 billion devices are connected worldwide and will reach to 50 billion by 2020 as estimated by CISCO. Presently, data generated, collected and sent by tablets and smart phones are processed at cloud without focusing on central data centers. This processing of raw data on cloud is not efficient solution. Therefore, new paradigm offers a platform in which generated data is filtered and analysed by nearer edge devices to generate views of local data and reduce network traffic referred to the cloud.

d. Low Latency Requirement:

Real-time data processing is required for mission critical applications like vehicle`s anti-lock brakes, control of fly-by-wire aircraft and cloud robotics. Motion control of a robot depends upon feedback of the control system and data collected by the sensors. Due to large amount of data processing at cloud, control system does not process at the required speed which leads to communication failures. In this scenario, real-time response can be made possible using fog computing by performing processing of data needed for control system nearer to the robot.

Finally, cloud computing paradigm, in with its virtually limitless resources can be a bottleneck, if whole raw data produced with sensors (end devices) is delivered to a centralized cloud. Fog computing is proficient in filtering and processing substantial amount of arriving data edge devices, making the data processing architecture distributed and thus scalable. Resource management is the main part of the architecture and comprises of components which consistently manage resources in such a way that application level QoS constraints are fulfilled and resource wastage are reduced. To this end, scheduler component plays a key role by keeping track of the state of available resources to find the best option for hosting an application component.

The motivation of this paper is to design and propose a Refinement of Resource Management technique for efficient management of resources which considers execution time network uses, energy consumptionand latency of Quality of service parameters. The aim of this research work is i) To optimize the execution time, network uses, energy consumption and latency. ii) To implement and perform evaluation in a fog computing environment using iFogSim toolkit.

The rest of the parts are follows, related work of existing technique in Section 2. The proposed technique presents in Section 3 and the experimental steps and the evaluation of result are Section 4. In Section 5, forthcoming scope and conclusion are present.

2. State of the Art

The problem of data processing in IoT environment is solved by an emerging paradigm of Fog computing. In Fog computing. Edges devices having are processing power are used to process the user request. Processing the requests closer to data sources than cloud resources reduces the latency as well. Amir *et al.*, [6] explored the concept of Fog computing along with its emergence. Further, recent related development and applications of Fog computing are discussed. Shanhe *et al.*, [7] discussed the definition of fog computing in different contexts and introduces application scenarios. Further, it has been identified that latency, security and reliability are important QoS parameters, which can be optimized in fog computing. Amir and Rajkumar [8] introduced the concept of Internet of Things (IoT) in fog computing to enhance the quality of life. Further, authors identified that edge computing can easily handle the unprecedented amount of data which are difficult to manage in traditional system. Moreover, iFogSim id proposed to simulate the fog environment to test resource scheduling policies. Ruilong *et al.*, [9] formulated the workload allocation problem mathematically to read the trade-off between energy consumption and cloud-fog computing delay. Further, primal problem is decomposed into three sub-problem to be solved independently and demonstrated that fog computing is efficient in reducing the transmission latency and communication bandwidth. Cuong *et al.*, [10] proposed a proximal algorithm for joint resource allocation in geo-distributed environment and reducing carbon footprint. Further, alternating direction method of multipliers is developed and demonstrated that the proposed solution reduces the carbon footprint for video streaming services. Lin *et al.*, [11] proposed a cost efficient resource management technique which is integrated with medical cyber-physical system in which virtual machine placement, task distribution and base station association towards cost efficient system is investigated and shown that proposed solution performed better than greedy algorithm in terms of cost efficiency. Wangbong *et al.*, [12] proposed a gateway based fog computing architecture for wireless sensors and actuator networks which mainly consists of master nodes, slave nodes and manages virtual gateway functions, flows and resources. Swati *et al.*, [13] proposed a resources provisioning algorithm and designed an architecture using the concept of virtualization. Further, algorithm is tested on cloud-analysis simulator and it is found that proposed solution performs better in terms of cost. Literature reported [6-13] that the following open issues and challenges are required to be addressed to realize the full potential of Fog computing.

a. Energy Minimization:

Fog environment consists of large number of distributed fog devices. Here computation may consume more energy than centralized cloud environment; hence it is an important research issue. Existing research reports that the latency produced in fog environment is lesser as computing system, trade-off between delay and power consumption is an open research area.

b. Resource Management:

Resource management is the organized method of scheduling available resource to the required customer`s workloads over the Internet [1]. Applications should be allocating virtual resources in an optimized manner and workload should be executed with minimum cost and time. The effective resource management in virtual environment can improve resource utilization and user satisfaction. There are problems of under-provisioning and over-

provisioning of resources in existing resource allocation techniques. To overcome these problems, a Refinement of Resource Management in Fog Computing Aspect of QoS is proposed in this research work.

Fog devices have additional compute and storage power but it is not possible for these devices to provide the resource capacity as provided by cloud, therefore efficient resource management is process the user requests in a timely manner. To solve this problem, resource usage of users should be predicted accurately in advance for making resources utilization efficient. Moreover, existing resource management techniques in fog computing consider execution time only. In addition proposed technique requires optimizing the important QoS parameters such as execution time, energy consumption, latency and network usage.

