Volume: 11 Issue: 01 Pages: 4162-4170(2019) ISSN: 0975-0290

Scalability of M/M/c Queue based Cloud-Fog Distributed Internet of Things Middleware

Dilip Rathod
Research Scholar, Dept. of Computer Science and Engineering, Research Center: Shri Guru Gobind Singhji Institute of Engineering and Technology, Under SRTM University, Nanded, Maharashtra, India.
Email: rathod.dt@gmail.com
Dr. Girish Chowdhary
Professor and Director, School of Computational Science, Swami Ramanand Teerth Marathwada (SRTM) University, Nanded, Maharashtra, India.
Email: girish.chowdhary@gmail.com

ABSTRACT

The Internet of Things (IoT) extends the Internet wherein real world things are part of a computing network. The IoT has seen exponential growth and according to Cisco prediction about 50 billion devices will be connected by 2020. Handling this massive scale is challenging research issue. In this paper, we use cloud-fog-edge based IoT middleware for distributed IoT service provisioning. We model IoT middleware using queueing network, perform analytical analysis of IoT middleware components. Followed by dynamic scaling algorithm which considers contention and coherency as limiting factors for scalability. It is used for quantitative analysis of IoT middleware with the increasing workload. The scalability function was evaluated using the simulation for important performance and scalability parameters namely throughput, CPU utilization and response time. It is observed that because of contention and coherency overhead, the proposed approach is able to scale sub-linearly which is practical compared ideal scalability of multi-server queueing network and not very restrictive as given universal scalability law(USL) applied for tightly coupled systems.

Keywords - **Fog computing, Internet of Things, middleware, queueing network, and scalability etc.**

Date of Submission: May 28, 2019 Date of Acceptance: July 15, 2019

1. INTRODUCTION

The Internet of Thing is the emerging networking and and computing paradigm of this decade. The smart city, smart grid, health care, and Industrial IoT are the major deployments of IoT system till date. According to Cisco prediction about 50 billion devices will be connected by 2020 [1]. This huge scale of devices producing data at frequencies ranging from every second/minute/hour as per application demand. The data generated by IoT system is one of the contributing factors for big data [2]. The scalability will emerge as one of pressing issue for IoT system management point of view. The current research trends IoT hardly considers scalability issue as a prime concern to be addressed.

The IoT Middleware acts as bridge between underlying IoT infrastructure and application. It helps to create, deploy and manage IoT services distributed in fog and cloud. The middleware functionalities include management of services, data, user and device context [3]. The non-functional IoT middleware requirements are scalability, heterogeneity, extensibility, security and privacy [4]. Cloud computing offers huge computing power and voluminous data storage which can be used for IoT. However it requires lot of data transfer, huge bandwidth and high latency. The fog computing is recently introduced computing paradigm placed near the edge of the IoT network. Fog computing enables service provisioning with low latency, local processing and temporary storage.

Scalability of IoT middleware (and IoT system in general) is poorly understood by the research community, in IoT literature scalability is mentioned and discussed subjectively. Quantitative analysis of scalability for IoT middleware is completely missing, which is motivation for our work in this paper. The specific contributions of this paper are as follows:

i. Modeling of IoT scale network based on queueing theory [5], mathematical treatment of performance and scalability parameters.
ii. We propose the scalability function which can be used for quantitative analysis of IoT middleware and reason upon the behavior of the IoT system particularly with respect to increased workload (both IoT devices and users). The dynamic scaling algorithm is presented for computing the optimal resource requirement to handle the current workload.
iii. Evaluate the scalability and performance of IoT middleware with respect to increased workload

Multi-server queues in ideal settings are infinitely scalable, this is far from reality. In real-world system implementation the scalability is limited by contention and coherency. The proposed scalability functions consider contention and coherency overhead, the results of implementation and simulation show that the proposed system is able to scale sub-linearly with the increase in workload or size or both.

The rest of paper is organized as follows; in Section 2 we discuss related work on the scalability of IoT middleware. In Section 3, typical IoT components and middleware internals are presented. Modeling of cloud-fog based IoT middleware using queueing theory for fog,

Volume: 11 Issue: 01 Pages: 4162-4170(2019) ISSN: 0975-0290

cloud and overall IoT system is elaborated in Section 4. In Section 5, we present scalability function based on modeling in the earlier section along with dynamic scaling algorithm. Section 6 presents the setting in which simulation is performed, and discuss results of the simulation for important performance and scalability parameters.

2. LITERATURE SURVEY

Scalability of IoT middleware is mostly neglected area by research community [3, 4, 26]. The scalability testing of IoT system can be done using quantitative measurement and quantitative analysis. In quantitative measurement, assessment of scalability is carried out by setting up testbed or benchmark system and performance of a system is monitored in a controlled environment. In [6] benchmark was set up for Fosstalk RFID middleware to analyze scalability and find out which component is a bottleneck. In [7] testbed was set up to test the scalability of components in IoT Enabled Service Architecture of FIWARE initiative [8]. Microservice based simulation of Smart city middleware in [9] conducts scalability test to handle dynamic smart city workload in the cloud environment, deploying an optimum number of VMs using auto-scaling. Scalability analysis done using quantitative measurement cannot be generalized to any kind of IoT system. The scalability of IoT middleware is required to consider diversity in size of the network, type of network, the number of users, type service provided, where data is processed; each of these factors caters to the complexity of scalability assessment problem.