3. QoS-aware Resource Management: Architecture

The architecture of fog computing is shown in Fig 1, in which it is divided into three main layers: Cloud and Big data, Fog Computing and IoT. Based on functionality, the architecture has four layers. End devices like gateways, fog devices and sensors lie in the lowest layer. User can interact with fog paradigm through IoT applications using IoT sensors. It is functionally enhanced by installing the intelligent and innovative applications on end devices. Data generated by this layer is collected by next layer i.e, Network management and make communications between end devices and cloud. Network management has two sub layers: Field Area Network and Internet Protocol/Multi-Protocol Label Switching (IP/MPLS). In Field Area Network, end devices are interacting with each other using 3G/4G/WiFi and IP/MPLS is used to transfer the data from end devices to the centralized cloud system. The next layer contains cloud resources and services which enable management of resources and processing of big data and IoT tasks. Finally, the topmost layer contains software to enable Quality of Services (QoS) to applications of Fog computing and manage whole infrastructure.

Many middleware-like services are implemented in this layer to optimize the use of fog and cloud resources fog IoT applications. The main aim of these services is to maintain the performance of applications by executing tasks on fog devices by keeping latency at acceptable levels and decreasing the cost of using cloud simultaneously. The number of services working in a collective manner is described below:

a. Monitoring:

The status and performance of the services and applications are estimated by this service and other services can access this information if necessary.

b. Knowledge Base:

This service is used to store the past information about demands of resources and applications to improve decision-making process.

c. Flow and Task placement:

This service process the information provided by monitoring service which contains the information about state of available cloud at a particular period of time. Further, this information is used to find the best components to contain tasks and flows coming for execution. This service is further interconnected with the Resource Provisioning service to find the requirement of new allocation of resources for execution of current number of flows and tasks.

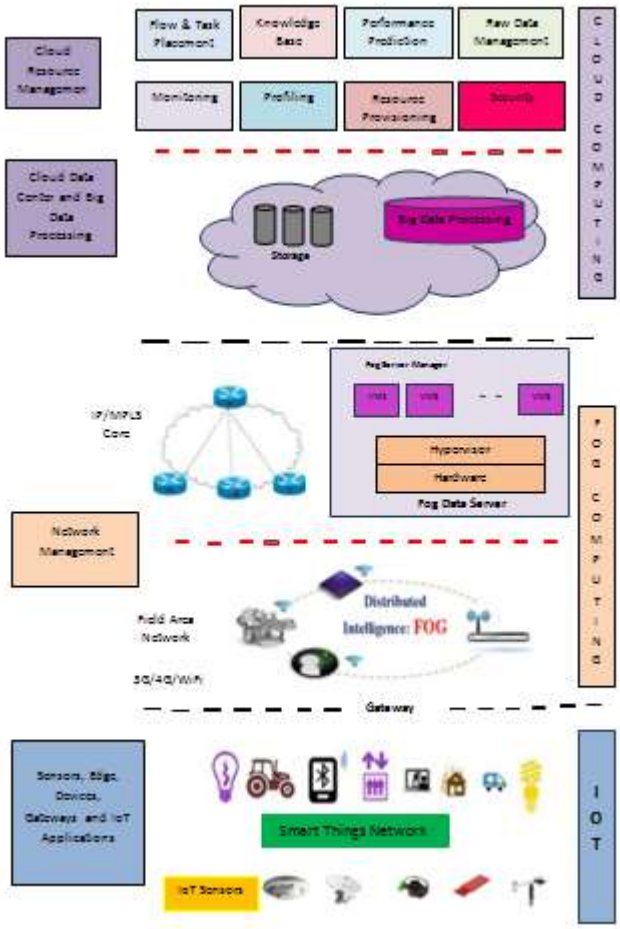


Figure 1. Architecture of Fog Computing

d. Raw Data Management:

Views about other services and data sources are directly accessed by this layer. Both complex (MapReduce) and simple querying (NOSQL REST APIs or SQL) are used to obtain views from the data for other services.

e. Profiling:

This service obtains information from Monitoring services and Knowledge Base. The information collected is used to make application and resource profile.

f. Security:

This service provides the information to applications and services about cryptography, authorization and authentication if necessary.

g. Resource Provisioning:

Fog hosting of applications, this layer attains network, fog and cloud resources. Due to change in the number of hosted applications and requirements of applications with period of time, resources are allocated dynamically. Other services like Monitoring. Performance Prediction and provides information for provisioning of resources. Security service is used to manage user credentials and user requirements on latency. For example, execution of low latency oriented tasks on fog devices, when free resources are accessible.

h. Performance Prediction:

The performance of free cloud resources is visualized by utilizing the information of Knowledge Base service. This information is further forwarded to the Resource Provisioning service to find the resource requirement to process the pending large number of tasks and flows.

3.1. Design Model

Cloud-fog environment is used to design a model having three layers as cloud layer. Fog layer (intermediate) and IoT Layer or Client layer (bottom). Initially user workloads are executed in IoT and fog layer. Further, user workload can be executed in cloud layer in case of unavailability of resources.

Step 1. All the data centres will be organized in cloud layer and fog layer. Every cloud layer has huge Cloud Data Centres (CDC) and fog layer has no. of fog data servers (FS).

Step 2. Each FS will contain Fog Server Manager (FSM) which will check the resource availability and accountability to supervise Virtual Machines (VMs).

Step 3. Firstly, any user will submit his/her request (workload) to any FS and then FS load the request to its FSM.

Step 4. Following conditions are shown the FSM will process the service request:

If all demanded resources are available to FS, then it process the user workload and user submit an acknowedgement to FSM regarding execution status.

Ifonly some demanded resources are available to FSthen user workload is divided into number of subtasks as per resource availability.

If FS is already executing other workloads but at initial release state, then user needs to wait, user workload will be processed after execution.

If all the resources executing workloads at one FS but some suffer deterioration during execution thenit will again process the workload as in (ii) condition.