Scalability and performance test of IoT middleware can be done using quantitative analysis using the proven approach of queueing theory. Queueing theory provides more reliable answers to the behavior of IoT middleware with an increase in workload as well as identifying the bottleneck in building scalable system [5]. There are only a few works using queue theory [27] for analyzing the scalability and performance of the IoT system. The work in [10] proposes edge-cloud distributed algorithm to address delay sensitive processing requirement of for augmented reality application for a citywide network, author mention about the scalability of their optimization algorithm work but details are missing. Authors in [11] propose a load balancing scheme of fog nodes by associating fog node with suitable base station node to reduce computing and communication delay. In [12] analytical model using continuous time Markov chain(CTMC) is proposed for evaluating the performance of fog node for heterogeneous fog nodes/containers. Capacity planning to determine the optimal amount of resources required at the fog, cloud-based on queueing network and Petri Nets given in [13] only initial stage of work is found, the details of work are missing. The work in [14] IoT is modeled as a closed system, experimental analysis of middleware API is done to predict the performance IoT system. In [15] the queueing model is used for resource management and scheduling. The performance analysis of IoT network consisting of cloud data center and edge computing is carried out in [16]. Our proposal differs from work in [6]-[16], i) we consider the more practical model of request handling at fog using M/M/c/N, because it has limited capacity to handle growing traffic of user and IoT devices, ii) Our focus was to evaluate the scalability of middleware, so we propose scalability function with due consideration to contention and coherency in distributed processing.

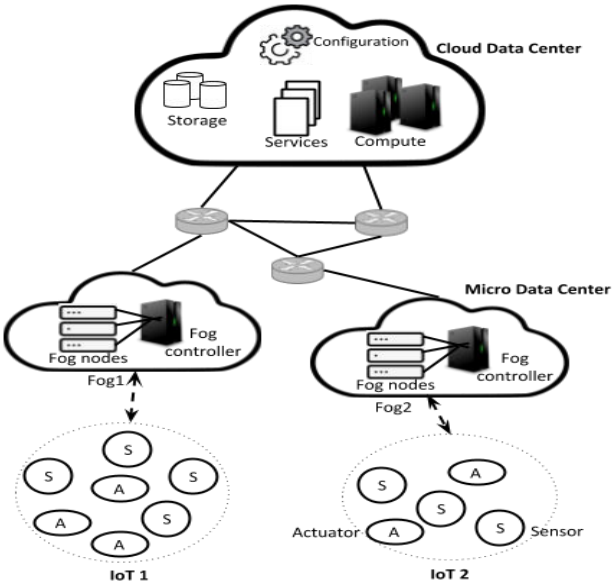


Figure1. Cloud-Fog based Internet of Things middleware architecture

3. CLOUD-FOG BASED INTERNET OF THINGS ARCHITECTURE

The cloud-fog-edge based IoT architecture is as shown in Fig. 1. The **IoT devices** are at the edge of network. It can be sensors actuators, monitoring the phenomenon of interest and producing observation/action continuously or on demand. These devices have built-in communication capability to connect nearby devices. They also have limited computing capability. IoT devices are connected to fog directly or through gateway. **Fog computing** made up of Micro Data Center (MDC). Micro Data Center consists of computing resources of two types: fog controller and fog nodes. Fog controller manages all the resources at MDC. Fog nodes provide computational capacity close to edge of IoT network, which will help in pre-processing data, fulfilling most of requests locally, only computationally intensive requests are moved to cloud, thus saving bandwidth and energy. **Cloud data center** offers on-demand services, computing power required for intensive processing, archival data storage. It uses machine learning techniques to learn from data and adjust system operation. **Users** consume the services enabled by the IoT. They connect to the IoT network through various connecting technologies (BLE, Wi-Fi) and devices (smartphones, computers). User may be local to IoT network and has access to the services locally. Remote users connect to IoT network through cloud.

Volume: 11 Issue: 01 Pages: 4162-4170(2019) ISSN: 0975-0290

3.1 Distributed IoT Middleware

Based on hierarchical structure of IoT system, the distributed IoT middleware is able to provide service orchestration in cloud and fog as shown in Fig 2.

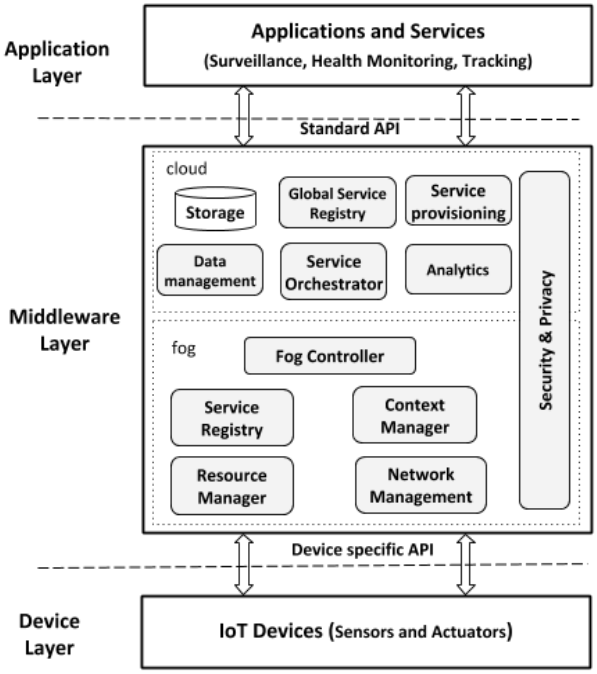


Figure 2: Distributed cloud-fog middleware

Fog controller is responsible for execution of service request locally and controls fog nodes. Fog controller can function independently if service orchestrator is unavailable. It distributes service execution request among fog nodes. *Fog nodes* interact perform sensing, actuating task either on-demand or periodically. *Resource manager* keeps track of all fog nodes, service request, service request queue, utilization, availability of gateways, and devices added to/removed from IoT network. *Service registry* maintains directory of computing service offered and their corresponding implementation. It exposes this directory through API which can be used by resource manager and fog controller. *Context manager* monitors device and user context. Context are used to provide situational awareness. *Network manager* is responsible for seamless connectivity among different elements and device mobility. IoT middleware uses multiple communication technologies for connecting IoT devices, gateways, fog and cloud. e.g. 4G, WiFi, BLE etc. Applications use standard API (e.g. HTTP or web service interface) to interact with middleware. Fog nodes use device specific APIs/drivers are used to interact with IoT devices.

4. MODELING DISTRIBUTED IOT MIDDLEWARE

Users are sending the various request to IoT middleware which includes sensing, actuation, historical data, device management related data such as adding devices, updating device profile. IoT device also keeps on sending data to middleware e.g. sensor reading upon user request, periodically, or when the phenomenon the device is monitoring is changed, the profile of the device is changed. IoT devices generated data is received by fog node for processing.. The data arrival rate λ is assumed

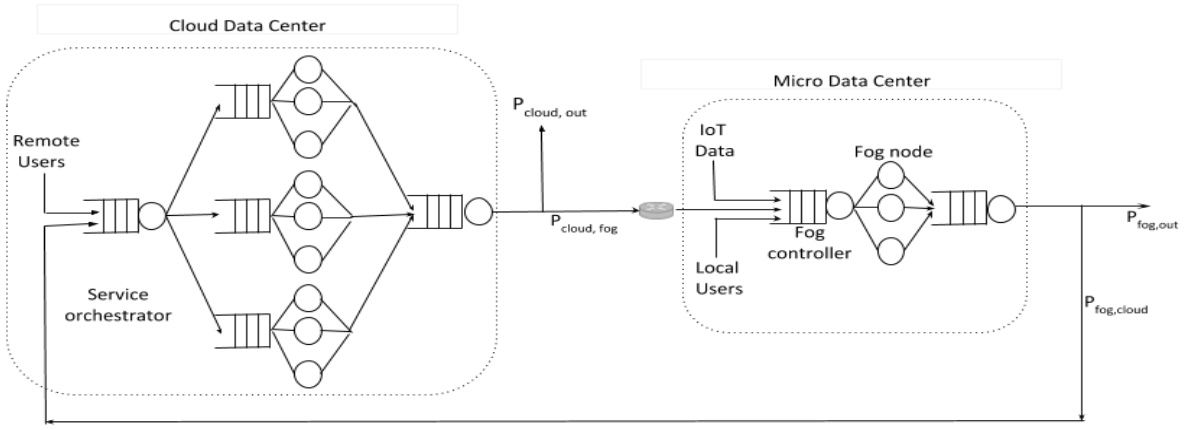


Figure3: Model of IoT Middleware using queuing network

Middleware manages distributed processing using *service orchestrator* and *fog controller*. Service orchestrator handles service which are computationally intensive or not delay-sensitive e.g. big data analytics request. Service orchestrator is central component of middleware. It controls the operation of fog controller and prepares service execution plan.

follow Poisson distribution. Fog nodes are modeled as M/M/c queue. Service rate is number of requests the fog node can execute per second and is exponentially distributed. For fog nodes, if the request processing rate from IoT device is greater than service rate. Fog node will for incoming requests to fog controller for offloading to cloud. The resource manager continuously monitors the utilization, queue length and throughput for every fog node. When the server finishes the execution of the current request, it retrieves the request waiting first come first

Volume: 11 Issue: 01 Pages: 4162-4170(2019) ISSN: 0975-0290

serve manner. All fog nodes share the same queue. Modeling IoT middleware using the network of queues is as shown in Fig.3.

A fog node can accept at most K number of request into the system, K >= c. If there is the arrival of request while the queue is full, the fog controller will search for lightly loaded nearby fog node if available otherwise offloaded to cloud.

Table1: Queueing parameters of IoT system

Notation	Description
λ_{cloud}	Net arrival rate of requests at cloud data center
λ_{fog}	Net arrival rate of requests at fog
μ	Service rate of a computing node
Π	Steady state probability
P	Load or intensity
T	Response time
T_q	Waiting time in queue
S	Mean service time
N	Number of requests in system
N_q	Number of requests in queue waiting for queue
C	Number of servers sharing same queue
P_b	Probability that request is blocked because queue is full
$P_{cloud,fog}$	Probability that request is forwarded to fog server to fulfill
$P_{fog,\ cloud}$	Probability that request is forwarded to cloud for intensive processing or for storing data to datastore
K	Queueing capacity of fog computing center
R	Minimum resource requirement to handling given arrival rate
X	Throughput
N_{users}	Number of active users
N_{iot}	Number of IoT devices managed by single MDC