If all the resources are unavailable in FS within its fog cluster then the workload is forwarded to cloud data server.

Step 5. If the user has not obtained the expected result for his/her request within maximum allocated time then users have to wait for further processing.

Step 6. For further processing user workload is transmitted to CDC.

Step 7. CDC will deliver the resource to user directly to improve the response time and sends an acknowledgement to respectiveFSM.

3.2. Functional Components

The following three different functional components used in this algorithm.

Role of FSM: It contains list of available resources to process user workloads.

Role of VMs: To process the user workload for FS and forwards the execution result to

FSM.
Role of FS: It comprises one FSM and no. of VMs to process user workloads.

3.3. Proposed Algorithm

[Algorithm 1] shows the pseudo code of QoS-aware Resource Allocation Algorithm. The main aim of this algorithm is to offer efficient utilization by decreasing the congestion using the fog layer (intermediate).

Algorithm 1: QoS-aware Resource Allocation Algorithm
if : The request from user to FS. ic: The request from FS to the Cloud data server. CDCi: The request processed by the cloud server. FSi: The request processed by the fog server. VMi: VMs at the fog server. Mst: Minimum constraint time to release the resources. Max_time: Maximum allocated time to release the processor. Ti: Threshold value for each request if T: Total tasks t1,t2, t3,…: Subtasks of T. t1: no. of processors available in FS1. t2: no. of processors available in FS2. for each request if Every user request if is submitted to closest location FS (users). Every FS will process the user request FSM will be start processing of the service request. if all demanded resources are available to first FS. then FS process the user workload and user submits an acknowledgement to FSM regarding execution status. endif If only some demanded resources are available to FS. then the T is divided into no. of subtasks as per resource availability. T=t1*t2*t3*…..tn. endif if FS is already executing other workloads but at initial release state. then users have to wait for Mst and then submit its request to FS. endif if all the resources executing workloads are at one FS but some are failing during execution. then goto step 10. if all the resources are available in FS within its fog cluster. then request if is inseminated to CDC over appropriate communication network. endif Calculate the Ti threshold. if ti<= Max_time then user will receive a message “ Wait for processing” endif for each request ic Every request ic is referring to closest location CDC as FS location. Every CDC will process the service request. CDC ends the result CDC sends an acknowledgement to their respective FSM.

[Algorithm 1] allocates the resources efficiently for execution of resources and maximizes the throughput. If the users requests isn`t process in a specified time period due to unavailability of appropriate resources then middle layer (fog) forwards this request to cloud layer.

4. Experimental Setup and Results

QoS resource management technique has been tested in Fog computing based simulated environment and experimental results of QoS parameters have been presented

4.1. Simulated Environment

In this research work, basic event simulation functionalities of CloudSim [15] have been used for implementing functionalities of iFogSim architecture. CloudSimentities like datacenters and communication among datacenters is implemented through message sending operations. Therefore, events between Fog computing components in iFogSim [14] are handled by core CloudSim layer. Therefore, the core CloudSim layer is responsible for handling events between Fog computing components in iFogSim [14], iFogSim implementation is established by simulated services and entities, iFogSim classes used in this research work are:

a. FogDevice:

This class describes the hardware features of Fog devise and their relations with sensors and other Fog devices. Further, PowerDatcenter class of CloudSim is extended to make main attributes of the FogDevice class are accessible downlink and uplink bandwidths (specifying the communication capacity of FogDevice), storage size, processor and memory. Function of this class specify the resource scheduling of Fog device among application modules executing on it and their development and release after execution. The above mentioned function can override to enables developers to plug-in custom policies.

Sensor: IoT sensors are respresented by instances of the sensor class. The features of a sensor, extending from its connectivity to output aspects are represented by attributes of this class. The class holds a reference attribute to the gateway Fog device to which the sensor is attached. Tuple arrival rate at the gateway is identified using distribution of tuple inter-transmission or inter-arival time and output characteristics of a sensor.

Tuple: It makes the central unit of communication among entities in Fog. The instances of tuple class in iFogSim are represented as tuples, which is inherited from the CloudSim. The categorization of tuple and processing requirements (defined as Million Instructions (MI) is specified by the attributes of the class.

b. Application Scheduling:

The resources of the host Fog devices are scheduled to application modules. Devices resources are equally divided among all active application modules by default resource scheduler. By overriding the method updateAllocatedMips inside the class FogDevice, the application scheduling policy can be customized.

Two application module placement policies are packaged in CloudSim: edge-ward placement and cloud-only placement.

c. Cloud-only placement:

The cloud-only placement policy is based on the traditional cloud-based implementation of applications where all modules of an application are executing in datacentres. The sense-process-actuate loop in such applications is implemented by having sensors transfer-

ring the sensed data to the cloud where it is executed and actuators are informed if action is needed.

d. Edge-ward placement:

Edge-ward placement policy favors the development of application modules close to the network edge. Though, devices close to the edge of the network like access points and routers-may not be computationally powerful enough to host all operators of the application. In such circumstances, the policy iterates on Fog devices towards cloud and tries to place pending operators on substitute devices. This policy establishes the interaction between the Fog and the cloud by retaining modules both near the cloud and network edge.

A sensor generates a tuple and directs it to the gateway, to which the sensor is linked to. When tuple reaches the Fog device (gaeway), the call back function for managing an arriving tuple processTupleArrival () is called once. Tuple is directed instantly without processing if it needs to be routed to another Fog device. Otherwise, if the application module on which the tuple desires to be executed is placed on the receiving Fog device, the tuple is submitted for execution. On completion of execution of tuple, the function check CloudletCompletion () is called on the Fog device.