4.1 Micro Data Center

The arrival rate of requests at fog (λ_{fog}) will be a combination of a portion of requests arriving from the cloud and, the portion of requested generated on-premise user and IoT devices is :

$$\lambda_{fog} = \lambda_{cloud} * P_{cloud,fog}(1 - P_b) + \lambda_{local}(1 - P_b) \quad (1)$$

$$\lambda_{fog} = (\lambda_{cloud} * P_{cloud,fog} + \lambda_{local})(1 - P_b) \quad (2)$$

The notations for parameters used in this paper are shown in Table 1. For steady state the arrival rate λ_{fog} and service time μ_{fog}should follow λ_{fog}/μ_{fog}<1. The expected number of requests $P_K$in at any time for M/M/c/K system is given below [3]:

$$P_K = \frac{(c_f\rho)^N}{c_f!c_f^{N-c_f}}P_0 \quad (3)$$

The number of requests N, requests waiting in queue N_q, waiting time in the queue T_q, response time T, utilization ρ, and throughput for fog nodes can be calculated using formulae given in Table 2.

4.2 Cloud Computing Center

The cloud computing center is modeled as M/M/c queue with the inter-arrival time between requests follows a Poisson distribution. The service time μ_{cloud} is identical for all c servers which is exponentially distributed. The requests are processed in a FCFS manner and queue has an infinite capacity to hold incoming request. The total arrival rate of requests at the cloud server is consists of a rate of requests coming from outside of network and rate of requests coming from fog:

$$\lambda_{cloud} = \lambda_{fog} * P_{cloud,fog} + \lambda_{remote} \quad (4)$$

At steady state, the flow balance equation is (subscript cloud is omitted for sake of clarity):

$$\pi_i = \frac{c\rho^i\pi_0}{i!} c \le i$$

$$\pi_i = \frac{c^c\rho^i\pi_0}{i!} c > i \quad (5)$$

The system utilization is $\rho = \lambda/c\mu$, the arrival rate of the individual server is :

$$R = \frac{\lambda_{cloud}}{\mu_{cloud}} \quad (6)$$

where R is a resource requirement, the minimum number of servers required to process given arrival rate. For example, if the arrival rate is λ = 30 req/sec and service rate μ = 5 req/sec, the minimum of 6 servers are required for a stable system. A request arrives at the data center, the probability that request has to queue because all servers are busy is:

$$P_Q = \sum_{i=k}^{\infty}\pi_i = \frac{K^K}{K!}\pi_0\sum_{i=k}^{\infty}\rho^i \quad (7)$$

$$P_Q = \frac{(c\rho)^i}{c!(1-\rho)}\pi_0 \quad (8)$$

where π_0 is:

$$\pi_0 = \left[\sum_{i=c}^{\infty}\frac{k\rho^i}{i!} + \frac{(k\rho)^k}{k!(1-\rho)}\right]^{-1} \quad (9)$$

The above equation is the Erlang-C formula for M/M/k queueing system [7]. The number of requests N, requests waiting in queue for service Nq , waiting time in queue T_q, response time T , utilization of fog system ρ, throughput for cloud can be calculated using formulae given in Table 2. The minimum resource requirement R given in (6) to service given incoming load at arrival rate λ. If the workload increases by a factor of x, the resource requirement R for the increased workload is obtained using square root staffing rule [5] as below:

$$c = R + d\sqrt{R} \quad (10)$$

where positive d is constant determined according to based on desired QoS and the second term is the additional resource (staffing) required in order to meet desired QoS given in SLA under the uncertain workload.

5. SCALABILITY

IoT is a hot research area nowadays, scalability of IoT middleware (and IoT system in general) has remained a neglected research topic. It is used ambiguously, and there is no agreed upon definitions. Bondi define [19] scalability as “the capability of a system, network, or process to handle a growing amount of work, or its potential to be enlarged in order to accommodate that growth”. This

Volume: 11 Issue: 01 Pages: 4162-4170(2019) ISSN: 0975-0290

definition can be used to understand the growing phenomenon but lacks the quantification required to measure and reason upon the behavior of the system. According to Weinstock [20], "scalability is the ability to handle the increased workload by repeatedly applying a cost-effective strategy for extending a system's capacity". In the art of scalability [21], the author introduces scalability cube in which x-axis represents breaking workload among parallel processing units, the y-axis represents breaking workload based on the type of request, z-axis represents dividing workload based on the type of user. The given system can apply any combination of x, y, and z-axis split to achieve scalability. The techniques used in the distributed system [22] for scalability

Table 2: Performance measures for IoT network of queues

Metrics	Fog (M/M/c/K)	Cloud (M/M/c)
Queue length (N_q)	$N_q = \frac{P_0(c\rho)^c\rho}{c!(1-\rho)^2}[1-\rho^{K-c}-(K-c)\rho^{K-c}]$	$N_q = P_Q\frac{\rho}{(1-\rho)}$
Number of in system (N)	$N = N_q + \frac{\lambda_{fog}}{\mu_{fog}}$	$N = \lambda T = P_Q\frac{\rho}{(1-\rho)} + c\rho$
Waiting time in queue(T_q)	$T_q = \frac{1}{\lambda_{fog}}N_q$	$T_q = \frac{1}{\lambda}N_q = \frac{1}{\lambda}P_Q\frac{\rho}{(1-\rho)}$
Response time (T)	$T = T_q + \frac{1}{\mu}$	$T = T_q + \frac{1}{\mu} = \frac{1}{\lambda}P_Q\frac{\rho}{(1-\rho)} + \frac{1}{\mu}$
System utilization (ρ)	$\rho = \frac{\lambda}{c\mu}$	$\rho = \frac{\lambda}{c\mu}$
Throughput (X)	$X_i = \lambda_i$	$X_i = \lambda_i$

are i) replication ii) caching iii) load distribution/balancing, and iv) asynchronous request handling.