4.2. Experiment Performed

IoT based smart city is newly discovered concept which comprises of member of use cases of IoT from management of traffic to management of energy of buildings. Smart traffic management based case study is presented in this section which describes that how fog computing efficiently optimizes the bandwidth consumption and response time to improve the performance of an application. Sensors are simulated to generate data set and set of stream queries are running on this data to realize smart traffic management system. Finding of traffic incidents and congestion estimation (for planning of route) are two best examples of such queries. Further, performance of query "FIND_TRAFFIC_INCIDENT" has been estimated for both cloud implementation and fog infrastructure as shown below:

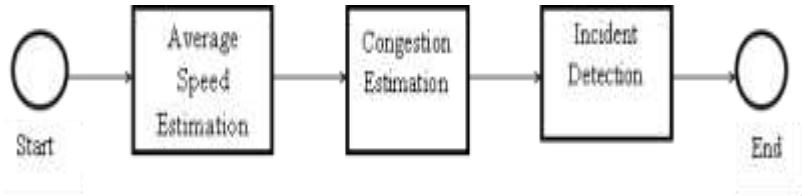


Figure 2. Directed Acyclic Graph of Query for Incident Detection

In this query, speed of every vehicle crossing the road is measured by sensors and is sent to the query processing engine as shown in Figure 2. For a particular period of time, the average speed of vehicles is estimated by an operator "Average Speed Estimation" and forward this estimated value to the next operator "Congestion Estimation". Based on average speed of vehicles, this operator estimates the congestion level in each lane. An occurrence of incident is detected by operator "Incident Detection" (based on normal level of congestion). Both cloud and fog based stream query processing engine are used to simulate this query and next section presents the comparison of both cloud and fog based processing.

The simulation of this case study is carried out for a period of 5 hours and the many metrics described by iFogSim are gathered. The results of the simulation determine how placement policies and different input workloads affect the end-to-end latency and network usage. Each camera is connected to the ISP Gateway. For the purpose of testing the performance of case study on changing topology sizes, number of WiFi gateways are tried by keeping the number of smart phones linked to each gateway continuously. In this perfor-

mance of an application on the Fog depends on latencies of the links connecting the Fog devices.

Smart traffic management system application comprises of main module which perform processing: Camera, Speed Detector, Congestion Detector, Congestion Tracker and User Interface as shown in Figure 3.

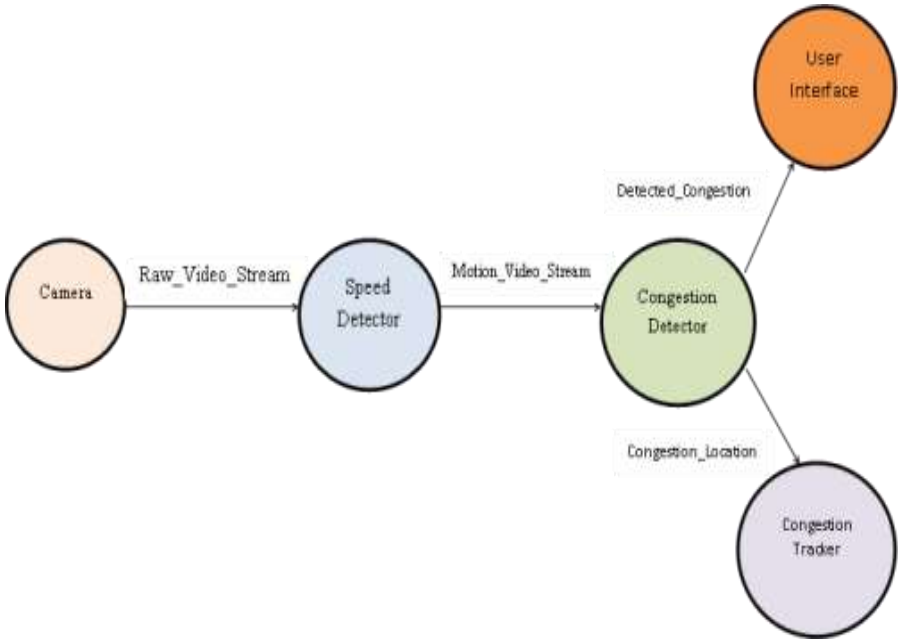


Figure 3. Application Model of Smart Traffic Management System Case Study

The application is input live video streams by a number of cameras. The functions of the above mentioned modules are as follows:

a. Speed Detection:

This module is implemented inside the smart cameras used in this experiment. To find the speed of an object, it constantly reads the raw video streams captured by the camera. In the event of detection of speed in the camera, the video stream is sent upstream to the Congestion Detection module for further processing.

b. Congestion Detection:

The Congestion Detection module gets video streams in which the smart cameras detect speed of an object. The module removes the moving object from the video streams and equates them with earlier revealed objects which are active in the area presently. Tracking is activated for this object, if detected object has not been in the area before. Further, it calculates the coordinates of the objects.

c. Congestion Tracker:

The congestion Tracker module gets the last calculated coordinates of the currently tracked objects and calculates an optimal configuration of all the cameras covering the area so that the tracked objects can be captures in an efficient way.

d. User Interface:

The application presents a user interface by transferring a fraction of the video streams comprising tracked congestion to the user's device. For this use-case, it requires such fil-

tered video streams from the Congestion Detector module to detect the incident.

The types of tuple and their CPU length and network length are described in Table 1.

Table 1. Types of Tuple and CPU Length and Network Length

Tuple Types	CPU Length(MIPS)	Network Length (Bytes)
Raw_Video_Stream	2000	50
Motion_Video_Stream	2200	66
Detected_Congestion	3500	55
Congestion_Location	3100	65

The latency of different fog devices from source to destination is described in Table 2.

Table 2. Latency of Different Fog Devices

Source	Destination	Latency(sec)
Traffic Camera	Smartphone	6
Smartphone	WiFi Gateway	2
WiFi Gateway	ISP Gateway	4
ISP Gateway	Cloud	100

The configuration (CPU GHz, RAM size and Power) of different fog devices is described in Table 3.

Table 3. Configuration of Different Fog Devices

Device Type	CPU GHz	RAM(GB)	Power (W)
Cloud VM	3.0	4	107.339
WiFi Gateway	3.0	4	107.339
Smartphone	1.6	1	87.53
ISP Gateway	3.0	4	107.339

4.3. Experimental Results and Discussions

In this experimental setup of fog devices, hierarchical network topology is used as described in [14]. In tree-like topology, cloud is situated at root of the tree and edge devices like gateways are considered as leaves. In a tree network, intermediate devices between edge and cloud are represented by intermediate nodes. Storage, network and compute capacity of these nodes are used to host applications. Both uplink network bandwidth and CPU capacity of the fog device are used to execute fog applications. Sumo (road Traffic simulator) [17] is used in which data forwarded to the query processing engine. Furthermore, cloud simulation environment [15] is used in which data center class is extended for realization of fog devices while VM is used to model stream operators. In additional, cloudlets are extended for realization of stream operators which are used to execute tuples. Only one host for every fog device is considered which provides resources for execution of fog applications. Network cost and CPU cost for every tuple are defined for processing. In this experimental work, two metrics are used to compare results of cloud (cloud-only placement) with fog (edge-ward placement): Tuple delay and Network Usage Time.

Tuple delay is defined as average processing time of tuple. Figure 4 show the E2E (End-to-End) tuple delay comparison for both cluster-based and fog-based query processing en-

gine by running the query. An experimental result shows that fog has capacity to reduce latency and save network bandwidth by placing operators on fog devices. E2E tuple delay is reduced after placing operators on fog devices in fog computing as compared to cloud based processing of data is done nearest to the IoT devices. Nevertheless, it is required to place operators optimally otherwise more delay is caused by devices due to resource contention.

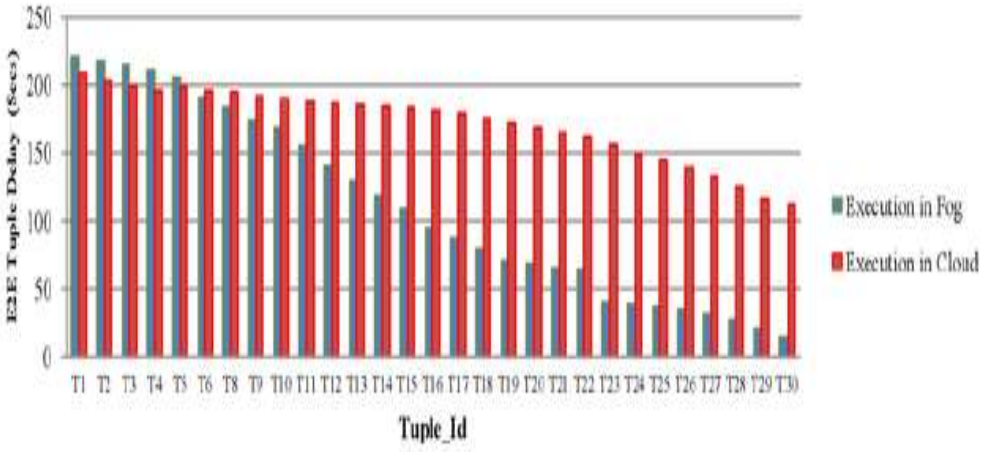


Figure 4. Variation of E2E Tuple Delay

Further, Network usage is calculated for executing query (FIND_TRAFFIC_INCIDENT) for both fog and cloud-based query execution engine and experimental results shows that small number of tuples crossing the network in fog-based as shown in Figure 5 because query execution on fog devices decreases the frequency of workload coming to the cloud. Fog-based query processing engine can optimize both the parameters (tuple delay and network usage) if operators are placed optimally on fog devices. Storage, networkand compute capacity is required in fog devices.