According to Jogalekar [23], "scalability Ψ(k1; k2) from one scale k1 to another scale k2 is the ratio of the efficiency figures for the two cases", as given below:

$$\Psi(k1;\ k2) = \frac{E(k2)}{E(k1)} \quad (11)$$

where $E(k1)$ $E(k2)$ are efficiencies at workload $k1$ and $k2$.

In the Guerrilla Capacity Planning book [24], Gunther defines scalability as "functional relation between the independent variable and dependent variable". e.g. the relation between response time and arrival rate, number of processor and throughput, etc. Universal scalability law (USL) provides the mathematical background for scalability in parallel and distributed computing environment as described below. Speed up [24] is defined as "the ratio of time required to compute given the task in *T1* time units on uniprocessor system to the same amount of work carried out on the p-way multiprocessor system". The given task is broken into sub-task, fed to processors, executed in a parallel fashion and speed up is achieved. There is a certain amount of work which cannot be parallelized given by Amdahl's law below:

$$X = \frac{ec}{1+\sigma(c-1)} \quad (12)$$

where σ is the coefficient of contention, e is slope constant.

The contention is part of work which cannot be parallelized and queue up. Thus degrades scalability and contribute to limiting factor on scalability.

In a multiprocessor environment, every processing (core or fog nodes) element maintains own copy of data in the cache. When one processor modifies data element it needs to update at all other processor holding the same data element. This introduces cache coherency overhead, it increases quadratically with the number of processors, given by the degree of a fully connected graph. The equation (12) becomes:

$$X = \frac{ec}{1+\sigma(c-1)+\beta c(c-1)} \quad (13)$$

where β is coherency coefficient. The equation (13) is Gunther's universal scalability law. Gunther has demonstrated the applicability of USL in distributed processing e.g. three-tier web application [24], Hadoop processing [25].

Fog middleware fetches the data on behalf of the user. Fog node maintains the latest sensor readings. Even if every fog node maintains a local copy of sensor reading, it does not modify the data and hence there is no cache consistency overhead. The USL in (12) cannot be directly applied to the IoT scenario. For multi server queue M/M/c or M/M/c/K, when utilization reaches threshold say 80% or response time is more than specified in SLA, more resources are added, e.g. in a cloud-based environment, additional fog nodes are deployed to meet the current load. Such configuration of multi-server queueing systems is infinitely scalable. But it practice scalability is limited by contention as given in equation (12). IoT system as a distributed system, organized in a hierarchical fashion with cloud data center is at root, fog nodes at level 1, then gateway connecting IoT device at level 2 and finally IoT devices are at edge of network.

i. As the number of users λ users increases the service manager module will interact more frequently with the service discovery module, data store, event handler, and context manager.
ii. Increase in the number of IoT devices will cause frequent updates in the data store, device registry, context manager, which will trigger interaction among the rest of the module and with cloud computing center.
iii. Load balancer and resource manager work grows with the increase in the number of active users N_{users}, the number of IoT device N_{iot} managed by single fog computing center and varying number

Volume: 11 Issue: 01 Pages: 4162-4170(2019) ISSN: 0975-0290

of fog nodes as per current workload demand. Therefore scalability function for IoT middleware is given by:

$$X = \frac{c}{1+\sigma(c_{fog}-1)+k\frac{(aN_{users}+bN_{iot})}{c_{fog}}} \quad (14)$$

coefficients a, b are coherency introduced because of users and IoT devices respectively. Equation (14) is scalability function for IoT middleware, modeling relation between throughput X, number of fog nodes c, to handle current workload, number of active users N_{users}, and number of IoT device N_{iot} .

5.1 Performance monitoring and dynamic scaling

The following algorithm gives the procedure for monitoring the performance of the system modeled in Section 4 and Section 5. In beginning the threshold values for response time, utilization, number of requests in the queue is set. The parameter values system utilization, the number of requests in the queue, number of requests in the system, waiting time, response time and throughput, at fog are calculated for given arrival rate. The program continuously monitors values for response time, utilization, number of requests in the queue, if any of these condition turns true. The resource demand for the current arrival rate is calculated and an additional number of instances of fog nodes are deployed to meet current processing demand.

Table4: Dynamic scaling algorithm for fog computing center

Ln	Statements
1	Input: λ, μ, c, K
2	Output: Utilization, X, T and N q are below threshold
3	Begin
4	Set thresholds N_t, U_t
	while(TRUE)
	{
5	//compute all performance parameters for current
	// arrival rate
6	//System utilization
7	$\rho = \frac{\lambda}{c\mu}$
8	// Waiting for queue length (N_q)
9	$N_q = \frac{P_0(c\rho)^c\rho}{c!(1-\rho)^2}[1-\rho^{K-c}-(K-c)\rho^{K-c}u]$
10	// Number of requests at fog
11	$N = N_q + \frac{\lambda_{fog}}{\mu_{fog}}$
12	// Waiting time in queue(T_q)
13	$T_q = \frac{1}{\lambda_{fog}}N_q$
14	// Response time
15	$T = T_q + \frac{1}{\mu}$
16	// Throughput
17	$X = \frac{c}{1+\sigma(c_{fog}-1)+k\frac{(aN_{users}+bN_{iot})}{c_{fog}}}$
18	// Monitor value of performance parameters every
	// t interval
19	if($N_q > N_t \| U > U_t$)
20	alert("More resources need to added to be added to handle the current load")
21	// Compute no. of fog nodes required to meet
	// service demand
22	$R = \frac{\lambda_{current}}{\mu}$
23	$c = R + d\sqrt{R}$
24	}
25	End