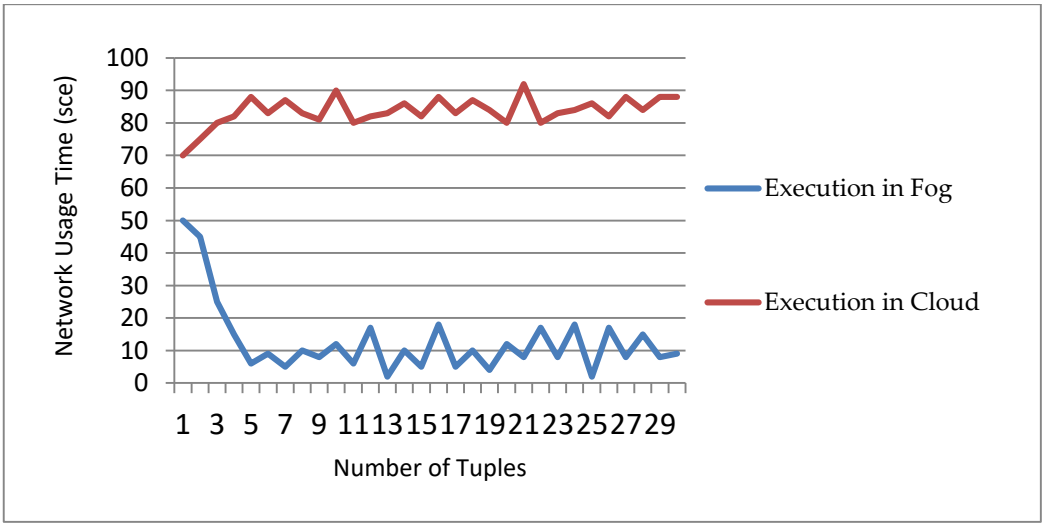
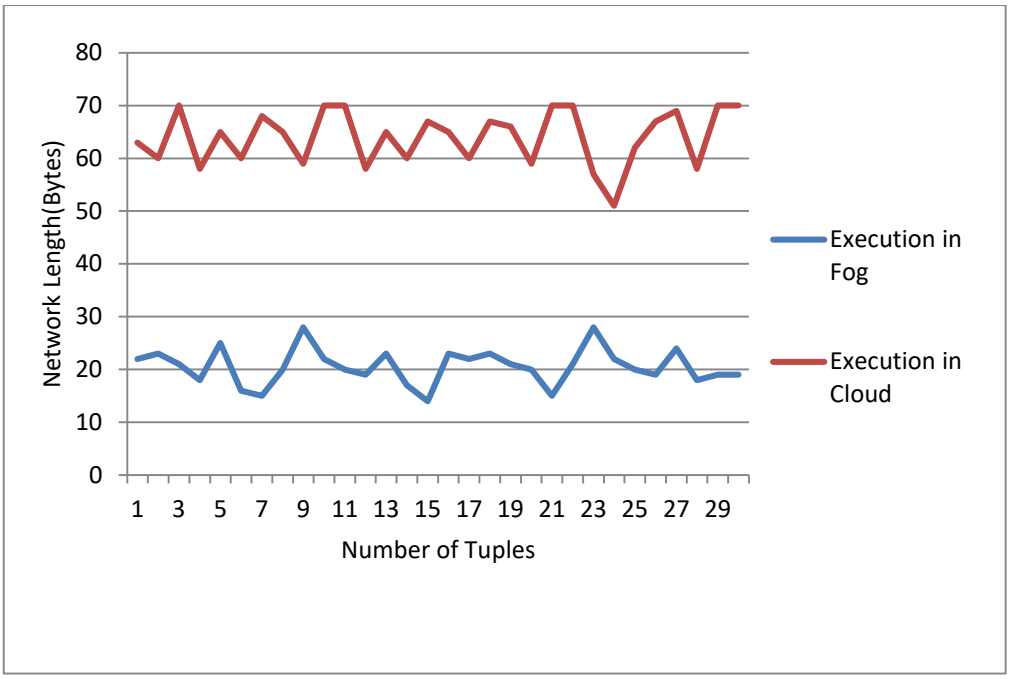


Figure 5. Effect of Time with Change in Number Tuples

Figure 6, shows the network length of different number of tuples for Fog and Cloud. It is clearly shown that fog computing performs better than cloud computing. For 23 number of tuples, network length in Fog is 54 Byte lesser than Cloud. The average network length used in Fog is 34% lesser as compared to Cloud.



Figrue 6. Effect of Network Length with various Tuples

The comparison of latency for Fog and Cloud with different number of tuples is shown in Figure 4. Fog performs better than Cloud in terms of latency. Fog reduces 16.16%-28.71% average latency as compared to Cloud.

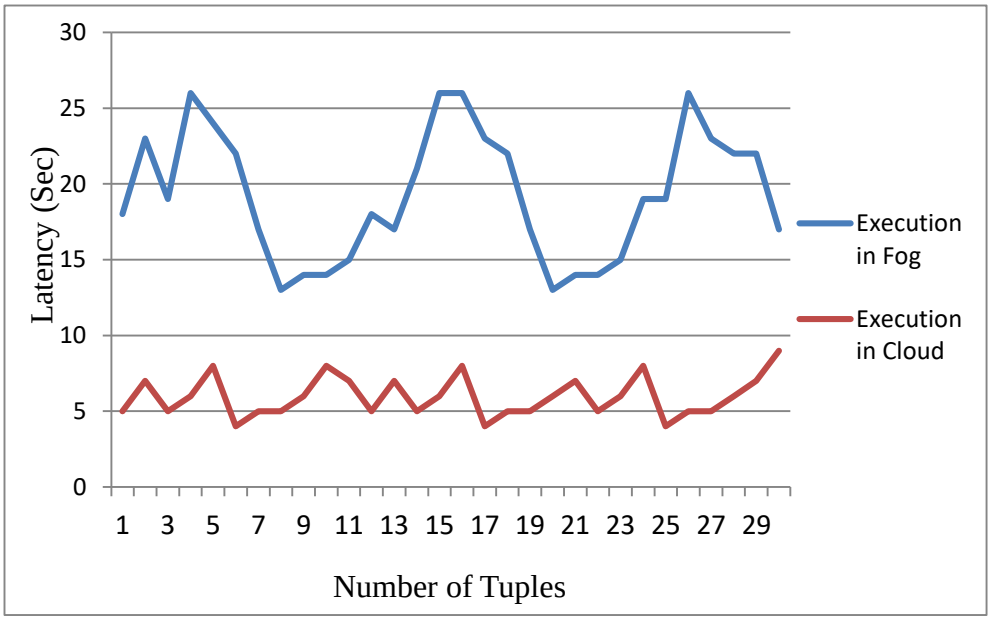


Figure 7. Effect of Latency with Change in Number of Tuples

Figure 8, shows the energy consumption of different number of tuples for fog and cloud. It is clearly shows that the Cloud consumes more energy than Fog for different number of tuples. For 29 number of tuples, energy consumption in Fog is 35.65% lesser than Cloud while for 21 number of tuples, 7.12% lesser than cloud. The average energy consumption in Fog is 22.26% lesser as compared to Cloud.