Ln: Line number

6. RESULTS AND DISCUSSIONS

The scalability functions introduced in Section 5 and M/M/c/K queue for fog nodes are implemented in Java. We have used Java Modeling Tool (JMT) for simulation of queueing network build for cloud-fog based IoT middleware Section 4. JMT is open source modeling tools used for solving a wide variety of problems modeled as queueing systems. The results Java program for FCC and its corresponding simulation in JMT are almost matching, which validates our experiment. In this section, we provide details of our experiment and simulations.

The fog has enough resources to carry out local, and real-time request processing. To evaluate the response of fog with an increase in load we have implemented scalability function. The arrival rate was increased from 1000 req/sec to 5000 req/sec, the number of fog nodes used to handle request requests was increased from 1 to 50. The observed throughput is given in Fig. 4(a). In the hierarchical organization of the cloud-fog based IoT system, the coherency overhead is less, hence the scalability function is given (14) shows significant improvement compared to USL. But there is coherency introduced because of interaction among middleware modules, so the scalability achieved is sub-linear. Compared to the ideal throughput is 5000 req/sec, Amdahl's equation achieves 4016 req/sec while the scalability function introduced in this paper achieves only 3508 req/sec.

Table 4: Specification of parameter values for simulation

Parameter	Fog	Cloud
λ_{fog}	1000(1-P_K)	250
λ_{local_user}	250	75
λ_{IoT_data}	250	175
K	100	NA
μ_{fog}	800	NA
c_{fog}	3	NA
$P_{fog,cloud}$	0.5	NA
$\lambda_{cloud,}$	500	1000
μ_{cloud}	NA	7000
c_{loud}	NA	10
$P_{cloud,fog}$	NA	0.75

The network of queues consisting of fog and cloud servers was built using the JSIM tool of JMT. The simulation started with an overall 1000 requests arriving per second. The parameter important for simulation and their values are given in Table 4. There are two category of traffic entering local users request and remote users'

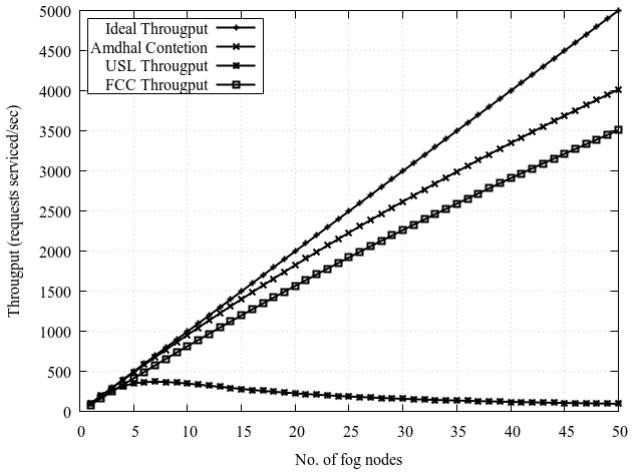
Volume: 11 Issue: 01 Pages: 4162-4170(2019) ISSN: 0975-0290

request. The requests entering into system at cloud are by remote users λ cloud at arrival rate of 1000 req/sec. The requests are processed at cloud leave system with 0.3 probability or forwarded to fog nodes with 0.7 probability.

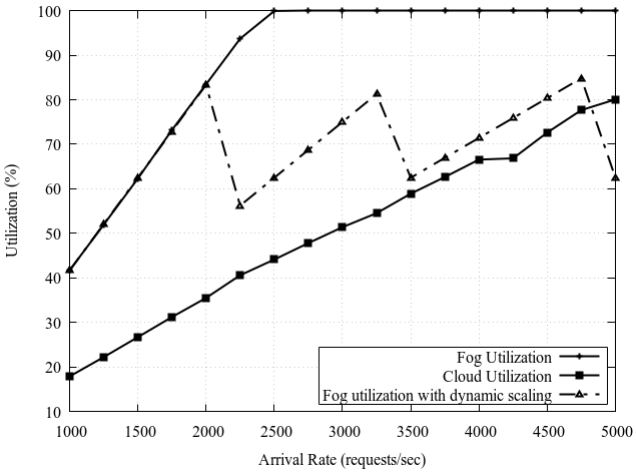
We used 10 physical servers (PS) at cloud data center, each can accommodate maximum 7 fog nodes, and each of fog nodes is capable of handling 100 requests per second. The total service rate at cloud is 7000 requests per second. At fog there are 3 fog nodes, each can process 800 incoming requests per second. The total service rate at fog is 2400 requests per second and queue capacity is 100 requests/sec.

Using what-if analysis feature of JMT, arrival rate was increased from initial 1000 req/sec to 5000 req/sec with increment of 250 in every step. Increase in utilization of fog nodes at fog and cloud is shown in Fig. 4(b), with an increase in arrival rate, utilization increases in proportion. At fog nodes since processing capacity is limited all nodes reach 100% utilization at an arrival rate of 2400 requests per second. For all arrival rate above 2400 req/sec the requests were forwarded to cloud as shown in Fig. 4(c), at 5000 req/sec 2581 requests for worded per second, the forwarding rate was rate was 51.64%, which indicated that fog computing center was overloaded and unable to handle current workload became, hence became bottleneck. In the case of cloud computing center, the utilization was 80% even at the maximum arrival rate.

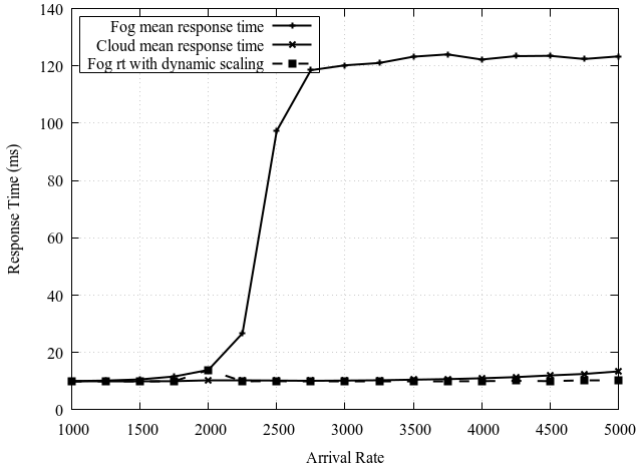
The mean response time increases with an increase in arrival rate as shown in Fig. 4(c). The SLA for maximum



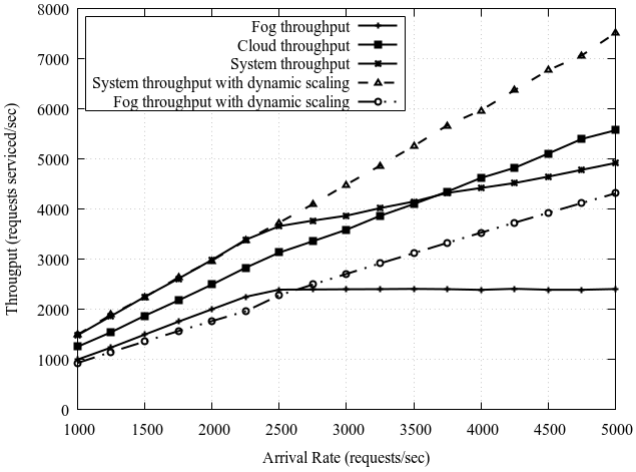
(a) Throughput comparison of Ideal, Amdahl's Contention USL and IoT scalability function



(b)CPU Utilization with increase in arrival rate



(c) Mean response time



(d)Throughput

Figure 4: Scalability and performance analysis

response time was set to 20 ms (it is time spent by request in the system, end to end response time which user will experience requires considering transmission delay of LAN and WAN based on the category of a user). At 2400 req/sec drastically rise from 15ms to 120ms. It was necessary to add more fog nodes in order to meet given SLA. At cloud computing center the response time was less than 20ms even at 5000 req/sec. The throughput in terms of the number of requests fulfilled per second is shown in Fig. 4(d). Throughput goes on increasing with arrival rate.

Volume: 11 Issue: 01 Pages: 4162-4170(2019) ISSN: 0975-0290

Fog computing center was unable to handle the increase in arrival rate from 1000 req/sec to 5000 req/sec. The utilization was full, response time grown unpredictably, throughput was saturated and the drop rate was 51.64%. JMT simulation provides very useful insight for understanding the working of cloud-fog based IoT middleware. However, JMT lacks dynamic resource allocation, which was required in case of fog computing center at the arrival rate of 2400 req/sec. In order to handle this dynamic behavior we have implemented dynamic scaling algorithm introduced in Section 5.1, the results are as shown in Fig. 4(b)-(d). Initially, 3 fog nodes were used to handle requests. As arrival rate of requests increased utilization increased proportionally at 2000 req/sec utilization is 83.33%, new resource demand at this arrival rate was calculated and two more additional fog nodes were deployed, in this fashion fog computing center was able to handle growing resource demands as shown in Fig. 4(b). For all the experiments and simulations we used Dell Inspiron 15 with i7 processor, 8GB memory and Ubuntu 18.04.

7. CONCLUSION

In order to address the tremendous growth the IoT network is witnessing, scalability is one of the pressing issues must be resolved by researchers. The hierarchical organization of cloud-fog based IoT system enables scalability by the architectural design itself. However, the rapid increase in load on fog by factor multiple of current workload, fog computing center will become the bottleneck. In this paper, we have systematically presented the scalability analysis of fog based Internet of Things middleware. We first modeled the IoT system using a queueing network, based on the foundation of this model; scalability function is presented which considers contention and coherency overhead which limits scalability.

We were interested in finding the effect of an increase in the number of users or IoT devices on throughput, utilization, response time of fog and cloud. These parameters are crucial for answering the questions related to the scalability of IoT middleware. We conducted a simulation using JMT simulator. The results of simulation and implementation show that the proposed model scales sub-linearly. These results are general enough to apply in any setting of IoT middleware and application set up, it does not depend on any particular suite of technology e.g. if fog nodes is replaced by docker container. When fog center becomes overloaded, investigating scalability by offloading requests to neighboring fog nodes rather than remote cloud will be our future work.