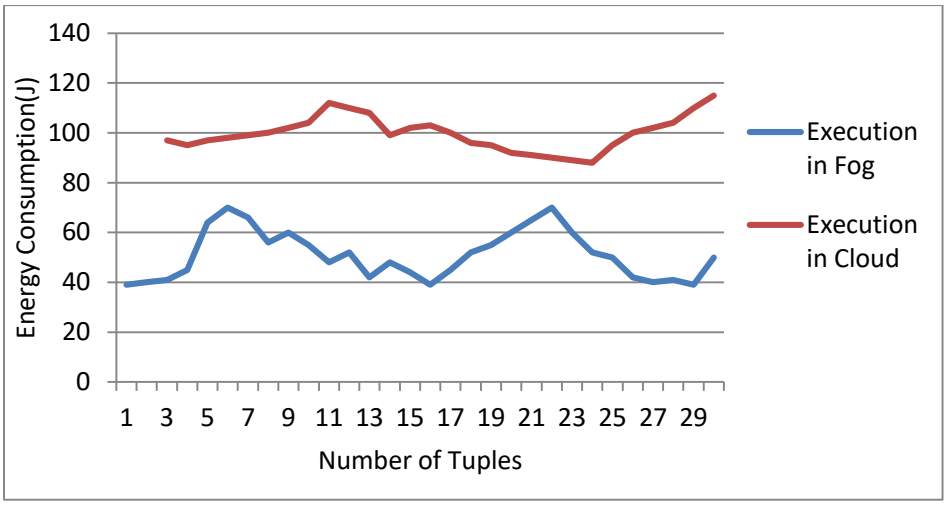


Figure 8. Effect of Energy Consumption with Change in Number of Tuples

Table 4 describes the comparison of different QoS parameters such as tuple delay, Network usage time, network length and energy consumption for Fog (Two layers) and Cloud Computing (one layer).

Table 4. Comparison of QoS parameters for Fog and Cloud Computing

Type	Processing Layer	Avg. Tuple Delay(sec)	Avg. Network Usage Time	Avg. Network Length(Bytes)	Avg. Energy Consumption
Cloud Computing	IoT to Cloud	181.17	83.16	65.71	100.29
Fog Computing	IoT to Fog	61.70	12.15	8.69	35.78
	Fog to Cloud	24.02	13.81	15.59	17.21
	Total	95.72	25.96	24.28	52.99

5. Conclusions and Future Scope

The problem of data processing in the IoT environment is solved by an emerging paradigm of Fog computing. In Fog computing, edge devices are used to process the user request which have more processing power and are closer to the data sources than cloud resources. In this research paper, QoS-aware resource management technique is proposed for efficient management of resources which considers execution time, network usage and energy consumption as QoSparameters. The performance of the proposed technique has been evaluated in Fog computing environment using iFogSim toolkit and the experimental result s show that the proposed technique performs better in terms of QoS parameters. The various open issues are still remaining though, ranging from programming models to security which needs efficient solutions. Further, the following open issues and challenges are required to be addressed to realize the full potential of Fog programming.

Programming Models: In mobile computing, computation offloading is a hotspot area but offloading to the cloud is not reasonable always. To solve this problem, adaptive Mobile Edge Computing based programming model called Cloud Aware is proposed in which scalable and elastic mobile applications are developed by offloading tasks to edge devices to provide save bandwidth, save energy and speed up computation. Further, latency can be reduced by incorporating the concept of fog computing in which application can be designed to run the components of APIs on different devices.

Security and Reliability:

It is very difficult to incorporate security protocols in fog computing due to its distributed environment. One of the main security issues is calling authentication at different levels of fog devices. Trusted execution environment and Public-key infrastructure based on authentication solutions can provide a security to fog computing. To reduce authentication cost, rogue devices can be detected using measurement-based distributed devices. Reliability is one of the main issue since fog computing comprises of a large number of geographically distributed devices. Reliable protocols for WSNs can be used in case of failure of application, service platform, network and individual sensors. Reading of sensors can be affected by noise; concept of redundancy can be used to solve the problem of information accuracy.

Acknowledgment

We would like to convey ourhearty respect and sincerest gratitude to our respected research advisor Asst. Prof. Dr. DebabrataSarddar for sparing his valuable time and assimilating new flawless ideas at every stage of our work. Without his kind co-operation, motivation, and careful guidance, we would not have been able to carry out this research work successfully.

References

[1] S. Singh and I. Chana, "QoS-aware Autonomic Resource Management in Cloud Computing: A System Review", ACM Computing Surveys, vol. 48, no. 3, **(2015)**, pp. 1-46.
[2] J. Gubbi, R. Buyyab, S. Marusica and M. Palaniswamia, "Internet of Things (IoT): A vision architectural elements, and future directions", Future Generation Computer System, vol. 29, no. 7, **(2013)**, pp. 1645-1600.
[3] L. Yao, Q. Z. Sheng and S. Dustdar, "Web-Based Management of the Internet of Things", in IEEE Internet Computing, vol. 19, no. 4, **(2015)** July-August, pp. 60-67.
[4] B. Mei, W. Cheng and X. Cheng, "Fog Compuitng Based Ultraviolet Radiation Measurement via Smartphones", Hot Topics in Web Systems and Technologies (HotWeb), 2015 Third IEEE Workshop on IEEE, **(2015)**.
[5] A. Al-Fuqaha, M. Guizani and M. Mohammadi "Internet of Things: A survey on enabling technologies, protocols and applications" Communications Surveys & Tutorials, IEEE, vol. 17, no. 4, **(2015)**, pp. 2347-2376.
[6] A. Vahid Dastjerdi, H. Gupta, R. N. Calheiros, Soumya K. Ghosh, and RajkumarBuyya, "Fog Computing: Principals, Architectures, and Applications." arXiv:1601.02752 (2016).
[7] Yi, Shanhe, Cheng Li, and Qun Li, "A survey of fog computing: concepts, applications and issues." In Proceeding of the 2015 Workshop on Mobile Big Data, pp, 37-42, ACM, 2015.
[8] Dastjerdi, Amir Vahid, and RajkumarBuyya. "Fog Computing: Helping the Internet of Things realize its Potential." Computer 49, no. 8 (2016); 112-116.
[9] Deng, Ruilong, Rongxing Lu, Chengzhe Lai, and Tom H. Luan. "Towards power consumption-delay tradeoff by workload allocation in cloud-fog computinr." In 2015 IEEE International Conference on Communications (ICC), pp. 3909-3914. IEEE 2015.
[10] Do, Cuong T., Nguyen H. Tran, Chuan Pham, MdGolamRabiulAlam, JacHyeok Son, and ChoongSeon Hong, "A proximal algorithm for joint resource allocation and minimizing carbon footprint in geo-distributed fog computing." In 2015 International Conference on Information Networking (ICOIN), IEEE, **(2015)**, pp. 324-329.
[11] L. Gu, D. Zeng, S. Guo, A Barnawr and Y. Xiang, "Cost-Efficient Resource Management in Fog Computing Supported Medical CPS", In IEEE Transactions on Emerging Topics in Computing, pp. 1-12, **(2015)**.
[12] W. Lee, K. Nam, H.-G. Roh and S.-H. Kim, "A gateway based fog computing Architecture for wireless sensors and actuator network", In 2016 18th International Conference on Advanced Communication Technology (ICACT), IEEE, **(2016)**, pp. 210-213.
[13] S. Agarwal, S. Yadav and A. K. Yadav, "An Efficient Architecture and Algorithm for Resource Provisioning in Fog Computing", International Journal of Information Engineering and Electronic Business, vol. 8, no. 1, **(2016)**, pp. 48.
[14] H. Gupta, A. VahidDasjerdi, S. K. Ghosh and R. Buyya, "iFogSim: A Toolkit for Modeling and Simulation of Resource Management Techniques in Internet of Things, Edge and Fog Computing Environments" arXiv preprint arXiv: 1606.02007, **(2016)**.

[15] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. AF De Rose and R. Buyya, "CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithm", Software Practice and Experimence, vol. 41, no. 1, **(2011)**, pp. 23-50.
[16] K. Hong, D. Lillethun, U. Ramachandran, D. Lillethun and U. Ramachandran, "Mobile Fog: A Programming Model for laege-scale Application on the IoT (Internet of Things)." Proceedings of the second ACM SIGCOMM workshop on Mobile Cloud Computing, ACM, **(2013)**.
[17] M. Behrisch, L. Bieker, J. Erdmann and D. Krajzewicz, "Sumo-simulation of urban mobility", The Third International Conference on Advances in System Simulation (SIMUL 2011), Barcelona, Spain, **(2011)**.
[18] "Dataset Collection", Available online: http://iot.ee.surrey.ac.uk:8080/datasets.html[Accessed 15-07-2016].

Authors

Debabrata Sarddar, Assistant Professor in the Department of Computer Science and Engineering, University of Kalyani, Kalyani, Nadia, West Bengal, INDIA, completed Ph. D at Jadavpur University. He completed M. Tech in Computer Science & Engineering from DAVV, Indore in 2006, and B.E in Computer Science & Engineering from NIT, Durgapur in 2001. He published more than 200 research papers in different journals and conferences. His research interest includes wireless and mobile system and Cloud computing. Email: dsarddar1@gmail.com

Sanjit Barman, completed M. Tech in Computer Science and Engineering at the Department of Computer Science and Engineering, University of Kalyani, Kalyani, Nadia, West Bengal, India in 2017. He has completed his B.E.in Information Technology from University Institute of Technology, The University of Burdwan, Burdwan, and West Bengal, India in 2015. His research interest includes Cloud Computing, Mobile and Wireless Computing, Data Structure and Algorithm. Email: s.it202.b@gmail.com

Priyajit Sen, completed M. Tech in Computer Science and Engineering in the Department of Computer Science and Engineering, University of Kalyani, Kalyani, Nadia, West Bengal, India. He had also completed his MCA from Department of Computer Science and Engineering, University of Kalyani, Kalyani, Nadia, West Bengal, India in 2015. His research interest includes Mobile Computing, Wireless Sensor Network and Cloud Computing. Email: priyajit91@gmail.com

Rajat Pandit is an assistant professor in the Department of Computer Science, West Bengal State University, Barasat, and West Bengal, India. He has completed his M.Tech (IT) from West Bengal University of Technology, West Bengal, India in 2009. He has completed his MCA from University of Jadavpur University, Jadavpur, and West Bengal, India in 2001. His research interest includes Mobile Computing, Wireless Sensor Network and Cloud Computing. Email: rajatpandit123@gmail.com