REFERENCES

[1] D. Evans, The Internet of Things: How the next evolution of the Internet is changing everything, *white paper, Cisco*, 2011.

[2] O.Sezer, E. Dogdu, A. Ozbayoglu, Context-Aware computing, learning, and big data in Internet of things: a survey, *IEEE IoT Journal, 5*(1), 2018, 1-27.

[3] D. Rathod, G. Chowdhary, Survey of middlewares for Internet of things, *Proc. IEEE Int. Conf. on Recent Trends in Advanced Computing: CPS*, Chennai, India, Sep. 10-11, 2018, 129-135.

[4] M. Razzaque, M. Milojevic-Jevric, A. Palade, S. Clarke, Middleware for Internet of things: a survey, *IEEE IoT Journal, 3*(1), 2016, 70-95.

[5] M. Harchol-Balter, *Performance modeling and design of computer systems:queueing theory in action*, Cambridge University Press, 2013.

[6] M. Gomes, R. Righi, C. da Costa, Internet of things scalability: analyzing the bottlenecks and proposing alternatives, *Proc. 6th Int. Congress on Ultra Modern Telecom. and Control Systems and Workshops (ICUMT)*, 2014, 269-276.

[7] V. Soto, *Performance evaluation of scalable and distributed IoT platforms for smart regions*, Masters thesis, Dept. of CS, Ele. and Space Engg., Luleå University of Technology, Skellefteå, 2017.

[8] Fiware: the open source platform for smart digital future, https://www.fiware.org/, (accessed 5 April 2019).

[9] A. Esposte, E. Santana, L. Kanashiro, F. Costa, K. Braghetto, N. Lago, F. Kon, Design and evaluation of a scalable smart city software platform with large-scale simulations, *Future Gen. Comp. Sys. 93,* 2018, 427-441.

[10] S. Maheshwari, D. Raychaudhuri, I. Seskar, F. Bronzino, Scalability and performance evaluation of edge cloud systems for latency constrained applications, *Proc. 3rd ACM/IEEE Symposium on Edge Computing*, 2018, 286-289.

[11] Q. Fan, N. Ansari, Towards workload balancing in fog computing empowered IoT, *IEEE Tran. on Network Sci. and Engg.*, Early Access.

[12] B. Liu, X. Chang, B. Liu, Z. Chen, Performance analysis model for fog services under multiple resource types, *Proc. IEEE Int. Conf. on Dependable Sys.and their Applications*, 2017, 110-117.

[13] R. Pinciroli, M. Gribaudo, M. Roveri, G. Serazzi, Capacity Planning of fog computing infrastructures for smart monitoring, *New Frontiers in Quantitative Methods in Informatics, Comm. in Computer and Information Sci. 825,* 2018, 72-81.

[14] S. Duttagupta, M. Kumar, R. Ranjan, M. Nambiar, Performance prediction of IoT application an experimental analysis, *Proc. ACM 6th Int. Conf. on the*

Volume: 11 Issue: 01 Pages: 4162-4170(2019) ISSN: 0975-0290

IoT, 2016, 43-51.

[15] J. Huang, S. Li, Y. Chen, J. Chen, Performance modeling and analysis for IoT services, *Int. J. Web and Grid Services, 14*(2), 2018, 146-169.

[16] S. Kafhali, K. Salah, Efficient and dynamic scaling of fog nodes for IoT devices, Springer *J Supercomp. 73*(12), 2017, 5261-5284.

[17] C. Mouradian, D. Naboulsi, S. Yangui, R.H. Glitho, M. J. Morrow, P. A. Polakos, A comprehensive survey on fog computing: state-of-the-art and research challenges, *IEEE Comm. Surveys and Tutorials 20*(1) 2018, 416-464.

[18] Fog computing and the Internet of things: extend the cloud to where the things are, Cisco, White paper, 2015.

[19] A. Bondi, Characteristics of scalability and their impact on performance, *Proc. 2nd int. workshop on sw and performance*, 2000, 195-203.

[20] C. Weinstock J. Goodenough, On System Scalability, *Software Engineering Institute*, Technical Note, CMU/SEI-2006-TN-012, 2006.

[21] M. Abbott, M. Fisher, *The Art of scalability: scalable web architecture, processes, and organizations for the modern enterprise* 2nd edition, Pearson Education, 2015.

[22] B. Neuman, Scale in distributed systems, *IEEE Computer Society Press*, 1994, 1-28.

[23] [23] P. Jogalekar, M. Woodside, Evaluating the scalability of distributed systems, *IEEE Tran. on Parallel and Distributed Systems, 11*(6), 2000, 589-603.

[24] N. Gunther, *Guerrilla capacity planning: a tactical approach to planning for highly scalable applications and services* Springer-Verlag Berlin Heidelberg, 2007.

[25] Neil Gunther, Paul Puglia, KristoferTomasette, HadoopSuperlinear Scalability, *Comm. of the ACM 58*(4), 2015, 46-55.

[26] Rajkumar, H. Chandrakanth, D. Anand, T. Peter, Research Challenges andCharacteristic Features in Wireless Sensor Networks, Int. J. Advanced Networking and Applications vol.09(01), 2017, pp: 3321-3328.

[27] Patel Rinkuben N., N. Bhatt, Design and Implementation of QoS Aware Priority based MAC for Delay Sensitive Areas of WSN, Int. J. Advanced Networking and Applications, vol.09(03), 2017, pp.3411-3420.